HỆ THỐNG BÀI TẬP J2ME

Bài 1 : **Vidu1**, đưa ra lời chào

```
//  Hello.java
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class TestMidlet  extends MIDlet implements CommandListener {
private Form mForm;
public TestMidlet() {
mForm = new Form("Lap trinh voi J2ME");
mForm.append(new StringItem(null, "Hello world!, MIDP!"));
mForm.addCommand(new Command("Exit", Command.EXIT, 0));
mForm.setCommandListener(this);
  }
public void startApp() {
    Display.getDisplay(this).setCurrent(mForm);
  }
  public void pauseApp() {}
  public void destroyApp(boolean unconditional) {}
  public void commandAction(Command c, Displayable s) {
    notifyDestroyed();
  }
}
```

Bài 2: **TaoForm**, tạo một Form và chèn các đối tượng (Item) vào Form

```
// CreatForm.java
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.StringItem;
import javax.microedition.lcdui.TextField;
import javax.microedition.midlet.MIDlet;
public class CreateForm extends MIDlet {
    // The MIDlet's Display object
    protected Display display;
    // Flag indicating first call of startApp
    protected boolean started;
    protected void startApp() {
        if (!started) {
            display = Display.getDisplay(this);
            Form form = new Form("Tieu de Form");
            form.append("Chao");
            form.append("Tat ca cac ban");
            form.append("\nChung ta bat dau lam viec nao!\n Mot dong moi\n");
form.append("Day la mot dong rat dai chung ta khong viet chung tren mot dong duoc");
form.append(new TextField("Ho va ten:", "Le Thi Cham Chi", 32, TextField.ANY));
            form.append("Dia chi:");
            form.append(new TextField(null, null, 32, TextField.ANY));
            display.setCurrent(form);
            started = true;
        }
    }
    protected void pauseApp() {}
    protected void destroyApp(boolean unconditional) {}
}
```

Bài 3: **CacHanhDong**, Tạo Form với các hành động (Command): thoát (EXIT), trở lại (BACK) và gọi hành động khác

```
// ManyCommands.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class ManyCommands extends MIDlet implements CommandListener{
  private Display display;      // Reference to Display object for this MIDlet
  private Form fmMain;          // The main Form
  private TextBox tbAction;     // Textbox to show when user selects upload/download
  private Command cmExit;       // Exit the MIDlet
  private Command cmBack;       // Go "back" to the main form
  private Command cmUload;      // "upload" data - no real action done
  private Command cmDload;      // "download" data - no real action done
  public ManyCommands(){
    display = Display.getDisplay(this);
    cmExit = new Command("Exit", Command.EXIT, 1);
    cmBack = new Command("Back", Command.BACK, 1);
    cmUload = new Command("Upload", Command.SCREEN, 2);
    cmDload = new Command("Download", Command.SCREEN, 3);
    // Create the Form, add Commands, listen for events
    fmMain = new Form("Core J2ME");
    fmMain.addCommand(cmExit);
    fmMain.addCommand(cmUload);
    fmMain.addCommand(cmDload);
    fmMain.setCommandListener(this);
    // Create a Textbox, add Command, listen for events
    tbAction = new TextBox("Process Data", "Upload/download data ", 25, 0);
    tbAction.addCommand(cmBack);
    tbAction.setCommandListener(this);
  }
  // Called by application manager to start the MIDlet.
  public void startApp(){
    display.setCurrent(fmMain);
  }
  public void pauseApp(){ }
  public void destroyApp(boolean unconditional)  { }
  public void commandAction(Command c, Displayable s)  {
    if (c == cmExit){
      destroyApp(false);
      notifyDestroyed();
    }
    else if (c == cmUload || c == cmDload)
      display.setCurrent(tbAction);
    else if (c == cmBack)
      display.setCurrent(fmMain);
  }
}
```

Bài 4: **Chuoi**, dùng StringItem để viết Text lên màn hình, sau đó thay đổi Text đó (cả nhãn (setLable) và nội dung (setText).

```
// StringExample.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class StringExample extends MIDlet implements CommandListener{
  private Display display;      // Doi tuong Display
  private Form fMain;          // Main form
  private StringItem sMsg;     // StringItem
  private StringItem s1Msg;
  private Command cmChange;     // Thay doi label and message
  private Command cmExit;       // Exit the MIDlet
  public StringExample(){
    display = Display.getDisplay(this);
    // Tao chuoi van ban va commands
    sMsg = new StringItem("Hoc sinh Khoa CNTT: ", "la niem tu hao cua Hoc Vien");
    s1Msg = new StringItem("Lop C04CNTT: ","Cung the");
    cmChange = new Command("Change", Command.SCREEN, 1);
    cmExit = new Command("Exit", Command.EXIT, 1);
    // Tao Form, them Command and StringItem, listen for events
    fMain = new Form("Vi du ve nhan va chuoi van ban");
    fMain.addCommand(cmExit);
    fMain.addCommand(cmChange);
    fMain.append(sMsg);
    fMain.append(s1Msg);
    fMain.setCommandListener(this);
  }
  // Goi start the MIDlet.
  public void startApp(){
    display.setCurrent(fmMain);
  }
  public void pauseApp(){ }
  public void destroyApp(boolean unconditional){ }
  public void commandAction(Command c, Displayable s){
    if (c == cmChange){
      // change label
      sMsg.setLabel("Tuoi tuoi 20: ");
      // Change text
      sMsg.setText("La mua xuan cua cuoc doi. ");
// change label
      s1Msg.setLabel("The ky 21: ");
      // Change text
      s1Msg.setText("Phu nu khong chi don gian la ba me,ban dong y chu?? ");
      // Remove the command
      fMain.removeCommand(cmChange);
    }
    else if (c == cmExit){
      destroyApp(false);
      notifyDestroyed();
    }
  }
}
```

Bài 5: **Chuoi2**, dùng StringItem viết lên màn hình, sau đó thay đổi nó bằng cách thêm 1 hành động trên Form

```
// StringItemExample.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class StringItemExample extends MIDlet  implements CommandListener{
  private Display display;
  private Form form;
  private StringItem question;
  private Command answer;
  private Command exit;
  public StringItemExample(){
    display = Display.getDisplay(this);
    question = new StringItem("Cau hoi: ", "Hoc sinh pho thong vua hoc xong lop 12,"
              +" lieu nen thi vao truong nao o Ha Noi?");
    answer = new Command("Tra loi", Command.SCREEN, 1);
    exit = new Command("Exit", Command.EXIT, 1);
    form = new Form("THAC MAC");
    form.addCommand(exit);
    form.addCommand(answer);
    form.append(question);
    form.setCommandListener(this);
  }
  public void startApp(){
    display.setCurrent(form);
  }
  public void pauseApp(){ }
  public void destroyApp(boolean unconditional){ }
  public void commandAction(Command c, Displayable d){
    if (c == answer){
      question.setLabel("Tra loi: ");
      question.setText("Hoc Vien Cong Nghe Buu Chinh Vien Thong.");
      form.removeCommand(answer);
    }
    else if (c == exit){
     destroyApp(false);
     notifyDestroyed();
   }
  }
}
```

Bài 6: **Chuoi3**, thêm hành động Next để gọi 1 StringItem nữa, viết nội dung của các StringItem ra màn hình Toolkit

```
// StringItem1.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class StringItem1 extends MIDlet implements CommandListener{
  private Display display;      // Reference to Display object for this MIDlet
  private Form fmMain;          // The main form
  private Command cmNext;       // Next label and message
  private Command cmExit;       // Command to exit the MIDlet
  private int msgIndex;         // Index of our message text on form
```

```
  private static int count = 0; // How many times through our loop
  public StringItem1(){
    display = Display.getDisplay(this);
    // Create commands
    cmNext = new Command("Next", Command.SCREEN, 1);
    cmExit = new Command("Exit", Command.EXIT, 1);
    // Create Form, add Command & message, listen for events
    fmMain = new Form("Preferences");
    fmMain.addCommand(cmExit);
    fmMain.addCommand(cmNext);
    // Save the index location of this item
    msgIndex = fmMain.append("Ten khoa: CNTT1");
    fmMain.setCommandListener(this);
  }
  // Called by application manager to start the MIDlet.
  public void startApp(){
    display.setCurrent(fmMain);
  }
  public void pauseApp(){ }
  public void destroyApp(boolean unconditional){ }
  public void commandAction(Command c, Displayable s){
    if (c == cmNext){
      if (count++ == 0){
        //--------------------------------
        // Option # 1
        // First time through this method
        //--------------------------------
         StringItem tmpItem = (StringItem) fmMain.get(msgIndex);
         System.out.println("tmpItem.getLabel(): " + tmpItem.getLabel());
         System.out.println("tmpItem.getText(): " + tmpItem.getText());
         tmpItem.setLabel("Account #: ");  //inherited from Item class
         tmpItem.setText("1234");
      }
      else{
        //--------------------------------
        // Option # 2
        // Second time through this method
        //--------------------------------
         fmMain.set(msgIndex, new StringItem("Bo mon: ", "Cong Nghe Phan Mem"));
        // Remove the Update command
        fmMain.removeCommand(cmNext);
      }
    }
    else if (c == cmExit){
      destroyApp(false);
      notifyDestroyed();
    }
  }
}
```

Bài 7: **ONhapLieu**, dùng TextField nhập liệu (số), rồi thông báo ra màn hình Toolkit

```
// TextFieldExample.java
import javax.microedition.midlet.*;
```

```
import javax.microedition.lcdui.*;
public class TextFieldExample extends MIDlet implements CommandListener{
  private Display display;            // Doi tuong Display
  private Form fMain;         // form chinh
  private Command cmText;       // Nhan noi dung cua textfield
  private Command cmExit;       // Lenh exit the MIDlet
  private TextField tfText;     // Textfield
  public TextFieldExample(){
    display = Display.getDisplay(this);
    // Tao cac commands
    cmText = new Command("Nhap du lieu:", Command.SCREEN, 1);
    cmExit = new Command("Exit", Command.EXIT, 1);
    // Textfield so dien thoai
    tfText = new TextField("Phone:", "", 10, TextField.PHONENUMBER);
    // Create Form, add Commands and textfield, listen for events
    fMain = new Form("Vi du ve nhap lieu la so dien thoai");
    fMain.addCommand(cmExit);
    fMain.addCommand(cmText);
    fMain.append(tfText);
    fMain.setCommandListener(this);
  }
  // Loi goi start the MIDlet.
  public void startApp(){
    display.setCurrent(fMain);
  }
  public void pauseApp(){ }
  public void destroyApp(boolean unconditional){ }
  public void commandAction(Command c, Displayable s){
    if (c == cmText){
      System.out.println("TextField nhap vao la: " + tfText.getString());
    }
    else if (c == cmExit){
     destroyApp(false);
     notifyDestroyed();
    }
  }
}
```

Bài 8: **TextField1**, dùng TextField viết ra màn hình sau đó hiển thị lên thiết bị mô phỏng

```
// TextFieldname.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class TextFieldname extends MIDlet  implements CommandListener{
  private Display display;
  private Form form;
  private Command submit;
  private Command exit;
  private TextField textfield;
  public TextFieldname(){
   display = Display.getDisplay(this);
   submit = new Command("Submit", Command.SCREEN, 1);
   exit = new Command("Exit", Command.EXIT, 1);
   textfield = new TextField("Ho va ten:", "", 30, TextField.ANY);
```

```
   form = new Form("Nhap du lieu");
   form.addCommand(exit);
   form.addCommand(submit);
   form.append(textfield);
   form.setCommandListener(this);
  }
  public void startApp(){
   display.setCurrent(form);
  }
  public void pauseApp(){ }
  public void destroyApp(boolean unconditional){ }
  public void commandAction(Command c, Displayable s){
   if (c == submit){
     textfield.setString("Xin chao, " + textfield.getString());
     form.removeCommand(submit);
   }
   else if (c == exit){
      destroyApp(false);
      notifyDestroyed();
   }
  }
}
```

Bài 9: **Login**, nhập TextField dạng PASSWORD sau đó đưa thông báo

```
// TextPassword.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class TextPassword extends MIDlet implements CommandListener{
  private Display display;            // Doi tuong Display
  private Form fMain;          // form chinh
  private Command cmText;       // Nhan noi dung cua textfield
  private Command cmExit;       // Lenh exit the MIDlet
  private TextField pText;     // Textfield
  public TextPassword(){
    display = Display.getDisplay(this);
    // Tao cac commands
    cmText = new Command("Nhap du lieu:", Command.SCREEN, 1);
    cmExit = new Command("Exit", Command.EXIT, 1);
    // Textfield so dien thoai
    pText = new TextField("Phone:", "", 10, TextField.ANY|TextField.PASSWORD);
    // Create Form, add Commands and textfield, listen for events
    fMain = new Form("Vi du ve nhap lieu la password");
    fMain.addCommand(cmExit);
    fMain.addCommand(cmText);
    fMain.append(pText);
    fMain.setCommandListener(this);
  }
  // Loi goi start the MIDlet.
  public void startApp(){
    display.setCurrent(fMain);
  }
  public void pauseApp(){ }
  public void destroyApp(boolean unconditional){ }
```

```
  public void commandAction(Command c, Displayable s){
    if (c == cmText){
      if(pText.getString().equals("ab"))
        System.out.println("Nhap dung");
      else
        System.out.println("Nhap sai");
    }
    else if (c == cmExit){
     destroyApp(false);
     notifyDestroyed();
    }
  }
}
```

Bài 10: **DateHienThoi**, đưa ra ngày giờ hiện thời của hệ thống

```
// DateToday.java
import java.util.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
 public class DateToday extends MIDlet implements CommandListener{
  private Display display;
  private Form form;
  private Date today;
  private Command exit;
  private DateField datefield;
  public DateToday(){
    display = Display.getDisplay(this);
    form = new Form("Today's Date");
    today = new Date(System.currentTimeMillis());
    datefield = new DateField("Hom nay:", DateField.DATE_TIME);
    datefield.setDate(today);
    exit = new Command("Exit", Command.EXIT, 1);
    form.append(datefield);
    form.addCommand(exit);
    form.setCommandListener(this);
  }
  public void startApp (){
    display.setCurrent(form);
  }
  public void pauseApp(){ }
  public void destroyApp(boolean unconditional){ }
  public void commandAction(Command command, Displayable displayable){
   if (command == exit){
    destroyApp(false);
    notifyDestroyed();
   }
  }
}
```

Bài 11: **ThoiGian**, đưa ra ngày giờ hiện thời và thay đổi nó

```
// DateExample.java
```

```
import java.util.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.util.Timer;
import java.util.TimerTask;
public class DateExample extends MIDlet implements ItemStateListener, CommandListener{
  private Display display;            // Doi tuong display
  private Form fmMain;              // form chinh
  private Command cmExit;           // Exit MIDlet
  private DateField dfAlarm;        // Thanh phan DateField
  public DateExample(){
    display = Display.getDisplay(this);
    // Tao form chinh
    fmMain = new Form("DateField Test");
    // DateField voi thoi gian duoc dat hien thoi
    dfAlarm = new DateField("Thiet lap thoi gian:", DateField.DATE_TIME);
    dfAlarm.setDate(new Date());
    // Nut thoat
    cmExit = new Command("Exit", Command.EXIT, 1);
    // Them vao form va listen for events
    fmMain.append(dfAlarm);
    fmMain.addCommand(cmExit);
    fmMain.setCommandListener(this);
    fmMain.setItemStateListener(this);
  }
  public void startApp (){
    display.setCurrent(fmMain);
  }
  public void pauseApp(){ }
  public void destroyApp(boolean unconditional){ }
  public void itemStateChanged(Item item){
    System.out.println("Date field changed.");
  }
  public void commandAction(Command c, Displayable s) {
    if (c == cmExit){
      destroyApp(false);
      notifyDestroyed();
    }
  }
}
```

