Bài thực hành: Đồ họa và Multimedia trong Java ME

1.1 Canvas

1.1.1 Tạo một Canvas đơn giản

- Hướng dẫn:Canvas là lớp trừu tượng; để tạo các canvas bằng cách kế thừa lớp này và phải cài đặt lại phương thức paint(Graphics g), là phương thức thực sự vẽ cái gì đó lên thiết bị. Lớp Canvas và Graphics làm việc cùng nhau nhằm cung cấp các kiểm soát mức thấp trên một thiết bị.

Ví dụ sau vẽ một Canvas màu đỏ với chữ Hello World trắng lên màn hình.

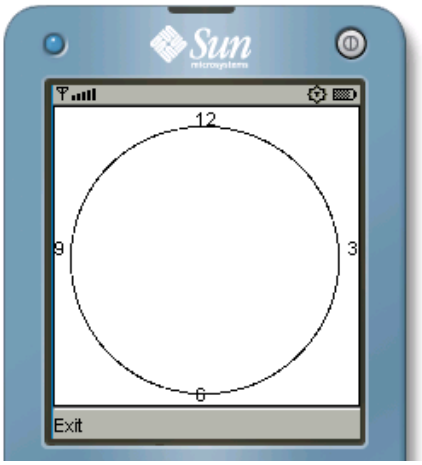
```
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Graphics;
import javax.microedition.midlet.MIDlet;

public class CanvasMyMidlet extends MIDlet {
  public CanvasMyMidlet() { // constructor
  }

  public void startApp() {
    Canvas canvas = new MyCanvas();
    Display display = Display.getDisplay(this);
    display.setCurrent(canvas);
  }

  public void pauseApp() {
  }

  public void destroyApp(boolean unconditional) {
  }
}

class MyCanvas extends Canvas {
  // viết lại hàm paint cho MyCanvas được mở rộng từ Canvas
  public void paint(Graphics g) {
    g.setColor(255, 0, 0);
    g.fillRect(0, 0, getWidth(), getHeight());
    g.setColor(255, 255, 255);
    g.drawString("Hello World!", 0, 0, g.TOP | g.LEFT);
  }
}
```

1.1.2 Lấy các thuộc tính của Canvas

Ví dụ sau dùng các phương thức để lấy giá trị các thuộc tính của Canvas và hiển thị lên màn hình.

```
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.Form;
import javax.microedition.lcdui.Graphics;
import javax.microedition.lcdui.StringItem;
import javax.microedition.midlet.MIDlet;

public class AttributesMIDlet extends MIDlet
                        implements CommandListener {

    private Display display;

    protected boolean started;
    private Command exitCommand;

    protected void startApp() {
        if (!started) {
            display = Display.getDisplay(this);
            Canvas canvas = new DummyCanvas();
            Form form = new Form("Attributes");
            exitCommand = new Command("Exit", Command.EXIT, 0);
            form.addCommand(exitCommand);

            boolean isColor = display.isColor();
            form.append(new StringItem(isColor ? "Colors: " : "Grays: ",
                                String.valueOf(display.numColors())));
            form.append(new StringItem("Width: ", String.valueOf(canvas.getWidth())));
            form.append(new StringItem("Height: ", String.valueOf(canvas.getHeight())));
            form.append(new StringItem("Pointer? ", String.valueOf(canvas.hasPointerEvents())));
            form.append(new StringItem("Motion? ",String.valueOf(canvas.hasPointerMotionEvents())));
            form.append(new StringItem("Repeat? ",String.valueOf(canvas.hasRepeatEvents())));
            form.append(new StringItem("Buffered? ", String.valueOf(canvas.isDoubleBuffered())));

            form.setCommandListener(this);

            display.setCurrent(form);

            started = true;
        }
    }


    protected void pauseApp() {
    }

    protected void destroyApp(boolean unconditional) {
    }

    public void commandAction(Command c, Displayable d) {
        if (c == exitCommand) {
            notifyDestroyed();
        }
    }
    static class DummyCanvas extends Canvas {
        protected void paint(Graphics g) {
        }
    }
}
```

1.1.3 Vẽ đồng hồ



Ví dụ sẽ sử dụng đối tượng Graphics và dùng các phương thức vẽ cung tròn **drawArc**, vẽ chuỗi **drawString** để vẽ các đối tượng cần thiết lên màn hình

```
ClockCanvas(int hour, int minute, int second) {
  this.hour = hour;
  this.minute = minute;
  this.second = second;
}

public void paint(Graphics g) {
  width = getWidth();
  height = getHeight();

  g.setGrayScale(255);
  g.fillRect(0, 0, width - 1, height - 1);
  g.setGrayScale(0);
  g.drawRect(0, 0, width - 1, height - 1);

  clockRadius = Math.min(width, height) - 20;

  xCenter = getWidth() / 2;
  yCenter = getHeight() / 2;

  // Vẽ cung tròn tròn tại điểm có tọa độ x=10, y=12 với góc là 360 độ
  g.drawArc(10, 12, clockRadius, clockRadius, 0, 360);

  // Vẽ các số 3, 6, 9, 12 tại các vị trí tương ứng
  g.drawString("12", xCenter, 0, Graphics.TOP | Graphics.HCENTER);
  g.drawString("9", 1, yCenter, Graphics.BASELINE | Graphics.LEFT);
  g.drawString("3", width - 1, yCenter, Graphics.BASELINE | Graphics.RIGHT);
  g.drawString("6", xCenter, height, Graphics.BOTTOM | Graphics.RIGHT);

}
```