Bài 12: **HoanThanh**, điều chỉnh âm lượng

```
// GaugeExample.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class GaugeExample extends MIDlet implements CommandListener{
  private Display display;   // Reference to display object
  private Form fmMain;       // The main form
  private Command cmExit;    // Exit the form
  private Gauge gaVolume;    // Volume adjustment
  public GaugeExample(){
    display = Display.getDisplay(this);
    // Create the gauge and exit command
    gaVolume = new Gauge("Sound Level", true, 5, 1);
```

```
    cmExit = new Command("Exit", Command.EXIT, 1);
    // Create form, add commands, listen for events
    fmMain = new Form("");
    fmMain.addCommand(cmExit);
    fmMain.append(gaVolume);
    fmMain.setCommandListener(this);
  }
  // Called by application manager to start the MIDlet.
  public void startApp(){
    display.setCurrent(fmMain);
  }
  public void pauseApp(){ }
  public void destroyApp(boolean unconditional) { }
 public void commandAction(Command c, Displayable s) {
    if (c == cmExit){
      destroyApp(false);
      notifyDestroyed();
    }
  }
}
```

Bài 13: **NhomChon**, nhóm chọn dạng CheckBox, có thêm hành động View hiển thị mục chọn ra màn hình Toolkit

```
// ChoiceGroupExample.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class ChoiceGroupExample extends MIDlet implements ItemStateListener,
CommandListener{
  private Display display;         // Doi tuong display
  private Form fmMain;          // Main form
  private Command cmExit;       // Lenh exit the MIDlet
  private Command cmView;       // Hien thi choice selected
  private int selectAllIndex;   // Index of the "Select All" option
  private ChoiceGroup cgPrefs;  // Choice Group of preferences
  private int choiceGroupIndex; // Index of choice group on form
  public ChoiceGroupExample() {
    display = Display.getDisplay(this);
    // Tao nhom chon - multiple choice group
    cgPrefs = new ChoiceGroup("Hay chon cac yeu thich cua minh:", Choice.MULTIPLE);
    // Them options
    cgPrefs.append("Hoc tap cham chi va sang tao", null);
    cgPrefs.append("Choi the thao", null);
    cgPrefs.append("Di mua sam", null);
    selectAllIndex = cgPrefs.append("Chon tat ca", null);
    cmExit = new Command("Exit", Command.EXIT, 1);
    cmView = new Command("View", Command.SCREEN,2);
    // Tao Form, them components, listen for events
    fmMain = new Form("");
    choiceGroupIndex = fmMain.append(cgPrefs);
    fmMain.addCommand(cmExit);
    fmMain.addCommand(cmView);
    fmMain.setCommandListener(this);
    fmMain.setItemStateListener(this);
```

```
  }
  public void startApp(){
    display.setCurrent(fmMain);
  }
  public void pauseApp() { }
  public void destroyApp(boolean unconditional) { }
  public void commandAction(Command c, Displayable s){
    if (c == cmView) {
      boolean selected[] = new boolean[cgPrefs.size()];
      // Dien dau tich khi phan tu duoc chon
      cgPrefs.getSelectedFlags(selected);
      for (int i = 0; i < cgPrefs.size(); i++)
        System.out.println(cgPrefs.getString(i) +(selected[i] ? ": selected" : ": not selected"));
    }
    else if (c == cmExit)  {
      destroyApp(false);
      notifyDestroyed();
    }
  }
  public void itemStateChanged(Item item) {
    if (item == cgPrefs)   {
      // Option "Chon tat ca" da chon chua?
      if (cgPrefs.isSelected(selectAllIndex))   {
        // Dat tat ca checkboxes la true
        for (int i = 0; i < cgPrefs.size(); i++)
          cgPrefs.setSelectedIndex(i, true);
        // Bo viec chon nho "Chon tat ca"
        cgPrefs.setSelectedIndex(selectAllIndex, false);
      }
    }
  }
}
```

Bài 14: **NhomChonRadio**, nhóm chọn dạng Radio thêm hành động Submit để thông báo ra màn hình hiển thị mục chọn

```
// RadioGroup.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class RadioGroup extends MIDlet implements CommandListener{
  private Display display;
  private Form form;
  private Command exit;
  private Command process;
  private ChoiceGroup gender;
  private int currentIndex;
  private int genderIndex;
  public RadioGroup() {
    display = Display.getDisplay(this);
    gender = new ChoiceGroup("Gioi tinh:", Choice.EXCLUSIVE);
    gender.append("Nu", null);
    gender.append("Nam", null);
    currentIndex = gender.append("Loai khac ", null);
    gender.setSelectedIndex(currentIndex, true);
```

```
    exit = new Command("Exit", Command.EXIT, 1);
    process = new Command("Submit", Command.SCREEN,2);
    form = new Form("Chon gioi tinh");
    genderIndex = form.append(gender);
    form.addCommand(exit);
    form.addCommand(process);
    form.setCommandListener(this);
  }
  public void startApp() {
    display.setCurrent(form);
  }
  public void pauseApp() {}
  public void destroyApp(boolean unconditional) {}
  public void commandAction(Command c, Displayable displayable) {
    if (c == exit) {
        destroyApp(false);
        notifyDestroyed();
    }
    else if (c == process){
      currentIndex = gender.getSelectedIndex();
      StringItem message = new StringItem("Gioi tinh: ", gender.getString(currentIndex));
      form.append(message);
      form.delete(genderIndex);
      form.removeCommand(process);
    }
  }
}
```

Bài 15: **NhomChonRadio1**, nhóm chọn dạng Radio, hiển thị mục chọn lên thiết bị

```
// Radio1.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class Radio1 extends MIDlet implements ItemStateListener, CommandListener{
  private Display display;
  private Form form;
  private Command exit;
  //private Item selection;
  private ChoiceGroup radioButtons;
  private int defaultIndex;
  private int radioButtonsIndex;
  public Radio1() {
    display = Display.getDisplay(this);
    radioButtons = new ChoiceGroup( "Chon mau ban yeu thich:", Choice.EXCLUSIVE);
    radioButtons.append("Red", null);
    radioButtons.append("White", null);
    radioButtons.append("Blue", null);
    radioButtons.append("Green", null);
    defaultIndex = radioButtons.append("All", null);
    radioButtons.setSelectedIndex(defaultIndex, true);
    exit = new Command("Exit", Command.EXIT, 1);
    form = new Form("");
    radioButtonsIndex = form.append(radioButtons);
    form.addCommand(exit);
```

```
    form.setCommandListener(this);
    form.setItemStateListener(this);
  }
  public void startApp(){
     display.setCurrent(form);
  }
  public void pauseApp(){}
  public void destroyApp(boolean unconditional){}
  public void commandAction(Command c, Displayable s)  {
    if (c == exit){
       destroyApp(true);
       notifyDestroyed();
    }
  }
  public void itemStateChanged(Item item){
   if (item == radioButtons){
      StringItem msg = new StringItem("Mau yeu thich cua ban la: ",
            radioButtons.getString(radioButtons.getSelectedIndex()));
      form.append(msg);
   }
  }
}
```

Bài 16: **HinhAnh**, đưa ảnh ra màn hình hiển thị (hỗ trợ ảnh .png)

```
// ImageExample.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class ImageExample extends MIDlet  implements CommandListener{
  private Display display;     // Reference to Display object
  private Form fmMain;         // The main form
  private Command cmExit;      // Command to exit the MIDlet
  public ImageExample(){
    display = Display.getDisplay(this);
    cmExit = new Command("Exit", Command.EXIT, 1);
    fmMain = new Form("");
    fmMain.addCommand(cmExit);
    fmMain.setCommandListener(this);
    try{
      // Read the appropriate image based on color support
      Image im = Image.createImage((display.isColor()) ?"/terrain1.png":"/terrain2.png");
      fmMain.append(new ImageItem(null, im, ImageItem.LAYOUT_CENTER, null));
      display.setCurrent(fmMain);
    }
    catch (java.io.IOException e){
      System.err.println("Unable to locate or read .png file");
    }
  }
  public void startApp(){
    display.setCurrent(fmMain);
  }
  public void pauseApp(){}
  public void destroyApp(boolean unconditional){}
  public void commandAction(Command c, Displayable s){
```

```
    if (c == cmExit){
      destroyApp(false);
      notifyDestroyed();
    }
  }
}
```

Bài 17: DanhSach, danh sách với icon đi kèm

```
// ListExample.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class ListExample extends MIDlet implements CommandListener{
  private Display display;          // Doi tuong Display
  private List lsDocument;     // Main list
  private Command cmExit;      // Lenh exit
  public ListExample(){
    display = Display.getDisplay(this);
    // Tao lenh Exit
    cmExit = new Command("Exit", Command.EXIT, 1);
    try{
      // Tao mang cac doi tuong anh
      Image images[] = {Image.createImage("/Sau.png"),Image.createImage("/Truoc.png"),
                  Image.createImage("/Moi.png")};
      // Create array of corresponding string objects
      String options[] = {"Sau", "Truoc", "Moi"};
   // Create list using arrays, add commands, listen for events
   lsDocument = new List("Document Option:", List.IMPLICIT, options, images);
   // If you have no images, use this line to create the list
       //      lsDocument = new List("Document Option:", List.IMPLICIT, options, null);
   lsDocument.addCommand(cmExit);
   lsDocument.setCommandListener(this);
 }
 catch (java.io.IOException e){
   System.err.println("Unable to locate or read .png file");
 }
        }
  public void startApp(){
    display.setCurrent(lsDocument);
  }
  public void pauseApp(){}
  public void destroyApp(boolean unconditional){}
  public void commandAction(Command c, Displayable s){
    // If an implicit list generated the event
    if (c == List.SELECT_COMMAND){
      switch (lsDocument.getSelectedIndex()){
        case 0:
          System.out.println("Option Sau la: selected");
          break;
        case 1:
          System.out.println("Option Truoc la: selected");
          break;
        case 2:
          System.out.println("Option Moi la: selected");
```

```
        break;
      }
    }
    else if (c == cmExit){
      destroyApp(false);
      notifyDestroyed();
    }
  }
}
```

Bài 18: **DanhSachCheckBox**, danh sách dạng CheckBox với thông báo chọn (kèm Ticker)

```
// ListCheckBox.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class ListCheckBox extends MIDlet implements CommandListener{
  private Display display;
  private Command exit;
  private Command submit;
  private List list;
  public ListCheckBox(){
    display = Display.getDisplay(this);
    list = new List("Select Media", List.MULTIPLE);
    list.append("Books", null);
    list.append("Movies", null);
    list.append("Television", null);
    list.append("Radio", null);
    exit = new Command("Exit", Command.EXIT, 1);
    submit = new Command("Submit", Command.SCREEN,2);
    list.addCommand(exit);
    list.addCommand(submit);
    list.setCommandListener(this);
  }
  public void startApp(){
    display.setCurrent(list);
  }
  public void pauseApp(){}
  public void destroyApp(boolean unconditional){}
  public void commandAction(Command command, Displayable Displayable){
    if (command == submit){
      boolean choice[] = new boolean[list.size()];
      StringBuffer message = new StringBuffer();
      list.getSelectedFlags(choice);
      for (int x = 0; x < choice.length; x++){
        if (choice[x]){
          message.append(list.getString(x));
          message.append(" ");
        }
      }
      Alert alert = new Alert("Choice", message.toString(),null, null);
      alert.setTimeout(Alert.FOREVER);
      alert.setType(AlertType.INFO);
      display.setCurrent(alert);
      list.removeCommand(submit);
```

```
    }
    else if (command == exit){
      destroyApp(false);
      notifyDestroyed();
    }
  }
}
```

Bài 19: **DanhSach1**, danh sách cùng các mode (listType) của nó

```
// ListDemo.java
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class ListDemo extends MIDlet {
    private Display          display;
    private int              mode = List.IMPLICIT;
    private Command exitCommand = new Command( "Exit",Command.SCREEN, 2 );
    private Command selectCommand = new Command( "Select",Command.OK, 1 );
    private Command nextCommand = new Command( "Next",Command.SCREEN, 2 );
    public ListDemo(){}
    protected void destroyApp( boolean unconditional ) throws MIDletStateChangeException {
        exitMIDlet();
    }
    protected void pauseApp(){}
    protected void startApp() throws
    MIDletStateChangeException {
        if( display == null ){ // first time called...
            initMIDlet();
        }
    }
    private void initMIDlet(){
        display = Display.getDisplay( this );
        display.setCurrent( new SampleList( mode ) );
    }
    public void exitMIDlet(){
        notifyDestroyed();
    }
    public static final String[] items = {
        "First", "Second", "Third", "Fourth"
    };
    class SampleList extends List implement  CommandListener {
        private int mode;
        SampleList( int mode ){
            super( "", mode, items, null );
            addCommand( exitCommand );
            addCommand( selectCommand );
            addCommand( nextCommand );
            setCommandListener( this );
            switch( mode ){
                case IMPLICIT:
                    setTitle( "Implicit" );
                    break;
                case EXCLUSIVE:
                    setTitle( "Exclusive" );
```

```
            break;
        case MULTIPLE:
            setTitle( "Multiple" );
            break;
    }
    this.mode = mode;
}
public void commandAction( Command c,  Displayable d ){
    if( c == exitCommand ){
        exitMIDlet();
    } else if( c == selectCommand ){
        showSelection( false );
    } else if( c == SELECT_COMMAND ){
        showSelection( true );
    } else if( c == nextCommand ){
        if( mode == List.IMPLICIT ){
            mode = List.EXCLUSIVE;
        } else if( mode == List.EXCLUSIVE ){
            mode = List.MULTIPLE;
        } else {
            mode = List.IMPLICIT;
        }
        display.setCurrent( new SampleList( mode ) );
    }
}
private void showSelection( boolean implicit ){
    Alert alert = new Alert( implicit ? "Implicit Selection": "Explicit Selection" );
    StringBuffer buf = new StringBuffer();
    if( mode == MULTIPLE ){
        boolean[] selected = new boolean[ size() ];
        getSelectedFlags( selected );
        for( int i = 0; i < selected.length; ++i ){
            if( selected[i] ){
                if( buf.length() == 0 ){
                    buf.append(
                     "You selected: " );
                } else {
                    buf.append( ", " );
                }
                buf.append( getString( i ) );
            }
        }
        if( buf.length() == 0 ){
            buf.append( "No items are selected." );
        }
    } else {
        buf.append( "You selected " );
        buf.append( getString(
              getSelectedIndex() ) );
    }
    alert.setString( buf.toString() );
    alert.setTimeout( Alert.FOREVER );
    display.setCurrent( alert,display.getCurrent() );
}
```

```
    }
}
```

Bài 20: HelloTextBox, dùng TextBox viết ra màn hình

```
// HelloTextBox.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class HelloTextBox extends MIDlet{
    private Display display;    // The display for this MIDlet
    public HelloTextBox() {
        display = Display.getDisplay(this);
    }
    /**
     * Start up the Hello MIDlet by creating the TextBox
     */
    public void startApp() {
        TextBox t = new TextBox("LOI CHAO", "Hello everybody, you have a lucky day and much
money!", 256, 0);
        display.setCurrent(t);
    }
    public void pauseApp() {}
    public void destroyApp(boolean unconditional) {}
}
```

Bài 21: NhapTextBox, nhập dữ liệu dùng TextBox

```
// TextBox1.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class TextBox1 extends MIDlet implements CommandListener{
  private Display display;
  private TextBox textbox;
  private Command submit;
  private Command exit;
  public TextBox1(){
   display = Display.getDisplay(this);
   submit = new Command("Submit", Command.SCREEN, 1);
   exit = new Command("Exit", Command.EXIT, 1);
   textbox = new TextBox("Enter your name ", "", 30, TextField.ANY);
   textbox.addCommand(exit);
   textbox.addCommand(submit);
   textbox.setCommandListener(this);
  }
  public void startApp(){
   display.setCurrent(textbox);
  }
  public void pauseApp(){}
  public void destroyApp(boolean unconditional){}
  public void commandAction(Command command, Displayable displayable){
   if (command == submit){
     textbox.setString("Hello, " + textbox.getString());
     textbox.removeCommand(submit);
```

```
  }
  else if (command == exit){
    destroyApp(false);
    notifyDestroyed();
  }
 }
}
```

Bài 22: **ThongBao1**, đưa thông báo ra màn hình

```
// AlertExample.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class AlertExample extends MIDlet{
   private Display display;    // The display for this MIDlet
  private Alert myAlert = null;
   public AlertExample() {}
   public void startApp() {
     display = Display.getDisplay(this);
    TextBox t = new TextBox("Hello", "Hello all of you!", 256, 0);
    display.setCurrent(t);
    System.out.println( "Gonna create Alert.." );
    createAlert();
  }
  /**
  * Puts up an Alert with an Image
  */
  private void createAlert() {
    myAlert = new Alert("Thong diep");
    String[] alertString = { " Thong diep gui toi cac ban!" };
    myAlert.setTimeout(Alert.FOREVER);
     // Add an image to Alert
     if (display.numColors() > 2){
 String icon = (display.isColor()) ?"/JavaPowered-8.png" : "/JavaPowered-2.png";
        try {
          Image image = Image.createImage( icon );
          if (image != null){
             myAlert.setImage(image);
             System.out.println( "Image created and added to alert.. " );
          }
          else{
            System.out.println( "No Image created... " );
          }
          // Add string to Alert
          for ( int i = 0; i < alertString.length; i++ ) {
            myAlert.setString( alertString[i] );
          }
          if ( myAlert != null ) {
           display.setCurrent( myAlert );
          }
        }
        catch( Exception e ){
          System.out.println( "Exception in CreateImage() " );
        }
```

```
    }
   }
   public void pauseApp() {}
   public void destroyApp(boolean unconditional) {}
}
```

Bài 23: **ThongBao2**, đưa 2 thông báo ra màn hình

```
//  MultiAlert.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class MultiAlert extends MIDlet    implements CommandListener {
  private Display mDisplay;
  private TextBox mTextBox;
  private Alert mTimedAlert;
  private Alert mModalAlert;
  private Command mAboutCommand, mGoCommand, mExitCommand;
  public MultiAlert() {
    mAboutCommand = new Command("About", Command.SCREEN, 1);
    mGoCommand = new Command("Go", Command.SCREEN, 1);
    mExitCommand = new Command("Exit", Command.EXIT, 2);
    mTextBox = new TextBox("TwoAlerts", "", 32, TextField.ANY);
    mTextBox.addCommand(mAboutCommand);
    mTextBox.addCommand(mGoCommand);
    mTextBox.addCommand(mExitCommand);
    mTextBox.setCommandListener(this);
    mTimedAlert = new Alert("TAT DUONG", "Hien gio duong Chua Boc dang bi tac nghen!",
       null,AlertType.INFO); mModalAlert = new Alert("THONG BAO",
       "Cac thay co giao cung cac em hoc sinh than men,"
       +" toi thanh that chia buon la Tet con lon nam nay phai nghi hoc hoi
dai .",null,AlertType.INFO);
    mModalAlert.setTimeout(Alert.FOREVER);
  }
  public void startApp() {
    mDisplay = Display.getDisplay(this);
    mDisplay.setCurrent(mTextBox);
  }
  public void pauseApp() {}
  public void destroyApp(boolean unconditional) {}
  public void commandAction(Command c, Displayable s) {
    if (c == mAboutCommand)
      mDisplay.setCurrent(mModalAlert);
    else if (c == mGoCommand)
      mDisplay.setCurrent(mTimedAlert, mTextBox);
    else if (c == mExitCommand)
      notifyDestroyed();
  }
}
```

Bài 24: **HopThoaiBao**, đưa ra các dạng của dạng thông báo

```
// AlertExample.java
import javax.microedition.midlet.*;
```

```
import javax.microedition.lcdui.*;
public class AlertExample extends MIDlet implements ItemStateListener, CommandListener {
  private Display display;          // Doi tuong display
  private Form fmMain;         // form chinh
  private Command cmExit;      // Lenh exit the MIDlet
  private ChoiceGroup cgSound; // Nhom chon:Choice group
  public AlertExample(){
    display = Display.getDisplay(this);
    // Tao nhom chon la cac exclusive (radio) (choice group)
    cgSound = new ChoiceGroup("Choose a sound", Choice.EXCLUSIVE);
    // Them cac muc chon ma khong co hinh anh
    cgSound.append("Info", null);
    cgSound.append("Confirmation", null);
    cgSound.append("Warning", null);
    cgSound.append("Alarm", null);
    cgSound.append("Error", null);
    cmExit = new Command("Exit", Command.EXIT, 1);
    // Tao Form, them cac components, listen for events
    fmMain = new Form("");
    fmMain.append(cgSound);
    fmMain.addCommand(cmExit);
    fmMain.setCommandListener(this);
    fmMain.setItemStateListener(this);
  }
  public void startApp(){
    display.setCurrent(fmMain);
  }
  public void pauseApp(){ }
  public void destroyApp(boolean unconditional){ }
  public void commandAction(Command c, Displayable s){
    if (c == cmExit){
              destroyApp(false);
              notifyDestroyed();
             }
  }
  public void itemStateChanged(Item item){
    Alert al = null;
    switch (cgSound.getSelectedIndex()){
      case 0:
        al = new Alert("Alert sound", "Info sound",   null, AlertType.INFO);
        break;
      case 1:
        al = new Alert("Alert sound", "Confirmation sound",   null, AlertType.INFO);
        break;
      case 2:
        al = new Alert("Alert sound", "Warning sound",  null, AlertType.INFO);
        break;
      case 3:
        al = new Alert("Alert sound", "Alarm sound",     null, AlertType.INFO);
        break;
      case 4:
        al = new Alert("Alert sound", "Error sound",     null, AlertType.INFO);
        break;
    }
```

```
    if (al != null){
      // Doi nguoi dung doc thong bao
      al.setTimeout(Alert.FOREVER);
      // Hien thi thong bao, chuyen ve form chinh khi chon done
      display.setCurrent(al, fmMain);
    }
  }
}
```

Bài 25: ChuoiChay, xuất hiện dòng chữ chạy trang trí

```
// TickerExample.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class TickerExample extends MIDlet implements CommandListener{
  private Display display;   // Doi tuong Display
  private List lsDepartments;   // Khoa chon
  private Ticker tkStudy;     // Dong chu chay
  private Command cmExit;    // Lenh exit the MIDlet
  public TickerExample(){
    display = Display.getDisplay(this);
    cmExit = new Command("Exit", Command.SCREEN, 1);
    tkStudy = new Ticker("Hoc Vien Cong Nghe Buu Chinh Vien Thong");
    lsDepartments = new List("Products", Choice.IMPLICIT);
    lsDepartments.append("Cong Nghe Thong Tin", null);
    lsDepartments.append("Vien Thong", null);
    lsDepartments.append("Quan Tri Kinh Doanh", null);
    lsDepartments.addCommand(cmExit);
    lsDepartments.setCommandListener(this);
    lsDepartments.setTicker(tkStudy);
  }
  public void startApp(){
    display.setCurrent(lsDepartments);
  }
  public void pauseApp(){}
  public void destroyApp(boolean unconditional){}
  public void commandAction(Command c, Displayable s){
    if (c == List.SELECT_COMMAND){
      switch (lsDepartments.getSelectedIndex()){
        case 0:
          System.out.println("Cong Nghe Thong tin la: selected");
          break;
        case 1:
          System.out.println("Vien Thong la: selected");
          break;
        case 2:
          System.out.println("Quan Tri Kinh Doanh la: selected");
          break;
      }
    }
    else if (c == cmExit){
      destroyApp(true);
      notifyDestroyed();
    }
```

```
  }
}
```

Bài 26: Ticker1, dòng Ticker trang trí và thông báo mục chọn

```
// TickerExample.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class TickerExample extends MIDlet implements CommandListener{
  private Display display;    // Reference to Display object
  private List cla;   // Main productlist
  private Alert alHelp;      // Alert to show text and image
  private Ticker tkUni;     // Ticker of what's on sale
  private Command cmExit;    // Command to exit the MIDlet
  public TickerExample() {
    display = Display.getDisplay(this);
    cmExit = new Command("Exit", Command.SCREEN, 1);
    tkUni = new Ticker("HOC VIEN CONG NGHE BUU CHINH THONG");
    cla = new List("Cac Khoa:", Choice.IMPLICIT);
    cla.append("Quan Tri Kinh Doanh", null);
    cla.append("Vien Thong", null);
    cla.append("Cong Nghe Thong Tin", null);
    cla.addCommand(cmExit);
    cla.setCommandListener(this);
    cla.setTicker(tkUni);
  }
  public void startApp(){
    display.setCurrent(cla);
  }
  public void pauseApp() {}
  public void destroyApp(boolean unconditional)  {}
  public void showAlert(){
    try{
      // Create an image
      Image im = Image.createImage("/java.png");
      // Create Alert, add text and image, no sound
      alHelp = new Alert("Co hoi cho cac ban", "Con cho gi nua, hay co gang hoc tap!", im, null);
      alHelp.setTimeout(Alert.FOREVER);
      alHelp.setTicker(tkUni);
    }
    catch(Exception e){
      System.out.println("Unable to read png image.");
    }
       // Display the Alert. Once dismissed, return to product list
    display.setCurrent(alHelp, cla);
  }
  public void commandAction(Command c, Displayable s){
       if (c == List.SELECT_COMMAND){
      switch (cla.getSelectedIndex()){
        case 0:
          System.out.println("Ban chon Khoa Quan Tri Kinh Doanh");
          break;
        case 1:
          System.out.println("Ban chon Khoa Vien Thong");
```

```
            break;
          case 2:
            showAlert();
            break;
        }
      }
      else if (c == cmExit){
        destroyApp(true);
        notifyDestroyed();
      }
    }
}
```

Bài 27: **KeyEvents**, hiển thị tên của phím ấn

```
//  KeyEvents.java
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Graphics;
import javax.microedition.midlet.MIDlet;
public class KeyEvents extends MIDlet  implements CommandListener {
    // The MIDlet's Display object
    private Display display;
    // Flag indicating first call of startApp
    protected boolean started;
    // Exit command
    private Command exitCommand;
    protected void startApp() {
        if (!started) {
            display = Display.getDisplay(this);
            Canvas canvas = new EventsCanvas();
            exitCommand = new Command("Exit", Command.EXIT, 0);
            canvas.addCommand(exitCommand);
            canvas.setCommandListener(this);
            display.setCurrent(canvas);
            started = true;
        }
    }
    protected void pauseApp() {}
    protected void destroyApp(boolean unconditional) {}
    public void commandAction(Command c, Displayable d) {
        if (c == exitCommand) {
            // Exit. No need to call destroyApp
            // because it is empty.
            notifyDestroyed();
        }
    }
}
class EventsCanvas extends Canvas {
    static int[] keyCodes = {KEY_NUM0, KEY_NUM1, KEY_NUM2, KEY_NUM3, KEY_NUM4,
                             KEY_NUM5, KEY_NUM6, KEY_NUM7, KEY_NUM8, KEY_NUM9,
```

```
                    KEY_POUND, KEY_STAR};
  static String[] keyNames = {"KEY_NUM0", "KEY_NUM1", "KEY_NUM2", "KEY_NUM3",
"KEY_NUM4",
                    "KEY_NUM5", "KEY_NUM6", "KEY_NUM7", "KEY_NUM8",
"KEY_NUM9",
                    "KEY_POUND", "KEY_STAR"};
  static int[] gameActions = {
                    UP, DOWN, LEFT, RIGHT, FIRE,
                    GAME_A, GAME_B, GAME_C, GAME_D};
  static String[] gameNames = {
                    "UP", "DOWN", "LEFT", "RIGHT", "FIRE",
                    "GAME_A", "GAME_B", "GAME_C", "GAME_D" };
  int lastKeyCode = 0;
  int lastX;
  int lastY;
  boolean pointer;
  protected void keyPressed(int keyCode) {
    lastKeyCode = keyCode;
    repaint();
  }
   protected void keyRepeated(int keyCode) {
    lastKeyCode = keyCode;
    repaint();
  }
   protected void keyReleased(int keyCode) {
    lastKeyCode = 0;
    repaint();
  }
  protected void pointerPressed(int x, int y) {
    lastX = x;
    lastY = y;
    pointer = true;
    repaint();
  }
  protected void pointerDragged(int x, int y) {
    lastX = x;
    lastY = y;
    pointer = true;
    repaint();
  }
  protected void pointerReleased(int x, int y) {
    pointer = false;
    repaint();
  }
  protected void paint(Graphics g) {
    g.setColor(0xffffff);
    g.fillRect(0, 0, getWidth(), getHeight());
    g.setColor(0);
    if (lastKeyCode != 0) {
      String keyText = "keyCode " + lastKeyCode;
      String keyName = null;
      // See if it is a standard key
      for (int i = 0; i < keyCodes.length; i++) {
        if (lastKeyCode == keyCodes[i]) {
```

```
                keyName = keyNames[i];
                break;
             }
          }
          if (keyName == null) {
             // See if it is a game action
             for (int i = 0; i < gameActions.length; i++) {
                if (lastKeyCode == getKeyCode(gameActions[i])) {
                   keyName = gameNames[i];
                   break;
                }
             }
          }
          g.drawString(keyText, getWidth()/2, getHeight()/2,
                       Graphics.BASELINE|Graphics.HCENTER);
          if (keyName != null) {
             g.drawString(keyName, getWidth()/2, getHeight()/2 + g.getFont().getHeight(),
                       Graphics.BASELINE|Graphics.HCENTER);
          }
       } else if (pointer) {
          g.drawString("(" + lastX + ", " + lastY + ")", getWidth()/2, getHeight()/2,
                       Graphics.BASELINE|Graphics.HCENTER);
       }
    }
}
```

Bài 28: **KeyCodes**, hiển thị tên của phím dùng hàm getKeyname()

```
// KeyCodes.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class KeyCodes extends MIDlet{
  private Display  display;       // The display
  private KeyCodeCanvas canvas;   // Canvas
   public KeyCodes() {
    display = Display.getDisplay(this);
    canvas  = new KeyCodeCanvas(this);
  }
   protected void startApp() {
    display.setCurrent( canvas );
  }
   protected void pauseApp() { }
  protected void destroyApp( boolean unconditional )  { }
   public void exitMIDlet() {
    destroyApp(true);
    notifyDestroyed();
  }
}
/*-------------------------------------------------
* Class KeyCodeCanvas
*
* Key codes and commands for event handling
*-------------------------------------------------*/
class KeyCodeCanvas extends Canvas implements CommandListener{
```

```
  private Command cmExit;          // Exit midlet
  private String keyText = null;    // Key code text
  private KeyCodes midlet;
  /*-------------------------------------------------
  * Constructor
  *-------------------------------------------------*/
  public KeyCodeCanvas(KeyCodes midlet){
    this.midlet = midlet;
    // Create exit command & listen for events
    cmExit = new Command("Exit", Command.EXIT, 1);
    addCommand(cmExit);
    setCommandListener(this);
  }
  /*-------------------------------------------------
  * Paint the text representing the key code
  *-------------------------------------------------*/
  protected void paint(Graphics g)  {
    // Clear the background (to white)
    g.setColor(255, 255, 255);
    g.fillRect(0, 0, getWidth(), getHeight());
       // Set color and draw text
    if (keyText != null){
      // Draw with black pen
      g.setColor(0, 0, 0);
      // Close to the center
      g.drawString(keyText, getWidth()/2, getHeight()/2, Graphics.TOP | Graphics.HCENTER);
    }
  }
  /*-------------------------------------------------
  * Command event handling
  *-------------------------------------------------*/
  public void commandAction(Command c, Displayable d)  {
    if (c == cmExit)
      midlet.exitMIDlet();
  }
  /*-------------------------------------------------
  * Key code event handling
  *-------------------------------------------------*/
  protected void keyPressed(int keyCode)  {
    keyText = getKeyName(keyCode);
    repaint();
  }
}
```