1.2 Graphics

*1.2.1 Sử dụng hàm **setColor***

Ví dụ sau dùng hàm **setColor**để thiết lập màu cần vẽ với mã màu 0xFFFF00 (màu vàng) cho đối tượng Graphics và vẽ 1 đường thẳng có màu đã thiết lập trên màn hình

```
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Graphics;
import javax.microedition.midlet.MIDlet;

public class SetColorGraphicsMIDlet extends MIDlet {
  private Display display;
  protected void startApp() {
    Canvas canvas = new LineCanvas();
    display = Display.getDisplay(this);
    display.setCurrent(canvas);
  }
  protected void pauseApp() {
  }
  protected void destroyApp(boolean unconditional) {
  }
}

class LineCanvas extends Canvas {
  public void paint(Graphics g) {
    int width = getWidth();
    int height = getHeight();
    g.setColor(0xFFFF00);
    g.drawLine(0, height / 4, width - 1, height / 4);
  }
}
```

1.2.2 Sử dụng hàm SetStroke để thiết lập nét vẽ

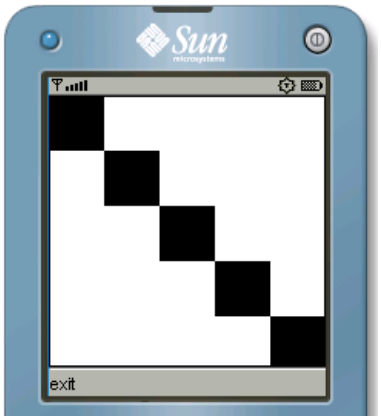
Ví dụ sau vẽ một đường thẳng với nét vẽ là nét đứt *Graphics.DOTTED*, lên màn hình, ngoài ra còn thuộc tính khác để thiết lập nét vẽ cho đối tượng Graphics là *Graphics.SOLID* (nét liền).

```
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Graphics;
import javax.microedition.midlet.MIDlet;

public class SetStrokeStyleGraphicsMIDlet extends MIDlet {
  private Display display;
  protected void startApp() {
    Canvas canvas = new LineCanvas();
    display = Display.getDisplay(this);
    display.setCurrent(canvas);
  }
  protected void pauseApp() {
  }
  protected void destroyApp(boolean unconditional) {
  }
}

class LineCanvas extends Canvas {
  public void paint(Graphics g) {
    int width = getWidth();
    int height = getHeight();
    g.setColor(0);
    g.setStrokeStyle(Graphics.DOTTED);

    g.drawLine(0, height / 2, width - 1, height / 2);
  }
}
```

1.2.3 *Sử dụng hàm SetGrayStyle*

```
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Graphics;
import javax.microedition.midlet.MIDlet;

public class SetGrayScaleGraphicsMIDlet extends MIDlet {
  private Display display;
  protected void startApp() {
    Canvas canvas = new LineCanvas();
    display = Display.getDisplay(this);
    display.setCurrent(canvas);
  }
  protected void pauseApp() {
  }
  protected void destroyApp(boolean unconditional) {
  }
}

class LineCanvas extends Canvas {
  public void paint(Graphics g) {
    int width = getWidth();
    int height = getHeight();
    g.setColor(0);
    g.setGrayScale(100);

    g.drawLine(0,  2, width - 1, height / 2);
  }
```

1.2.4 *Dịch chuyển tọa độ vẽ*



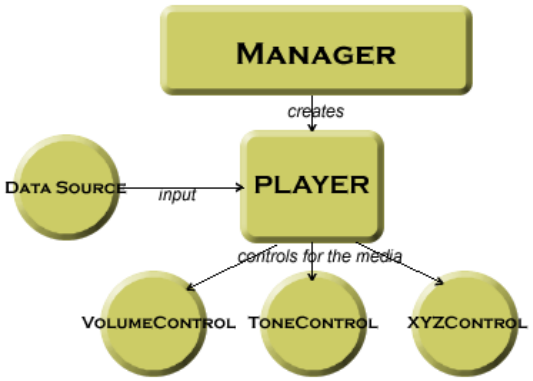
Ví dụ sau vẽ cách hình vuông có cạnh là **width/5** và sử dụng phương thức translate để dịch chuyển tọa độ điểm vẽ kế tiếp x=**width/5,**y= **width/5** của đối tương Graphics

```
public void startApp() {
  Display display = Display.getDisplay(this);

  Displayable d = new OrigTransCanvas();
  exitCommand = new Command("exit", Command.EXIT, 1);

  d.addCommand(exitCommand);
  d.setCommandListener(this);

  display.setCurrent(d);
}

class OrigTransCanvas extends Canvas {
int width = 0;

int height = 0;

public void paint(Graphics g) {
  width = getWidth();
  height = getHeight();

  g.setGrayScale(255);
  g.fillRect(0, 0, width - 1, height - 1);
  g.setGrayScale(0);
  g.drawRect(0, 0, width - 1, height - 1);

  int counts = 5;
  int step = width / 5;
  for (int i = 1; i <= 5; i++) {
    g.fillRect(0, 0, step, step);
    g.translate(step, step);
  }
}
```