Bài 29: KeyMIDlet, viết tên phím ra màn hình

```
//  KeyMIDlet.java
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class KeyMIDlet extends MIDlet {
  public void startApp() {
    Displayable d = new KeyCanvas();
       d.addCommand(new Command("Exit", Command.EXIT, 0));
    d.setCommandListener(new CommandListener() {
```

```
      public void commandAction(Command c, Displayable s) {
        notifyDestroyed();
      }
    });
      Display.getDisplay(this).setCurrent(d);
  }
    public void pauseApp() {}
  public void destroyApp(boolean unconditional) {}
}
// KeyCanvas.java
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class KeyCanvas extends Canvas {
  private Font mFont;
  private String mMessage = "[Press keys]";
  public KeyCanvas() {
    mFont = Font.getFont(Font.FACE_PROPORTIONAL,
      Font.STYLE_PLAIN, Font.SIZE_MEDIUM);
  }
  public void paint(Graphics g) {
    int w = getWidth();
    int h = getHeight();
    // Clear the Canvas.
    g.setGrayScale(255);//mau den
    g.fillRect(0, 0, w - 1, h - 1);
    g.setGrayScale(0);// mau trang
    g.drawRect(0, 0, w - 1, h - 1);
    g.setFont(mFont);
    int x = w / 2;
    int y = h / 2;
    g.drawString(mMessage, x, y, Graphics.BASELINE | Graphics.HCENTER);
  }
  protected void keyPressed(int keyCode) {
    int gameAction = getGameAction(keyCode);
    switch(gameAction) {
      case UP:     mMessage = "UP";           break;
      case DOWN:   mMessage = "DOWN";         break;
      case LEFT:   mMessage = "LEFT";         break;
      case RIGHT:  mMessage = "RIGHT";        break;
      case FIRE:   mMessage = "FIRE";         break;
      case GAME_A: mMessage = "GAME_A";       break;
      case GAME_B: mMessage = "GAME_B";       break;
      case GAME_C: mMessage = "GAME_C";       break;
      case GAME_D: mMessage = "GAME_D";       break;
      default:     mMessage = ""; break;
    }
    repaint();
  }
}
```

Bài 30: **VeCungCanvas**, vẽ một cung ra màn hình

```
//  DrawArcCanvas.java
import javax.microedition.midlet.*;
```

```
import javax.microedition.lcdui.*;
public class DrawArcCanvas extends MIDlet{
  private Display  display;     // The display
  private ShapesCanvas canvas;  // Canvas
  public DrawArcCanvas()  {
    display = Display.getDisplay(this);
    canvas  = new ShapesCanvas(this);
  }
  protected void startApp()  {
    display.setCurrent( canvas );
  }
  protected void pauseApp()  { }
  protected void destroyApp( boolean unconditional )  { }
  public void exitMIDlet()  {
    destroyApp(true);
    notifyDestroyed();
  }
}
//  ShapsCanvas.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
/*-------------------------------------------------
* Class ShapesCanvas
*
* Draw arcs
*-------------------------------------------------*/
class ShapesCanvas extends Canvas implements CommandListener{
  private Command cmExit;  // Exit midlet
  private DrawArcCanvas midlet;
  public ShapesCanvas(DrawArcCanvas midlet){
    this.midlet = midlet;
    // Create exit command and listen for events
    cmExit = new Command("Exit", Command.EXIT, 1);
    addCommand(cmExit);
    setCommandListener(this);
  }
  /*-------------------------------------------------
  * Draw shapes
  *-------------------------------------------------*/
  protected void paint(Graphics g)  {
    // Clear background to white
    g.setColor(255, 255, 255);
    g.fillRect(0, 0, getWidth(), getHeight());
    // Black pen
    g.setColor(0, 0, 0);
    // Start at 3 o'clock and rotate 150 degrees
    g.drawArc(10, 10, 100, 100, 0, 150);
    // Fill the arc
    g.fillArc(10, 10, 100, 100, 0, 150);
    // Start at 12 o'clock and rotate 150 degrees
//    g.drawArc(10, 10, 100, 100, 90, 150);
    // Change the size of the bounding box
    // Start at 12 o'clock and rotate 150 degrees
//    g.drawArc(15, 45, 30, 70, 90, 150);
```

```
  }
  public void commandAction(Command c, Displayable d){
    if (c == cmExit)
      midlet.exitMIDlet();
  }
}
```

Bài 31: **VeHinhChuNhat**, vẽ hình chữ nhật ra màn hình

```
//  RectangleExample.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class RectangleExample extends MIDlet{
  private Display  display;
  private MyCanvas canvas;
  public RectangleExample (){
    display = Display.getDisplay(this);
    canvas  = new MyCanvas(this);
  }
  protected void startApp(){
    display.setCurrent( canvas );
  }
  protected void pauseApp(){}
  protected void destroyApp( boolean unconditional ){}
  public void exitMIDlet(){
    destroyApp(true);
    notifyDestroyed();
  }
}
class MyCanvas extends Canvas implements CommandListener{
  private Command exit;
  private RectangleExample filledRectangleExample;
  public MyCanvas (RectangleExample filledRectangleExample){
    this. filledRectangleExample = filledRectangleExample;
    exit = new Command("Exit", Command.EXIT, 1);
    addCommand(exit);
    setCommandListener(this);
  }
  protected void paint(Graphics graphics){
    graphics.setColor(255,255,255);
    graphics.fillRect(0, 0, getWidth(), getHeight());
    graphics.setColor(0, 0, 255);
    graphics.fillRect(2, 2, 20, 20);
    graphics.fillRoundRect(20, 20, 60, 60, 15, 45);
  }
  public void commandAction(Command command, Displayable s){
    if (command == exit){
     filledRectangleExample.exitMIDlet();
    }
  }
}
```

Bài 32: FontChuDonGian, hiển thị các loại Font

```
// FontCanvasExample.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class FontCanvasExample extends MIDlet {
  public FontCanvasExample() { // constructor
  }
  public void startApp() {
    Canvas canvas = new FontCanvas();
    Display display = Display.getDisplay(this);
    display.setCurrent(canvas);
  }
  public void pauseApp() {}
  public void destroyApp(boolean unconditional) {}
}
class FontCanvas extends Canvas {
  public void paint(Graphics g) {
    g.setColor(0xffffff);
    g.fillRect(0, 0, getWidth(), getHeight());
    g.setColor(0x000000);
    g.setFont(Font.getFont(Font.FACE_SYSTEM, Font.STYLE_PLAIN, Font.SIZE_LARGE));
    g.drawString("System Font", 0, 0, g.LEFT | g.TOP);
    g.setFont(Font.getFont(Font.FACE_SYSTEM, Font.STYLE_PLAIN, Font.SIZE_MEDIUM));
    g.drawString("Medium Size", 0, 15, g.LEFT | g.TOP);
    g.setFont(Font.getFont(Font.FACE_SYSTEM, Font.STYLE_BOLD, Font.SIZE_MEDIUM));
    g.drawString("Bold Style", 0, 30, g.LEFT | g.TOP);
    g.setFont(Font.getFont(Font.FACE_SYSTEM, Font.STYLE_ITALIC, Font.SIZE_MEDIUM));
    g.drawString("Italic Style", 0, 45, g.LEFT | g.TOP);
g.setFont(Font.getFont(Font.FACE_SYSTEM, Font.STYLE_UNDERLINED,Font.SIZE_MEDIUM));
    g.drawString("Underlined Style", 0, 60, g.LEFT | g.TOP);
  }
}
```

Bài 33: FontChu1, hiển thị các loại Font. Menu thao tác chọn Font rồi đặt lại

```
// FontViewer.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class FontViewer extends MIDlet{
 protected Display  display;     // The display
 protected PrefsForm fmPrefs;    // Form to choose font prefs
 protected FontCanvas cvFont;    // Canvas to display text (in preferred font)
 public FontViewer(){
   display = Display.getDisplay(this);
   cvFont = new FontCanvas(this);
   fmPrefs = new PrefsForm("Preferences", this);
 }
 protected void startApp(){
   showCanvas();
 }
 protected void showCanvas(){
   display.setCurrent(cvFont);
 }
```

```
  protected void pauseApp(){ }
  protected void destroyApp( boolean unconditional ){ }
  public void exitMIDlet(){
    destroyApp(true);
    notifyDestroyed();
  }
}
// FontCanvas.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
class FontCanvas extends Canvas implements CommandListener{
  private int face,          // Font face
              style,         //      style
              size;          //      size
  private String text = "developerWorks";   // Text to display in preferred font
  private Command cmExit;     // Exit midlet
  private Command cmPrefs;    // Call the preferences form
  private FontViewer midlet;  // Reference to the main midlet
  public FontCanvas(FontViewer midlet){
    this.midlet = midlet;
    // Create commands  and listen for events
    cmExit = new Command("Exit", Command.EXIT, 1);
    cmPrefs = new Command("Prefs", Command.SCREEN, 2);
    addCommand(cmExit);
    addCommand(cmPrefs);
    setCommandListener(this);
  }
  /*-----------------------------------------------
  * Draw text
  *-----------------------------------------------*/
  protected void paint(Graphics g){
    // Clear the display
    g.setColor(255, 255, 255);      // White pen
    g.fillRect(0, 0, getWidth(), getHeight());
    g.setColor(0, 0, 0);            // Black pen
    // Use the user selected font preferences
            g.setFont(Font.getFont(face, style, size));
    // Draw text at center of display
    g.drawString(text, getWidth() / 2, getHeight() / 2,
    Graphics.BASELINE | Graphics.HCENTER);
  }
  protected void setFace(int face){
    this.face = face;
  }
  protected void setStyle(int style){
    this.style = style;
  }
  protected void setSize(int size){
    this.size = size;
  }
  public void setText(String text){
    this.text = text;
  }
  public int getFace(){
```

```
    return face;
  }
  public int getStyle(){
    return style;
  }
  public int getSize(){
    return size;
  }
  public void commandAction(Command c, Displayable d){
    if (c == cmExit)
      midlet.exitMIDlet();
    else if (c == cmPrefs)
      midlet.display.setCurrent(midlet.fmPrefs);
  }
}
// PrefsForm.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class PrefsForm extends Form implements CommandListener, ItemStateListener{
  private FontViewer midlet;      // Main midlet
  private Command cmBack,         // Back to canvas
                  cmSave;         // Save new font prefs
  protected ChoiceGroup cgFace,   // Font face
                  cgStyle,        // style
                  cgSize;         // size
  protected TextField tfText;     // Text string to display on canvas
                                  // (with the requested font prefs)
            private int face = 0,           // Values for font attributes
            style = 0,          // that are obtained from the choicegroups
                        size = 0;
  private String text = null;     // Value for text string from the textfield
  public PrefsForm(String title, FontViewer midlet){
    // Call the form constructor
    super(title);
    // Save reference to MIDlet so we can
    // access the display manager class
    this.midlet = midlet;
    // Commands
    cmSave = new Command("Save", Command.SCREEN, 1);
    cmBack = new Command("Back", Command.BACK, 2);
      // Exclusive choice group for font face
    cgFace = new ChoiceGroup("Face Options", Choice.EXCLUSIVE);
    cgFace.append("System", null);
    cgFace.append("Monospace", null);
    cgFace.append("Proportional", null);
    // Multiple choice group for font style
    cgStyle = new ChoiceGroup("Style Options", Choice.MULTIPLE);
    cgStyle.append("Plain", null);
    cgStyle.append("Bold", null);
    cgStyle.append("Italic", null);
    cgStyle.append("Underlined", null);
    // Exclusive choice group for font size
    cgSize = new ChoiceGroup("Size Options", Choice.EXCLUSIVE);
    cgSize.append("Small", null);
```

```
  cgSize.append("Medium", null);
  cgSize.append("Large", null);
  // Textfield for text to display on canvas
  tfText = new TextField("", "developerWorks", 20, TextField.ANY);
  // Create form contents and listen for events
  append(cgFace);
  append(cgStyle);
  append(cgSize);
  append(tfText);
  addCommand(cmSave);
  addCommand(cmBack);
  setCommandListener(this);
  setItemStateListener(this);
}
/*-------------------------------------------------
* Command event handling
*-------------------------------------------------*/
public void commandAction(Command c, Displayable s)  {
  if (c == cmSave){
    // Set the values in canvas class to those selected here
    midlet.cvFont.setFace(face);
    midlet.cvFont.setStyle(style);
    midlet.cvFont.setSize(size);
    // Make sure we aren't passing a null value
    midlet.cvFont.setText(tfText.getString() !=
    null ? tfText.getString() : "developerWorks");
  }
  // No specific check needed for "Back" command
  // Always return to the canvas at this point
  midlet.showCanvas();
}
public void itemStateChanged(Item item)  {
  if (item == cgFace){
            switch (cgFace.getSelectedIndex()){
      case 0:
        face = Font.FACE_SYSTEM;
        break;
      case 1:
        face = Font.FACE_MONOSPACE;
        break;
      case 2:
        face = Font.FACE_PROPORTIONAL;
        break;
    }
  }
  else if (item == cgStyle){
    boolean[] b = new boolean[4];
    cgStyle.getSelectedFlags(b);
    style = 0;
    // If bold selected
            if (b[1])
      style = Font.STYLE_BOLD;
    // If italic selected
            if (b[2])
```

```
        style |= Font.STYLE_ITALIC;
      // If underlined selected
            if (b[3])
        style |= Font.STYLE_UNDERLINED;
    }
    else if (item == cgSize){
            switch (cgSize.getSelectedIndex()){
        case 0:
          size = Font.SIZE_SMALL;
          break;
        case 1:
          size = Font.SIZE_MEDIUM;
          break;
        case 2:
          size = Font.SIZE_LARGE;
          break;
      }
    }
    else if (item == tfText){
      text = tfText.getString();
    }
  }
}
```

Bài 34: **Ve2**, chèn 1 ảnh vào và dùng phím dịch chuyển ảnh trong màn hình hiển thị

```
// Translate.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class Translate extends MIDlet{
  private Display  display;     // The display
  private TranslateCanvas canvas;   // Canvas
   public Translate(){
    display = Display.getDisplay(this);
    canvas  = new TranslateCanvas(this);
  }
   protected void startApp(){
    display.setCurrent( canvas );
  }
   protected void pauseApp() { }
  protected void destroyApp( boolean unconditional )  { }
   public void exitMIDlet() {
    destroyApp(true);
    notifyDestroyed();
  }
}
/*-------------------------------------------------
* Class Translate
*
* Draw image using translated coordinates
*-------------------------------------------------*/
class TranslateCanvas extends Canvas implements CommandListener{
  private Command cmExit;  // Exit midlet
  private Translate midlet;
```

```
private Image im = null;
private int translatex = 0, translatey = 0;
 public TranslateCanvas(Translate midlet) {
  this.midlet = midlet;
     // Create exit command & listen for events
  cmExit = new Command("Exit", Command.EXIT, 1);
  addCommand(cmExit);
  setCommandListener(this);
  try    {
   // Create immutable image
   im = Image.createImage("/bolt.png");
  }
  catch (java.io.IOException e){
   System.err.println("Unable to locate or read .png file");
  }
}
protected void paint(Graphics g)  {
  if (im != null){
   // Clear the background
   g.setColor(255, 255, 255);
   g.fillRect(0, 0, getWidth(), getHeight());
   // Translate coordinates
   g.translate(translatex, translatey);
   // Always draw at 0,0
   g.drawImage(im, 0, 0, Graphics.LEFT | Graphics.TOP);
  }
}
public void commandAction(Command c, Displayable d)  {
  if (c == cmExit)
   midlet.exitMIDlet();
}
  protected void keyPressed(int keyCode){
  switch (getGameAction(keyCode)){
    case UP:
          // If scrolling off the top, roll around to bottom
     if (translatey - im.getHeight() < 0)
      translatey = getHeight() - im.getHeight();
     else
      translatey -= im.getHeight();
     break;
        case DOWN:
     // If scrolling off the bottom, roll around to top
     if ((translatey + im.getHeight() + im.getHeight()) > getHeight())
      translatey = 0;
     else
      translatey += im.getHeight();
     break;
    case LEFT:
     // If scrolling off the left, bring around to right
     if (translatex - im.getWidth() < 0)
      translatex = getWidth() - im.getWidth();
     else
      translatex -= im.getWidth();
     break;
```

```
      case RIGHT:
        // If scrolling off the right, bring around to left
        if ((translatex + im.getWidth() + translatex) > getWidth())
          translatex = 0;
        else
          translatex += im.getWidth();
        break;
      }
      repaint();
    }
  }
```

Bài 35: **ExampleGameCanvas**, dùng GameCanvas dịch chuyển ký tự 'x' trong màn hình hiển thị

```
// ExampleGameCanvasMidlet.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class ExampleGameCanvasMidlet extends MIDlet {
  private Display display;
  public void startApp() {
    display = Display.getDisplay(this);
    ExampleGameCanvas gameCanvas = new ExampleGameCanvas();
    gameCanvas.start();
    display.setCurrent(gameCanvas);
  }
  public Display getDisplay() {
    return display;
  }
  public void pauseApp() {}
  public void destroyApp(boolean unconditional) {
    exit();
  }
  public void exit() {
    System.gc();
    destroyApp(false);
    notifyDestroyed();
  }
}
// ExampleGameCanvas
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;
public class ExampleGameCanvas extends GameCanvas implements Runnable {
  private boolean isPlay;   // Game Loop runs when isPlay is true
  private long delay;       // To give thread consistency
  private int currentX, currentY;  // To hold current position of the 'X'
  private int width;        // To hold screen width
  private int height;       // To hold screen height
  // Constructor and initialization
  public ExampleGameCanvas() {
    super(true);
    width = getWidth();
    height = getHeight();
    currentX = width / 2;
```

```
    currentY = height / 2;
    delay = 20;
  }
  // Automatically start thread for game loop
  public void start() {
    isPlay = true;
    Thread t = new Thread(this);
    t.start();
  }
  public void stop() { isPlay = false; }
  // Main Game Loop
  public void run() {
    Graphics g = getGraphics();
    while (isPlay == true) {
      input();
      drawScreen(g);
      try { Thread.sleep(delay); }
      catch (InterruptedException ie) {}
    }
  }
  // Method to Handle User Inputs
  private void input() {
    int keyStates = getKeyStates();
    // Left
    if ((keyStates & LEFT_PRESSED) != 0)
      currentX = Math.max(0, currentX - 1);
    // Right
    if ((keyStates & RIGHT_PRESSED) !=0 )
      if ( currentX + 5 < width)
        currentX = Math.min(width, currentX + 1);
    // Up
    if ((keyStates & UP_PRESSED) != 0)
      currentY = Math.max(0, currentY - 1);
    // Down
    if ((keyStates & DOWN_PRESSED) !=0)
      if ( currentY + 10 < height)
        currentY = Math.min(height, currentY + 1);
  }
  // Method to Display Graphics
  private void drawScreen(Graphics g) {
    g.setColor(0xffffff);
    g.fillRect(0, 0, getWidth(), getHeight());
    g.setColor(0x0000ff);
    g.drawString("X",currentX,currentY,Graphics.TOP|Graphics.LEFT);
    flushGraphics();
  }
}
```

Bài 36: **GameCanvas**, hiển thị bầu trời sao với phím UP và DOWN để điều chỉnh tốc độ chảy của bầu trời

```
//  GameCanvasTest.java
import java.io.*;
import java.util.*;
```

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;
import javax.microedition.midlet.*;
public class GameCanvasTest extends MIDlet implements CommandListener {
   private Display display;
   public static final Command exitCommand = new Command( "Exit",Command.EXIT, 1 );
   public GameCanvasTest(){}
   public void commandAction( Command c, Displayable d ){
      if( c == exitCommand ){
         exitMIDlet();
      }
   }
   protected void destroyApp( boolean unconditional ) throws MIDletStateChangeException {
      exitMIDlet();
   }
   public void exitMIDlet(){
      notifyDestroyed();
   }
   public Display getDisplay(){ return display; }
   protected void initMIDlet(){
         GameCanvas c = new StarField();
      c.addCommand( exitCommand );
      c.setCommandListener( this );
      getDisplay().setCurrent( c );
   }
   protected void pauseApp(){}
   protected void startApp() throws MIDletStateChangeException {
      if( display == null ){
         display = Display.getDisplay( this );
         initMIDlet();
      }
   }
}
// StarField.java
import java.util.Random;
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.GameCanvas;
// A simple example of a game canvas that displays
// a scrolling star field. Use the UP and DOWN keys
// to speed up or slow down the rate of scrolling.
public class StarField extends GameCanvas implements Runnable {
   private static final int SLEEP_INCREMENT = 10;
   private static final int SLEEP_INITIAL = 150;
   private static final int SLEEP_MAX = 300;
   private Graphics       graphics;
   private Random         random;
   private int            sleepTime = SLEEP_INITIAL;
   private volatile Thread thread;
   public StarField(){
      super( true );
      graphics = getGraphics();
      graphics.setColor( 0,  0, 0 );
      graphics.fillRect( 0, 0, getWidth(), getHeight() );
   }
```

```
    // When the game canvas is hidden, stop the thread.
    protected void hideNotify(){
        thread = null;
    }
    // The game loop.
    public void run(){
        int w = getWidth();
        int h = getHeight() - 1;
        while( thread == Thread.currentThread() ){
            // Increment or decrement the scrolling interval
            // based on key presses
            int state = getKeyStates();
            if( ( state & DOWN_PRESSED ) != 0 ){
                sleepTime += SLEEP_INCREMENT;
                if( sleepTime > SLEEP_MAX ) sleepTime = SLEEP_MAX;
            } else if( ( state & UP_PRESSED ) != 0 ){
                sleepTime -= SLEEP_INCREMENT;
                if( sleepTime < 0 ) sleepTime = 0;
            }
            // Repaint the screen by first scrolling the
            // existing starfield down one and painting in
            // new stars...
            graphics.copyArea( 0, 0, w, h, 0, 1,
                               Graphics.TOP | Graphics.LEFT );
            graphics.setColor( 0, 0, 0 );
            graphics.drawLine( 0, 0, w, 1 );
            graphics.setColor( 255, 255, 255 );
            for( int i = 0; i < w; ++i ){
                int test = Math.abs( random.nextInt() ) % 100;
                if( test < 4 ){
                    graphics.drawLine( i, 0, i, 0 );
                }
            }
            flushGraphics();
            // Now wait...
            try {
                Thread.currentThread().sleep( sleepTime );
            }
            catch( InterruptedException e ){
            }
        }
    }
    // When the canvas is shown, start a thread to
    // run the game loop.
    protected void showNotify(){
        random = new Random();
        thread = new Thread( this );
        thread.start();
    }
}
```

Bài 37: **ExampleGameSprite**, dùng Sprite quản lý các Frame ảnh (5 frames)

```
// ExampleGameSpriteMidlet.java
```

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class ExampleGameSpriteMidlet extends MIDlet {
  private Display display;
  public void startApp() {
    try {
      display = Display.getDisplay(this);
      ExampleGameCanvas gameCanvas = new ExampleGameCanvas();
      gameCanvas.start();
      display.setCurrent(gameCanvas);
    } catch (Exception ex) {
      System.out.println(ex);
    }
  }
  public Display getDisplay() {
    return display;
  }
  public void pauseApp() {
  }
  public void destroyApp(boolean unconditional) {
    exit();
  }
  public void exit() {
    System.gc();
    destroyApp(false);
    notifyDestroyed();
  }
}
// ExampleGameCanvas.java
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;
public class ExampleGameCanvas extends GameCanvas implements Runnable {
  private boolean isPlay;   // Game Loop runs when isPlay is true
  private long delay;       // To give thread consistency
  private int currentX, currentY;  // To hold current position of the 'X'
  private int width;        // To hold screen width
  private int height;       // To hold screen height
  // Sprites to be used
  private Sprite sprite;
  private Sprite nonTransparentSprite;
  // Constructor and initialization
  public ExampleGameCanvas() throws Exception {
    super(true);
    width = getWidth();
    height = getHeight();
    currentX = width / 2;
    currentY = height / 2;
    delay = 20;
    // Load Images to Sprites
    Image image = Image.createImage("/transparent.png");
    sprite = new Sprite (image,32,32);
    Image imageTemp = Image.createImage("/nontransparent.png");
    nonTransparentSprite = new Sprite (imageTemp,32,32);
  }
```

```
// Automatically start thread for game loop
public void start() {
  isPlay = true;
  Thread t = new Thread(this);
  t.start();
}
public void stop() { isPlay = false; }
// Main Game Loop
public void run() {
  Graphics g = getGraphics();
  while (isPlay == true) {
    input();
    drawScreen(g);
    try { Thread.sleep(delay); }
    catch (InterruptedException ie) {}
  }
}
// Method to Handle User Inputs
private void input() {
  int keyStates = getKeyStates();
  sprite.setFrame(0);
  // Left
  if ((keyStates & LEFT_PRESSED) != 0) {
    currentX = Math.max(0, currentX - 1);
    sprite.setFrame(1);
  }
  // Right
  if ((keyStates & RIGHT_PRESSED) !=0 )
    if ( currentX + 5 < width) {
      currentX = Math.min(width, currentX + 1);
      sprite.setFrame(3);
    }
  // Up
  if ((keyStates & UP_PRESSED) != 0) {
    currentY = Math.max(0, currentY - 1);
    sprite.setFrame(2);
  }
  // Down
  if ((keyStates & DOWN_PRESSED) !=0)
    if ( currentY + 10 < height) {
      currentY = Math.min(height, currentY + 1);
      sprite.setFrame(4);
    }
}
// Method to Display Graphics
private void drawScreen(Graphics g) {
  //g.setColor(0xffffff);
  g.setColor(0xFF0000);
  g.fillRect(0, 0, getWidth(), getHeight());
  g.setColor(0x0000ff);
  // display sprites
  sprite.setPosition(currentX,currentY);
  sprite.paint(g);
  nonTransparentSprite.paint(g);
```

```
    flushGraphics();
  }
}
```

Bài 38: **ExampleLayerManager**, có 2 LayerManager (nền và ảnh)

```
//  ExampleLayerManagerMidlet.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class ExampleLayerManagerMidlet extends MIDlet {
  private Display display;
  public void startApp() {
    try {
      display = Display.getDisplay(this);
      ExampleGameCanvas gameCanvas = new ExampleGameCanvas();
      gameCanvas.start();
      display.setCurrent(gameCanvas);
    } catch (Exception ex) {
      System.out.println(ex);
    }
  }
  public Display getDisplay() {
    return display;
  }
  public void pauseApp() {
  }
  public void destroyApp(boolean unconditional) {
    exit();
  }
  public void exit() {
    System.gc();
    destroyApp(false);
    notifyDestroyed();
  }
}
// ExampleGameCanvas.java
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;
public class ExampleGameCanvas extends GameCanvas implements Runnable {
  private boolean isPlay;   // Game Loop runs when isPlay is true
  private long delay;       // To give thread consistency
  private int currentX, currentY;  // To hold current position of the 'X'
  private int width;        // To hold screen width
  private int height;       // To hold screen height
  // Sprites to be used
  private Sprite playerSprite;
  private Sprite backgroundSprite;
  // Layer Manager
  private LayerManager layerManager;
  // Constructor and initialization
  public ExampleGameCanvas() throws Exception {
    super(true);
    width = getWidth();
    height = getHeight();
```

```
  currentX = width / 2;
  currentY = height / 2;
  delay = 20;
  // Load Images to Sprites
  Image playerImage = Image.createImage("/transparent.png");
  playerSprite = new Sprite (playerImage,32,32);
  Image backgroundImage = Image.createImage("/background.png");
  backgroundSprite = new Sprite(backgroundImage);
  layerManager = new LayerManager();
  layerManager.append(playerSprite);
  layerManager.append(backgroundSprite);
}
// Automatically start thread for game loop
public void start() {
  isPlay = true;
  Thread t = new Thread(this);
  t.start();
}
public void stop() { isPlay = false; }
// Main Game Loop
public void run() {
  Graphics g = getGraphics();
  while (isPlay == true) {
    input();
    drawScreen(g);
    try { Thread.sleep(delay); }
    catch (InterruptedException ie) {}
  }
}
// Method to Handle User Inputs
private void input() {
  int keyStates = getKeyStates();
  playerSprite.setFrame(0);
  // Left
  if ((keyStates & LEFT_PRESSED) != 0) {
    currentX = Math.max(0, currentX - 1);
    playerSprite.setFrame(1);
  }
  // Right
  if ((keyStates & RIGHT_PRESSED) !=0 )
    if ( currentX + 5 < width) {
      currentX = Math.min(width, currentX + 1);
      playerSprite.setFrame(3);
    }
  // Up
  if ((keyStates & UP_PRESSED) != 0) {
    currentY = Math.max(0, currentY - 1);
    playerSprite.setFrame(2);
  }
  // Down
  if ((keyStates & DOWN_PRESSED) !=0)
    if ( currentY + 10 < height) {
      currentY = Math.min(height, currentY + 1);
      playerSprite.setFrame(4);
```

```
    }
  }
  // Method to Display Graphics
  private void drawScreen(Graphics g) {
    //g.setColor(0x00C000);
    g.setColor(0xffffff);
    g.fillRect(0, 0, getWidth(), getHeight());
    g.setColor(0x0000ff);
    // updating player sprite position
    playerSprite.setPosition(currentX,currentY);
    // display all layers
    //layerManager.paint(g,0,0);
    layerManager.setViewWindow(55,20,140,140);
    layerManager.paint(g,20,20);
    flushGraphics();
  }
}
```

Bài 39: **CuonManHinhNen**, dùng phím dịch chuyển cuộn màn hình

```
//  ScrollingLayerManager.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class ScrollingLayerManager extends MIDlet {
private Display display;
public void startApp() {
try {
display = Display.getDisplay(this);
ExampleGameCanvas gameCanvas = new ExampleGameCanvas();
gameCanvas.start();
display.setCurrent(gameCanvas);
} catch (Exception ex) {
System.out.println(ex);
}
}
public Display getDisplay() {
return display;
}
public void pauseApp() {
}
public void destroyApp(boolean unconditional) {
exit();
}
public void exit() {
System.gc();
destroyApp(false);
notifyDestroyed();
}
}
// ExampleGameCanvas.java
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;
public class ExampleGameCanvas extends GameCanvas implements Runnable {
private boolean isPlay; // Game Loop runs when isPlay is true
```

```
private long delay; // To give thread consistency
private int width; // To hold screen width
private int height; // To hold screen height
private int scnX, scnY; // To hold screen starting viewpoint
// Sprites to be used
Image backgroundImage;
private Sprite backgroundSprite;
// Layer Manager
private LayerManager layerManager;
// Constructor and initialization
public ExampleGameCanvas() throws Exception {
super(true);
width = getWidth();
height = getHeight();
scnX = 55;
scnY = 20;
delay = 20;
// Load Images to Sprites
backgroundImage = Image.createImage("/background.png");
backgroundSprite = new Sprite(backgroundImage);
layerManager = new LayerManager();
layerManager.append(backgroundSprite);
}
// Automatically start thread for game loop
public void start() {
isPlay = true;
Thread t = new Thread(this);
t.start();
}
public void stop() { isPlay = false; }
// Main Game Loop
public void run() {
Graphics g = getGraphics();
while (isPlay == true) {
input();
drawScreen(g);
try { Thread.sleep(delay); }
catch (InterruptedException ie) {}
}
}
// Method to Handle User Inputs
private void input() {
int keyStates = getKeyStates();
if ((keyStates & LEFT_PRESSED) != 0) {
if (scnX - 1 > 0)
  scnX--;
}
if ((keyStates & RIGHT_PRESSED) != 0) {
if (scnX + 1 + 140 < backgroundImage.getWidth())
  scnX++;
}
if ((keyStates & UP_PRESSED)!=0){
if (scnY-1>0)
  scnY--;
```

```
}
if ((keyStates & DOWN_PRESSED)!=0){
          if (scnY+1+140<backgroundImage.getHeight())
            scnY++;
}
}
// Method to Display Graphics
private void drawScreen(Graphics g) {
//g.setColor(0x00C000);
g.setColor(0xffffff);
g.fillRect(0, 0, getWidth(), getHeight());
g.setColor(0x0000ff);
// display all layers
layerManager.setViewWindow(scnX,scnY,140,140);
layerManager.paint(g,20,20);
flushGraphics();
}
}
```

Bài 40:**ExampleTiledLayer**, chia ảnh ra thành 8 x 9 tiles

```
// ExampleTileLayer.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class ExampleTiledLayer extends MIDlet {
  private Display display;

  public void startApp() {
    try {
      display = Display.getDisplay(this);
      ExampleGameCanvas gameCanvas = new ExampleGameCanvas();
      gameCanvas.start();
      display.setCurrent(gameCanvas);
    } catch (Exception ex) {
      System.out.println(ex);
    }
  }
  public Display getDisplay() {
    return display;
  }
  public void pauseApp() {
  }
  public void destroyApp(boolean unconditional) {
    exit();
  }
  public void exit() {
    System.gc();
    destroyApp(false);
    notifyDestroyed();
  }
}
// ExampleGameCanvas.java
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;
```

```
public class ExampleGameCanvas extends GameCanvas implements Runnable {
  private boolean isPlay;   // Game Loop runs when isPlay is true
  private long delay;       // To give thread consistency
  private int width;        // To hold screen width
  private int height;       // To hold screen height
  // Layer Manager
  private LayerManager layerManager;
  // TiledLayer
  private TiledLayer tiledBackground;
  // Constructor and initialization
  public ExampleGameCanvas() throws Exception {
    super(true);
    width = getWidth();
    height = getHeight();
    delay = 20;
    tiledBackground = initBackground();
    layerManager = new LayerManager();
    layerManager.append(tiledBackground);
  }
  // Automatically start thread for game loop
  public void start() {
    isPlay = true;
    Thread t = new Thread(this);
    t.start();
  }
  public void stop() { isPlay = false; }
  // Main Game Loop
  public void run() {
    Graphics g = getGraphics();
    while (isPlay == true) {
      input();
      drawScreen(g);
      try {
        Thread.sleep(delay);
      } catch (InterruptedException ie) {
      }
    }
  }
  // Method to Handle User Inputs
  private void input() {
     // no inputs
  }
  // Method to Display Graphics
  private void drawScreen(Graphics g) {
    g.setColor(0xffffff);
    g.fillRect(0, 0, getWidth(), getHeight());
    g.setColor(0x0000ff);
    layerManager.paint(g,0,0);
    flushGraphics();
  }
  private TiledLayer initBackground() throws Exception {
    Image tileImages = Image.createImage("/tiles.png");
    TiledLayer tiledLayer = new TiledLayer(8,9,tileImages,32,32);
    int[] map = {
```

```
      5, 1, 1, 4, 1, 1, 1, 1,
      5, 1, 3, 1, 1, 3, 1, 1,
      5, 1, 2, 1, 1, 2, 1, 1,
      5, 1, 2, 3, 1, 2, 1, 1,
      5, 1, 4, 2, 1, 2, 1, 1,
      5, 1, 1, 4, 1, 2, 1, 1,
      5, 1, 1, 1, 1, 4, 1, 1,
      5, 1, 1, 1, 1, 1, 1, 1,
      5, 1, 1, 1, 1, 1, 1, 1
    };
    for (int i=0; i < map.length; i++) {
      int column = i % 8;
      int row = (i - column) / 8;
      tiledLayer.setCell(column,row,map[i]);
    }
    return tiledLayer;
  }
}
```

Bài 41: **ExampleTiledLayerAnimated**, lấy mẫu có Index 3 và 4 làm Animated đặt vào vị trí 1 x 1

```
//  ExampleTiledLayerAnimated.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class ExampleTiledLayerAnimated extends MIDlet {
  private Display display;
  public void startApp() {
    try {
      display = Display.getDisplay(this);
      ExampleGameCanvas gameCanvas = new ExampleGameCanvas();
      gameCanvas.start();
      display.setCurrent(gameCanvas);
    } catch (Exception ex) {
      System.out.println(ex);
    }
  }
  public Display getDisplay() {
    return display;
  }
  public void pauseApp() {
  }
  public void destroyApp(boolean unconditional) {
    exit();
  }
  public void exit() {
    System.gc();
    destroyApp(false);
    notifyDestroyed();
  }
}
// ExampleGameCanvas.java
import javax.microedition.lcdui.*;
```

```
import javax.microedition.lcdui.game.*;
public class ExampleGameCanvas extends GameCanvas implements Runnable {
  private boolean isPlay;   // Game Loop runs when isPlay is true
  private long delay;       // To give thread consistency
  private int currentX, currentY;  // To hold current position of the 'X'
  private int width;        // To hold screen width
  private int height;       // To hold screen height
  // Layer Manager
  private LayerManager layerManager;
  // TiledLayer
  private TiledLayer tiledBackground;
  // Flag to indicate tile switch
  private boolean switchTile;
  // To hold the animated tile index
  private int animatedIdx;
  // Constructor and initialization
  public ExampleGameCanvas() throws Exception {
    super(true);
    width = getWidth();
    height = getHeight();
    currentX = width / 2;
    currentY = height / 2;
    delay = 20;
    tiledBackground = initBackground();
    layerManager = new LayerManager();
    layerManager.append(tiledBackground);
  }
  // Automatically start thread for game loop
  public void start() {
    isPlay = true;
    Thread t = new Thread(this);
    t.start();
  }
  public void stop() { isPlay = false; }
  // Main Game Loop
  public void run() {
    Graphics g = getGraphics();
    while (isPlay == true) {
      input();
      drawScreen(g);
      try {
        Thread.sleep(delay);
      } catch (InterruptedException ie) {
      }
    }
  }
  // Method to Handle User Inputs
  private void input() {
     // no inputs
  }
  // Method to Display Graphics
  private void drawScreen(Graphics g) {
    g.setColor(0xffffff);
    g.fillRect(0, 0, getWidth(), getHeight());
```

```
    g.setColor(0x0000ff);
    // Determine which tile to show
    if (switchTile) {
      tiledBackground.setAnimatedTile(animatedIdx,3);
    } else {
      tiledBackground.setAnimatedTile(animatedIdx,4);
    }
    // Set tile file to opposite boolean value
    switchTile = !switchTile;
    layerManager.paint(g,0,0);
    flushGraphics();
  }
  private TiledLayer initBackground() throws Exception {
    Image tileImages = Image.createImage("/tiles.png");
    TiledLayer tiledLayer = new TiledLayer(10,10,tileImages,32,32);
    int[] map = {
       5,  1,  1,  4,  1,  1,  1,  1,  1,  6,
       5,  1,  3,  1,  1,  3,  1,  1,  1,  6,
       5,  1,  2,  1,  1,  2,  1,  1,  1,  6,
       5,  1,  2,  3,  1,  2,  1,  1,  1,  6,
       5,  1,  4,  2,  1,  2,  1,  1,  1,  6,
       5,  1,  1,  4,  1,  2,  1,  1,  1,  6,
       5,  1,  1,  1,  1,  4,  1,  1,  1,  6,
       5,  1,  1,  1,  1,  1,  1,  1,  1,  6,
       5,  1,  1,  1,  1,  1,  1,  1,  1,  6,
       5,  1,  1,  1,  1,  1,  1,  1,  1,  6
    };
    for (int i=0; i < map.length; i++) {
      int column = i % 10;
      int row = (i - column) / 10;
      tiledLayer.setCell(column,row,map[i]);
    }
    // Created animate tile and hold animated tile index
    animatedIdx = tiledLayer.createAnimatedTile(5);
    // Set Cell with animated tile index
    tiledLayer.setCell(1,1,animatedIdx);
    return tiledLayer;
  }
}
```

Bài 42: **Animation**, lấy mẫu có Index 3 và 4 làm Animated đặt vào vị

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class Animation extends MIDlet implements CommandListener
{
private Display display; // Reference to display
private AnimationCanvas canvas; // Game canvas
private Command cmExit; // Exit command
public Animation()
{
display = Display.getDisplay(this);
cmExit = new Command("Exit", Command.EXIT, 1);
// Create game canvas and exit command
if ((canvas = new AnimationCanvas()) != null)
```

```
{
canvas.addCommand(cmExit);
canvas.setCommandListener(this);
}
}
public void startApp()
{
if (canvas != null)
{
display.setCurrent(canvas);
canvas.start();
}
}
public void pauseApp()
{}
public void destroyApp(boolean unconditional)
{
canvas.stop();
}

public void commandAction(Command c, Displayable s)
{
if (c == cmExit)
{
destroyApp(true);
notifyDestroyed();
}
}
}

import javax.microedition.lcdui.game.*;
import javax.microedition.lcdui.*;
public class AnimationCanvas extends GameCanvas implements Runnable
{
// Size of one frame in the spiral image
private static final int FRAME_WIDTH = 32;
private static final int FRAME_HEIGHT = 32;
private AnimationSprite spSpiral; // Animated sprite
private LayerManager lmgr; // Manage layers
private boolean running = false; // Thread running?
public AnimationCanvas()
{
// Gamecanvas constructor
super(true);
try
{
// Animated sprite
spSpiral = new AnimationSprite(Image.createImage("/tank.png"),
FRAME_WIDTH, FRAME_HEIGHT);
// Change the reference pixel to the middle of sprite
spSpiral.defineReferencePixel(FRAME_WIDTH / 2, FRAME_HEIGHT / 2);
// Center the sprite on the canvas
// (center of sprite is now in center of display)
spSpiral.setRefPixelPosition(80,100);
```

```
// Layer manager
lmgr = new LayerManager();
lmgr.append(spSpiral);
}
catch (Exception e)
{
System.out.println("Unable to read PNG image");
}
}
/*-------------------------------------------------
* Start thread
*-------------------------------------------------*/
public void start()
{
running = true;
Thread t = new Thread(this);
t.start();
}
/*-------------------------------------------------
* Main loop
*-------------------------------------------------*/
public void run()
{
Graphics g = getGraphics();
while (running)
{
// Update display
drawDisplay(g);
try
{
Thread.sleep(150);
}
catch (InterruptedException ie)
{
System.out.println("Thread exception");
}
}
}

private void drawDisplay(Graphics g)
{
// Animated sprite, show next frame in sequence
spSpiral.nextFrame();
// Paint layers
lmgr.paint(g, 0, 0);
// Flush off-screen buffer to display
flushGraphics();
}
/*-------------------------------------------------
* Stop thread
*-------------------------------------------------*/
public void stop()
{
running = false;
```

```
}
}

import javax.microedition.lcdui.game.*;
import javax.microedition.lcdui.*;
public class AnimationSprite extends Sprite
{
public AnimationSprite(Image image, int frameWidth, int frameHeight)
{
// Call sprite constructor
super(image, frameWidth, frameHeight);
}
}
```

Bài 43: **Collisions**, lấy mẫu có Index 3 và 4 làm Animated đặt vào vị

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.game.*;
import javax.microedition.lcdui.*;
public class Collisions extends MIDlet implements CommandListener
{
protected Display display; // Reference to display
private CollisionCanvas canvas; // Game canvas
private Command cmExit; // Exit command
public Collisions()
{
display = Display.getDisplay(this);
// Create game canvas and exit command
if ((canvas = new CollisionCanvas(this)) != null)
{
cmExit = new Command("Exit", Command.EXIT, 1);
canvas.addCommand(cmExit);
canvas.setCommandListener(this);
}
}
public void startApp()
{
if (canvas != null)
{
display.setCurrent(canvas);
canvas.start();
}
}
public void pauseApp()
{}
public void destroyApp(boolean unconditional)
{
canvas.stop();
}
public void commandAction(Command c, Displayable s)
{
if (c == cmExit)
{
destroyApp(true);
notifyDestroyed();
```

```
}
}
}

import javax.microedition.lcdui.game.*;
import javax.microedition.lcdui.*;
public class AnimatedSprite extends Sprite
{
public AnimatedSprite(Image image, int frameWidth, int frameHeight)
{
// Call sprite constructor
super(image, frameWidth, frameHeight);
}
}
import javax.microedition.lcdui.game.*;
import javax.microedition.lcdui.*;
public class AppleSprite extends Sprite
{
public AppleSprite(Image image)
{
// Sprite constructor
super(image);
// Set location on canvas
setRefPixelPosition(146, 35);
}
}
import javax.microedition.lcdui.game.*;
import javax.microedition.lcdui.*;
public class CubeSprite extends Sprite
{
public CubeSprite(Image image)
{
// Sprite constructor
super(image);
// Set location on canvas
setRefPixelPosition(170, 180);
}
}
import javax.microedition.lcdui.game.*;
import javax.microedition.lcdui.*;
public class ManSprite extends Sprite
{
private int x = 0, y = 0, // Current x/y
previous_x, previous_y; // Last x/y
private static final int MAN_WIDTH = 25; // Width in pixels
private static final int MAN_HEIGHT = 25; // Height in pixels
public ManSprite(Image image)
{
// Call sprite constructor
super(image);
}
public void moveLeft()
{
// If the man will not hit the left edge...
```

```
if (x > 0)
{
saveXY();
// If less than 3 from left, set to zero,
// otherwise, subtract 3 from current location
x = (x < 3 ? 0 : x - 3);
setPosition(x, y);
}
}
public void moveRight(int w)
{
// If the man will not hit the right edge...
if ((x + MAN_WIDTH) < w)
{
saveXY();
// If current x plus width of ball goes over right side,
// set to rightmost position. Otherwise add 3 to current location.
x = ((x + MAN_WIDTH > w) ? (w - MAN_WIDTH) : x + 3);
setPosition(x, y);
}
}
public void moveUp()
{
// If the man will not hit the top edge...
if (y > 0)
{
saveXY();
// If less than 3 from top, set to zero,
// otherwise, subtract 3 from current location.
y = (y < 3 ? 0 : y - 3);
setPosition(x, y);
}
}
public void moveDown(int h)
{
// If the man will not hit the bottom edge...
if ((y + MAN_HEIGHT) < h)
{
saveXY();
// If current y plus height of ball goes past bottom edge,
// set to bottommost position. Otherwise add 3 to current location.
y = ((y + MAN_WIDTH > h) ? (h - MAN_WIDTH) : y + 3);
setPosition(x, y);
}
}
/*-------------------------------------------------
* Save x and y, which are needed if collision is
* detected.
*-------------------------------------------------*/
private void saveXY()
{
// Save last position
previous_x = x;
previous_y = y;
```

```
}
/*-------------------------------------------------
* When a collision is detected, move back to
* the previous x/y.
*-------------------------------------------------*/
public void restoreXY()
{
x = previous_x;
y = previous_y;
setPosition(x, y);
}
}

import javax.microedition.lcdui.game.*;
import javax.microedition.lcdui.*;
public class StarSprite extends Sprite
{
public StarSprite(Image image)
{
// Sprite constructor
super(image);
// Set location on canvas
setRefPixelPosition(5, 65);
}
}

import javax.microedition.lcdui.game.*;
import javax.microedition.lcdui.*;
public class CollisionCanvas extends GameCanvas implements Runnable
{
private AnimatedSprite spSpiral; // Animated sprite
private static final int FRAME_WIDTH = 57; // Width of 1 frame
private static final int FRAME_HEIGHT = 53; // Height of 1 frame
private int canvas_width, canvas_height; // Save canvas info
private ManSprite spMan; // Man (moveable)
private AppleSprite spApple; // Apple (stationary)
private CubeSprite spCube; // Cube "
private StarSprite spStar; // Star "
private LayerManager lmgr; // Manage all layers
private boolean running = false; // Thread running?
private Collisions midlet; // Reference to main midlet
public CollisionCanvas(Collisions midlet)
{
// Gamecanvas constructor
super(true);
this.midlet = midlet;
try
{
// Nonanimated sprites
spMan = new ManSprite(Image.createImage("/man.png"));
spApple = new AppleSprite(Image.createImage("/apple.png"));
spCube = new CubeSprite(Image.createImage("/cube.png"));
spStar = new StarSprite(Image.createImage("/star.png"));
// Animated sprite
```

```
spSpiral = new AnimatedSprite(Image.createImage("/spiral.png"),
FRAME_WIDTH, FRAME_HEIGHT);
// Change the reference pixel to the middle of sprite
spSpiral.defineReferencePixel(FRAME_WIDTH / 2, FRAME_HEIGHT / 2);
// Center the sprite on the canvas
// (center of sprite is now in center of display)
spSpiral.setRefPixelPosition(getWidth() / 2, getHeight() / 2);
// Create and add to layer manager
lmgr = new LayerManager();
lmgr.append(spSpiral);
lmgr.append(spMan);
lmgr.append(spApple);
lmgr.append(spCube);
lmgr.append(spStar);
}
catch (Exception e)
{
System.out.println("Unable to read PNG image");
}
// Save canvas width and height
canvas_width = getWidth();
canvas_height = getHeight();
}
/*-------------------------------------------------
* Start thread
*-------------------------------------------------*/
public void start()
{
running = true;
Thread t = new Thread(this);
t.start();
}
/*-------------------------------------------------
* Main game loop
*-------------------------------------------------*/
public void run()
{
Graphics g = getGraphics();
while (running)
{
// Look for keypresses
checkForKeys();
if (checkForCollision() == false)
{
drawDisplay(g);
}
else
{
// Flash backlight and vibrate device
midlet.display.flashBacklight(500);
midlet.display.vibrate(500);
}
try
{
```

```
Thread.sleep(125);
}
catch (InterruptedException ie)
{
System.out.println("Thread exception");
}
}
}
/*-----------------------------------------------
* Check to see if ball collided with any other sprite.
*-----------------------------------------------*/
private boolean checkForCollision()
{
if (spMan.collidesWith(spSpiral, true) ||
spMan.collidesWith(spApple, true) ||
spMan.collidesWith(spCube, true) ||
spMan.collidesWith(spStar, true))
{
// Upon collision, restore the lasy x/y position
spMan.restoreXY();
return true;
}
else
return false;
}
/*-----------------------------------------------
* Upon keypresses, moveball...
*-----------------------------------------------*/
private void checkForKeys()
{
int keyState = getKeyStates();
if ((keyState & LEFT_PRESSED) != 0) {
spMan.moveLeft();
}
else if ((keyState & RIGHT_PRESSED) != 0) {
spMan.moveRight(canvas_width);
}
else if ((keyState & UP_PRESSED) != 0) {
spMan.moveUp();
}
else if ((keyState & DOWN_PRESSED) != 0) {
spMan.moveDown(canvas_height);
}
}
/*-----------------------------------------------
* Update display
*-----------------------------------------------*/
private void drawDisplay(Graphics g)
{
// Clear background
g.setColor(0xffffff);
g.fillRect(0, 0, getWidth(), getHeight());
// Animated sprite, show next frame in sequence
spSpiral.nextFrame();
```

```
// Paint all layers
lmgr.paint(g, 0, 0);
// Flush off-screen buffer to display
flushGraphics();
}
/*-------------------------------------------------
* Stop thread
*-------------------------------------------------*/
public void stop()
{
running = false;
}
}
```

Bài 44: **ReadWrite**, lấy mẫu có Index 3 và 4 làm Animated đặt vào vị

```
import java.io.*;
import javax.microedition.midlet.*;
import javax.microedition.rms.*;
public class ReadWrite extends MIDlet
{
	private RecordStore rs = null;
	static final String REC_STORE = "db_1";
	public ReadWrite()
	{
		openRecStore(); // Create the record store
		// Write a few records and read them back
		writeRecord("J2ME and MIDP");
		writeRecord("Wireless Technology");
		writeRecord("Wireless Programming");
		readRecords();
		closeRecStore(); // Close record store
		deleteRecStore(); // Remove the record store
	}
	public void destroyApp( boolean unconditional )
	{
	}
	public void startApp()
	{
	// There is no user interface, go ahead and shutdown
	destroyApp(false);
	notifyDestroyed();
	}
	public void pauseApp()
	{
	}
	public void openRecStore()
	{
		try
		{
		// Create record store if it does not exist
			rs = RecordStore.openRecordStore(REC_STORE, true );
		}
		catch (Exception e)
```

```
        {
                db(e.toString());
        }
}
public void closeRecStore()
{
        try
        {
                rs.closeRecordStore();
        }
        catch (Exception e)
        {
                db(e.toString());
        }
}
public void deleteRecStore()
{
        if (RecordStore.listRecordStores() != null)
        {
                try
                {
                        RecordStore.deleteRecordStore(REC_STORE);
                }
                catch (Exception e)
                {
                        db(e.toString());
                }
        }
}
public void writeRecord(String str)
{
        byte[] rec = str.getBytes();
        try
        {
                rs.addRecord(rec, 0, rec.length);
        }
        catch (Exception e)
        {
                db(e.toString());
        }
}
public void readRecords()
{
        try
        {
                byte[] recData = new byte[50];
                int len;
                for (int i = 1; i <= rs.getNumRecords(); i++)
                {
                        len = rs.getRecord( i, recData, 0 );
                        System.out.println("Record #" + i + ": " +
                        new String(recData, 0, len));
                        System.out.println("------------------------------");
                }
```

```
                }
                catch (Exception e)
                {
                        db(e.toString());
                }
        }

        private void db(String str)
        {
                System.err.println("Msg: " + str);
        }
}
```

Bài 45: ReadWriteStreams, lấy mẫu có Index 3 và 4 làm Animated đặt vào vị

```
import java.io.*;
import javax.microedition.midlet.*;
import javax.microedition.rms.*;

public class ReadWriteStreams extends MIDlet
{
 private RecordStore rs = null;    // Record store
 static final String REC_STORE = "db_1"; // Name of record store

 public ReadWriteStreams()
 {
  openRecStore();   // Create the record store

  writeTestData();  // Write a series of records
  readStream();     // Read back the records

  closeRecStore();  // Close record store
  deleteRecStore(); // Remove the record store
 }

 public void destroyApp( boolean unconditional )
 {
 }

 public void startApp()
 {
  // There is no user interface, go ahead and shutdown
  destroyApp(false);
  notifyDestroyed();
 }

 public void pauseApp()
 {
 }

 public void openRecStore()
 {
  try
  {
```

```
    // The second parameter indicates that the record store
          // should be created if it does not exist
    rs = RecordStore.openRecordStore(REC_STORE, true );
  }
  catch (Exception e)
  {
    db(e.toString());
  }
}

public void closeRecStore()
{
  try
  {
    rs.closeRecordStore();
  }
  catch (Exception e)
  {
    db(e.toString());
  }
}

public void deleteRecStore()
{
  if (RecordStore.listRecordStores() != null)
  {
    try
    {
      RecordStore.deleteRecordStore(REC_STORE);
    }
    catch (Exception e)
    {
      db(e.toString());
    }
  }
}

/*-------------------------------------------------
* Create three arrays to write to record store
*-------------------------------------------------*/
public void writeTestData()
{
  String[] strings = {"Text 1", "Text 2"};
  boolean[] booleans = {false, true};
  int[] integers = {1 , 2};

  writeStream(strings, booleans, integers);
}

/*-------------------------------------------------
* Write to record store using streams.
*-------------------------------------------------*/
public void writeStream(String[] sData, boolean[] bData, int[] iData)
{
```

```
  try
  {
    // Write data into an internal byte array
   ByteArrayOutputStream strmBytes = new ByteArrayOutputStream();

   // Write Java data types into the above byte array
   DataOutputStream strmDataType = new DataOutputStream(strmBytes);

   byte[] record;

   for (int i = 0; i < sData.length; i++)
   {
    // Write Java data types
    strmDataType.writeUTF(sData[i]);
    strmDataType.writeBoolean(bData[i]);
    strmDataType.writeInt(iData[i]);

    // Clear any buffered data
    strmDataType.flush();

    // Get stream data into byte array and write record
    record = strmBytes.toByteArray();
    rs.addRecord(record, 0, record.length);

    // Toss any data in the internal array so writes
    // starts at beginning (of the internal array)
    strmBytes.reset();
   }

   strmBytes.close();
   strmDataType.close();

  }
  catch (Exception e)
  {
   db(e.toString());
  }
}

/*-------------------------------------------------
* Read from the record store using streams
*-------------------------------------------------*/
public void readStream()
{
  try
  {
   // Careful: Make sure this is big enough!
   // Better yet, test and reallocate if necessary
   byte[] recData = new byte[50];

   // Read from the specified byte array
   ByteArrayInputStream strmBytes = new ByteArrayInputStream(recData);

   // Read Java data types from the above byte array
```

```
    DataInputStream strmDataType = new DataInputStream(strmBytes);

    for (int i = 1; i <= rs.getNumRecords(); i++)
    {
     // Get data into the byte array
     rs.getRecord(i, recData, 0);

     // Read back the data types
     System.out.println("Record #" + i);
     System.out.println("UTF: " + strmDataType.readUTF());
     System.out.println("Boolean: " + strmDataType.readBoolean());
     System.out.println("Int: " + strmDataType.readInt());
     System.out.println("--------------------");

     // Reset so read starts at beginning of array
     strmBytes.reset();
    }

    strmBytes.close();
    strmDataType.close();

   }
   catch (Exception e)
   {
    db(e.toString());
   }
  }


  private void db(String str)
  {
   System.err.println("Msg: " + str);
  }
}
```

Bài 46: SimpleSort, lấy mẫu có Index 3 và 4 làm Animated đặt vào vị

```
import java.io.*;
import javax.microedition.midlet.*;
import javax.microedition.rms.*;

public class SimpleSort extends MIDlet
{
  private RecordStore rs = null;
  static final String REC_STORE = "db_1";

  public SimpleSort()
  {
   openRecStore();   // Create the record store

   // Write a few records
   writeRecord("Yen Chi");
   writeRecord("Cam Tu");
   writeRecord("An Binh");
```

```
  writeRecord("Hanh Dung");
  writeRecord("Sam Khang");
  writeRecord("Viet Dung");

  // Read back with enumerator, sorting the results
  readRecords();

  closeRecStore();  // Close record store
  deleteRecStore(); // Remove the record store
}

public void destroyApp( boolean unconditional )
{
}

public void startApp()
{
  // There is no user interface, go ahead and shutdown
  destroyApp(false);
  notifyDestroyed();
}

public void pauseApp()
{
}

public void openRecStore()
{
  try
  {
    // The second parameter indicates that the record store
          // should be created if it does not exist
    rs = RecordStore.openRecordStore(REC_STORE, true );
  }
  catch (Exception e)
  {
    db(e.toString());
  }
}

public void closeRecStore()
{
  try
  {
    rs.closeRecordStore();
  }
  catch (Exception e)
  {
    db(e.toString());
  }
}

public void deleteRecStore()
{
```

```
  if (RecordStore.listRecordStores() != null)
  {
    try
    {
      RecordStore.deleteRecordStore(REC_STORE);
    }
    catch (Exception e)
    {
      db(e.toString());
    }
  }
}

public void writeRecord(String str)
{
  byte[] rec = str.getBytes();

  try
  {
    rs.addRecord(rec, 0, rec.length);
  }
  catch (Exception e)
  {
    db(e.toString());
  }
}

public void readRecords()
{
  try
  {
    if (rs.getNumRecords() > 0)
    {
      Comparator comp = new Comparator();

      RecordEnumeration re = rs.enumerateRecords(null, comp, false);
      while (re.hasNextElement())
      {
        // Calls String constructor that takes an array of bytes as input
        String str = new String(re.nextRecord());

        System.out.println(str);
        System.out.println("------------------------------");

      }
    }
  }
  catch (Exception e)
  {
    db(e.toString());
  }
}
```

```
  private void db(String str)
  {
    System.err.println("Msg: " + str);
  }
}


/*-------------------------------------------------
* Compares two records to determine sort order
*-------------------------------------------------*/
class Comparator implements RecordComparator
{
  public int compare(byte[] rec1, byte[] rec2)
  {
    String str1 = new String(rec1), str2 = new String(rec2);

    int result = str1.compareTo(str2);
    if (result == 0)
      return RecordComparator.EQUIVALENT;
    else if (result < 0)
      return RecordComparator.PRECEDES;
    else
      return RecordComparator.FOLLOWS;
  }
}
```

Bài 47: **StringSort**, lấy mẫu có Index 3 và 4 làm Animated đặt vào vị

```
import java.io.*;
import javax.microedition.midlet.*;
import javax.microedition.rms.*;

public class StringSort extends MIDlet
{
  private RecordStore rs = null;    // Record store
  static final String REC_STORE = "db_3"; // Name of record store

  public StringSort()
  {
    openRecStore();   // Create the record store

    writeTestData();  // Write a series of records
    readStream();     // Read back the records

    closeRecStore();  // Close record store
    deleteRecStore(); // Remove the record store
  }

  public void destroyApp( boolean unconditional )
  {
  }

  public void startApp()
  {
    // There is no user interface, go ahead and shutdown
```

```
  destroyApp(false);
  notifyDestroyed();
}

public void pauseApp()
{
}

public void openRecStore()
{
  try
  {
    // The second parameter indicates that the record store
          // should be created if it does not exist
    rs = RecordStore.openRecordStore(REC_STORE, true );
  }
  catch (Exception e)
  {
    db(e.toString());
  }
}

public void closeRecStore()
{
  try
  {
    rs.closeRecordStore();
  }
  catch (Exception e)
  {
    db(e.toString());
  }
}

public void deleteRecStore()
{
  if (RecordStore.listRecordStores() != null)
  {
    try
    {
      RecordStore.deleteRecordStore(REC_STORE);
    }
    catch (Exception e)
    {
      db(e.toString());
    }
  }
}

/*-------------------------------------------------
* Create three arrays to write to record store
*-------------------------------------------------*/
public void writeTestData()
{
```

```
 String[] names = {"Thu", "Hanh", "Yen", "Khanh","Anh"};
 boolean[] sex = {false,true, false, true,true};
 int[] rank = {2, 0, 4, 3,1};

 writeStream(names, sex, rank);
}

/*-------------------------------------------------
* Write to record store using streams.
*-------------------------------------------------*/
public void writeStream(String[] sData, boolean[] bData, int[] iData)
{
  try
  {
    // Write data into an internal byte array
   ByteArrayOutputStream strmBytes = new ByteArrayOutputStream();

   // Write Java data types into the above byte array
   DataOutputStream strmDataType = new DataOutputStream(strmBytes);

   byte[] record;

   for (int i = 0; i < sData.length; i++)
   {
    // Write Java data types
    strmDataType.writeUTF(sData[i]);
    strmDataType.writeBoolean(bData[i]);
    strmDataType.writeInt(iData[i]);

    // Clear any buffered data
    strmDataType.flush();

    // Get stream data into byte array and write record
    record = strmBytes.toByteArray();
    rs.addRecord(record, 0, record.length);

    // Toss any data in the internal array so writes
    // starts at beginning (of the internal array)
    strmBytes.reset();
   }

   strmBytes.close();
   strmDataType.close();

  }
  catch (Exception e)
  {
   db(e.toString());
  }
}

/*-------------------------------------------------
* Read from the record store using streams
*-------------------------------------------------*/
```

```
public void readStream()
{
 try
 {
  // Careful: Make sure this is big enough!
  // Better yet, test and reallocate if necessary
  byte[] recData = new byte[50];

   // Read from the specified byte array
  ByteArrayInputStream strmBytes = new ByteArrayInputStream(recData);

  // Read Java data types from the above byte array
  DataInputStream strmDataType = new DataInputStream(strmBytes);

  if (rs.getNumRecords() > 0)
  {
   ComparatorString comp = new ComparatorString();

   int i = 1;
   RecordEnumeration re = rs.enumerateRecords(null, comp, false);
   while (re.hasNextElement())
   {
    // Get data into the byte array
    rs.getRecord(re.nextRecordId(), recData, 0);

    // Read back the data types
    System.out.println("Record #" + i++);

    System.out.println("Name: " + strmDataType.readUTF());
    if (strmDataType.readBoolean())
    System.out.println("Sex: Male");
    else
    System.out.println("Sex: Female" );
    //System.out.println("Sex: " + strmDataType.readBoolean());
    System.out.println("Rank: " + strmDataType.readInt());
    System.out.println("--------------------");

    // Reset so read starts at beginning of array
    strmBytes.reset();
   }

   comp.compareStringClose();

   // Free enumerator
   re.destroy();
  }

  strmBytes.close();
  strmDataType.close();

 }
 catch (Exception e)
 {
  db(e.toString());
```

```
    }
  }


  /*-------------------------------------------------*/
  private void db(String str)
  {
    System.err.println("Msg: " + str);
  }
}

/*-------------------------------------------------*/
class ComparatorString implements RecordComparator
{
  private byte[] recData = new byte[10];

  // Read from a specified byte array
  private ByteArrayInputStream strmBytes = null;

  // Read Java data types from the above byte array
  private DataInputStream strmDataType = null;

  public void compareStringClose()
  {
    try
    {
      if (strmBytes != null)
        strmBytes.close();
      if (strmDataType != null)
        strmDataType.close();
    }
    catch (Exception e)
    {}
  }

  public int compare(byte[] rec1, byte[] rec2)
  {
    String str1, str2;

    try
    {
      // If either record is larger than our buffer, reallocate
      int maxsize = Math.max(rec1.length, rec2.length);
      if (maxsize > recData.length)
        recData = new byte[maxsize];

      // Read record #1
      // Only need one read because the string to
      // sort on is the first "field" in the record
      strmBytes = new ByteArrayInputStream(rec1);
      strmDataType = new DataInputStream(strmBytes);
      str1 = strmDataType.readUTF();

      // Read record #2
```

```
      strmBytes = new ByteArrayInputStream(rec2);
      strmDataType = new DataInputStream(strmBytes);
      str2 = strmDataType.readUTF();

      // Compare record #1 and #2
      int result = str1.compareTo(str2);
      if (result == 0)
        return RecordComparator.EQUIVALENT;
      else if (result < 0)
        return RecordComparator.PRECEDES;
      else
        return RecordComparator.FOLLOWS;

    }
    catch (Exception e)
    {
     return RecordComparator.EQUIVALENT;
    }
  }
}
```

Bài 48: **SimpleSearch**, lấy mẫu có Index 3 và 4 làm Animated đặt vào vị

```
import java.io.*;
import javax.microedition.midlet.*;
import javax.microedition.rms.*;
import javax.microedition.lcdui.*;

public class SimpleSearch extends MIDlet implements CommandListener
{
  private Display display;           // Reference to Display object
  private Form fmMain;         // The main form
  private StringItem siMatch;  // The matching text, if any
  private Command cmFind;      // Command to search record store
  private Command cmExit;      // Command to insert items
  private TextField tfFind;    // Search text as requested by user
  private RecordStore rs = null;    // Record store
  static final String REC_STORE = "db_1"; // Name of record store

  public SimpleSearch()
  {
    display = Display.getDisplay(this);

    // Define textfield, stringItem and commands
    tfFind = new TextField("Find", "", 10, TextField.ANY);
    siMatch = new StringItem(null, null);
    cmExit = new Command("Exit", Command.EXIT, 1);
    cmFind = new Command("Find", Command.SCREEN, 2);

    // Create the form, add commands
    fmMain = new Form("Record Search");
    fmMain.addCommand(cmExit);
    fmMain.addCommand(cmFind);
```

```
  // Append textfield and stringItem
  fmMain.append(tfFind);
  fmMain.append(siMatch);

  // Capture events
  fmMain.setCommandListener(this);

  //--------------------------------
  // Open and write to record store
  //--------------------------------
  openRecStore();   // Create the record store
  writeTestData();  // Write a series of records
}

public void destroyApp( boolean unconditional )
{
  closeRecStore();  // Close record store
}

public void startApp()
{
  display.setCurrent(fmMain);
}

public void pauseApp()
{
}

public void openRecStore()
{
  try
  {
    // The second parameter indicates that the record store
          // should be created if it does not exist
    rs = RecordStore.openRecordStore(REC_STORE, true );
  }
  catch (Exception e)
  {
    db(e.toString());
  }
}

public void closeRecStore()
{
  try
  {
    rs.closeRecordStore();
  }
  catch (Exception e)
  {
    db(e.toString());
  }
}
```

```
/*------------------------------------------------
* Create array of data to write into record store
*------------------------------------------------*/
public void writeTestData()
{
  String[] Giadinh = {
              "Lam quen: vo tinh hay co y",
              "Ket ban: den nha choi hay ru di choi... vai lan",
              "Loi to tinh: khon ngoan nhat loai hoa ma ho thich",
              "Cau hon: khi doi tac da that san sang"};

  writeRecords(Giadinh);
}

/*------------------------------------------------
* Write to record store.
*------------------------------------------------*/
public void writeRecords(String[] sData)
{
  byte[] record;

  try
  {
    // Only add the records once
    if (rs.getNumRecords() > 0)
      return;

    for (int i = 0; i < sData.length; i++)
    {
      record = sData[i].getBytes();
      rs.addRecord(record, 0, record.length);
    }
  }
  catch (Exception e)
  {
    db(e.toString());
  }
}

/*------------------------------------------------
* Search using enumerator and record filter
*------------------------------------------------*/
private void searchRecordStore()
{
  try
  {
    // Record store is not empty
    if (rs.getNumRecords() > 0)
    {
      // Setup the search filter with the user requested text
      SearchFilter search = new SearchFilter(tfFind.getString());

      RecordEnumeration re = rs.enumerateRecords(search, null, false);
```

```
        // A match was found using the filter
        if (re.numRecords() > 0)
          // Show match in the stringItem on the form
          siMatch.setText(new String(re.nextRecord()));

        re.destroy();   // Free enumerator
      }
    }
    catch (Exception e)
    {
      db(e.toString());
    }
  }

  public void commandAction(Command c, Displayable s)
  {
    if (c == cmFind)
    {
      searchRecordStore();
    }
    else if (c == cmExit)
    {
      destroyApp(false);
      notifyDestroyed();
    }
  }


  private void db(String str)
  {
    System.err.println("Msg: " + str);
  }
}

/*--------------------------------------------------*/
class SearchFilter implements RecordFilter
{
  private String searchText = null;

  public SearchFilter(String searchText)
  {
    // This is the text to search for
    this.searchText = searchText.toLowerCase();
  }

  public boolean matches(byte[] candidate)
  {
    String str = new String(candidate).toLowerCase();

    // Look for a match
    if (searchText != null && str.indexOf(searchText) != -1)
      return true;
    else
      return false;
```

```
  }
}
```

Bài 49: **SearchStreams**, lấy mẫu có Index 3 và 4 làm Animated đặt vào vị

```
import java.io.*;
import javax.microedition.midlet.*;
import javax.microedition.rms.*;
import javax.microedition.lcdui.*;

public class SearchStreams extends MIDlet implements CommandListener
{
  private Display display;           // Reference to Display object
  private Form fmMain;         // The main form
  private StringItem siMatch;  // The matching text, if any
  private Command cmFind;      // Command to search record store
  private Command cmExit;      // Command to insert items
  private TextField tfFind;    // Search text
  private RecordStore rs = null;    // Record store
  static final String REC_STORE = "db_2"; // Name of record store

  public SearchStreams()
  {
    display = Display.getDisplay(this);

    // Define textfield, stringItem and commands
    tfFind = new TextField("Find", "", 10, TextField.ANY);
    siMatch = new StringItem(null, null);
    cmExit = new Command("Exit", Command.EXIT, 1);
    cmFind = new Command("Find", Command.SCREEN, 2);

    // Create the form, add commands
    fmMain = new Form("Record Search");
    fmMain.addCommand(cmExit);
    fmMain.addCommand(cmFind);

    // Append textfield and stringItem
    fmMain.append(tfFind);
    fmMain.append(siMatch);

    // Capture events
    fmMain.setCommandListener(this);

    //--------------------------------
    // Open and write to record store
    //--------------------------------
    openRecStore();   // Create the record store
    writeTestData();  // Write a series of records
  }

  public void destroyApp( boolean unconditional )
  {
    closeRecStore();  // Close record store
  }
```

```
public void startApp()
{
  display.setCurrent(fmMain);
}

public void pauseApp()
{
}

public void openRecStore()
{
  try
  {
    // The second parameter indicates that the record store
        // should be created if it does not exist
    rs = RecordStore.openRecordStore(REC_STORE, true );
  }
  catch (Exception e)
  {
    db(e.toString());
  }
}

public void closeRecStore()
{
  try
  {
    rs.closeRecordStore();
  }
  catch (Exception e)
  {
    db(e.toString());
  }
}

/*-------------------------------------------------
* Create three arrays to write into record store
*-------------------------------------------------*/
public void writeTestData()
{
 String[] names = {"Lan : Lop C04 CNTT HVCNBCVT",
                "Thu : K45 CNTT Dai Hoc Bach Khoa HN",
                "Hoai Anh : K39 QTDN Truong Kinh Te Quoc Dan",
                 "Yen Chi : Lop Anh Ngu Truong Dai Hoc Ngoai Ngu HN"};
 boolean[] sex = {true, false, true, true};
 int[] rank = {3, 0, 1, 2};

 writeStream(names, sex, rank);
}

/*-------------------------------------------------
* Write to record store using streams.
*-------------------------------------------------*/
public void writeStream(String[] sData, boolean[] bData, int[] iData)
```

```
{
 try
 {
  // Only add the records once
  if (rs.getNumRecords() > 0)
    return;

   // Write data into an internal byte array
  ByteArrayOutputStream strmBytes = new ByteArrayOutputStream();

  // Write Java data types into the above byte array
  DataOutputStream strmDataType = new DataOutputStream(strmBytes);

  byte[] record;

  for (int i = 0; i < sData.length; i++)
  {
    // Write Java data types
    strmDataType.writeUTF(sData[i]);
    strmDataType.writeBoolean(bData[i]);
    strmDataType.writeInt(iData[i]);

    // Clear any buffered data
    strmDataType.flush();

    // Get stream data into byte array and write record
    record = strmBytes.toByteArray();
    rs.addRecord(record, 0, record.length);

    // Toss any data in the internal array so writes
    // starts at beginning (of the internal array)
    strmBytes.reset();
  }

  strmBytes.close();
  strmDataType.close();
 }
 catch (Exception e)
 {
  db(e.toString());
 }
}

/*-------------------------------------------------
* Search using enumerator and record filter
*-------------------------------------------------*/
private void searchRecordStore()
{
try
  {
   // Record store is not empty
   if (rs.getNumRecords() > 0)
   {
    // Setup the search filter with the user requested text
```

```
        SearchFilter search =
            new SearchFilter(tfFind.getString());

        RecordEnumeration re =
            rs.enumerateRecords(search, null, false);

        // A match was found using the filter
        if (re.numRecords() > 0)
        {
         // Read from the specified byte array
         ByteArrayInputStream strmBytes =
              new ByteArrayInputStream(re.nextRecord());

         // Read Java data types from the above byte array
         DataInputStream strmDataType =
              new DataInputStream(strmBytes);

         // Show matching result in stringItem component on form
         siMatch.setText(strmDataType.readUTF());

         search.searchFilterClose(); // Close record filter
         strmBytes.close();          // Close stream
         strmDataType.close();       // Close stream
         re.destroy();               // Free enumerator
        }
       }
      }
      catch (Exception e)
      {
       db(e.toString());
      }

    }

    public void commandAction(Command c, Displayable s)
    {
      if (c == cmFind)
      {
       searchRecordStore();
      }
      else if (c == cmExit)
      {
       destroyApp(false);
       notifyDestroyed();
      }
    }

/*-------------------------------------------------*/
    private void db(String str)
    {
      System.err.println("Msg: " + str);
    }
}
```

```
/*-------------------------------------------------
* Search for text within a record
* Each record passed in contains multiple Java data
* types (String, boolean and integer)
*-------------------------------------------------*/
class SearchFilter implements RecordFilter
{
  private String searchText = null;

   // Read from a specified byte array
  private ByteArrayInputStream strmBytes = null;

  // Read Java data types from the above byte array
  private DataInputStream strmDataType = null;

  public SearchFilter(String searchText)
  {
    // This is the text to search for
    this.searchText = searchText.toLowerCase();
  }

  // Cleanup
  public void searchFilterClose()
  {
    try
    {
      if (strmBytes != null)
        strmBytes.close();
      if (strmDataType != null)
        strmDataType.close();
    }
    catch (Exception e)
    {}
  }

  public boolean matches(byte[] candidate)
  {
    String str = null;

    try
    {
      strmBytes = new ByteArrayInputStream(candidate);
      strmDataType = new DataInputStream(strmBytes);

      // Although 3 pieces of data were written to
      // the record (String, boolean and integer)
      // we only need one read because the string to
      // search is the first "field" in the record
      str = strmDataType.readUTF().toLowerCase();
    }
    catch (Exception e)
    {
      return false;
    }
```

```
    // Look for a match
    if (str != null && str.indexOf(searchText) != -1)
      return true;
    else
      return false;
  }
}
```

Bài 50: **EliminatorBasicMenu**, lấy mẫu có Index 3 và 4 làm Animated đặt vào vị

Bài 51: **EliminatorSubMenu**, lấy mẫu có Index 3 và 4 làm Animated đặt vào vị

Bài 52: **EliminatorScrolling**, lấy mẫu có Index 3 và 4 làm Animated đặt vào vị