1.3 MMAPI

MMAPI định nghĩa superset các khả năng đa phương tiện hiện diện trong MIDP 2.0. MMAPI được xây dựng trên một trừu tượng mức cao với tất cả các thiết bị đa phương tiện có thể được trên một thiết bị bị giới hạn tài nguyên (resource-limited). Sự trừu tượng này rõ ràng ở trong 3 class hình thành các khối thao tác mà bạn làm với API này. Những lớp này là các giao diện **Player** và **Control,** và lớp **Manager**.Một lớp khác nữa, lớp trừu tượng **DataSource**, được dùng để định vị các tài nguyên.

Lớp **Manager** để tạo các thể hiện **Player** cho phương tiện khác nhau bằng cách chỉ rõ các thể hiện **DataSource.** Các thể hiện **Player** vì thế được tạo ra là có khả năng cấu hình bằng cách sử dụng các thể hiện **Control**.Chẳng hạn, hầu hết mọi thể hiện **Player** sẽ hỗ trợ về mặt lý thuyết một **VolumeControl** để điều khiển volume của **Player.**Hình sau cho thấy quá trình này.



Ví dụ sau hướng dẫn tạo 1 ứng dụng play một file audio đơn giản,play chỉ các bài hát trong danh sách. Khi chọn bài hát này, màn hình thay đổi với văn bản "Playing media" và có 2 lệnh xuất hiên bên dưới: Stop và Pause, cho người dùng lựa chọn. Media đang được chơi và người dùng có thể tạm dừng lại hay dừng hẳn và trở về danh sách bài hát.

```
import java.util.Hashtable;
import java.util.Enumeration;

import javax.microedition.lcdui.List;
import javax.microedition.lcdui.Form;
import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.CommandListener;

import javax.microedition.media.Player;
import javax.microedition.media.Manager;
import javax.microedition.media.PlayerListener;

public class MediaMIDlet extends MIDlet
  implements CommandListener, PlayerListener {

  private Display display;
  private List itemList;
  private Form form;

  private Command stopCommand;
  private Command pauseCommand;
  private Command startCommand;

  private Hashtable items;
  private Hashtable itemsInfo;

  private Player player;

  public MediaMIDlet() {
    display = Display.getDisplay(this);
    // Tạo danh sách cái item để bạn có thể chọn bài để play
    itemList = new List("Select an item to play", List.IMPLICIT);

    // lệnh stop, pause và restart
    stopCommand = new Command("Stop", Command.STOP, 1);
    pauseCommand = new Command("Pause", Command.ITEM, 1);
    startCommand = new Command("Start", Command.ITEM, 1);
```

```
  // form hiển thị khi item được play
  form = new Form("Playing media");

  // thêm cách lệnh stop và pause media
  form.addCommand(stopCommand);
  form.addCommand(pauseCommand);
  form.setCommandListener(this);

  // tạo hastable cho các item
  items = new Hashtable();

  // hastable lưu giữ thông tin về các item
  itemsInfo = new Hashtable();

  // add item và thông tin item vào hastable
  items.put("Siren from jar", "file://siren.wav");
  itemsInfo.put("Siren from jar", "audio/x-wav");
}

public void startApp() {

  // Khi MIDlet được bắt đầu, sử dụng danh sách item để hiển thị item
  for(Enumeration en = items.keys(); en.hasMoreElements();) {
  itemList.append((String)en.nextElement(), null);
}

  itemList.setCommandListener(this);

  // hiển thị danh sách khi MIDlet được bắt đầu
  display.setCurrent(itemList);
}

public void pauseApp() {
  // tạm dừng player
  try {
    if(player != null) player.stop();
  } catch(Exception e) {}
}

public void destroyApp(boolean unconditional) {
  if(player != null) player.close(); // close player
}
```

```
public void commandAction(Command command, Displayable disp) {

  // danh sách được hiển thị, người dùng sẽ play item
  if(disp instanceof List) {
    List list = ((List)disp);

    String key = list.getString(list.getSelectedIndex());

    // play file được chọn
    try {
      playMedia((String)items.get(key), key);
    } catch (Exception e) {
      System.err.println("Unable to play: " + e);
      e.printStackTrace();
    }
  } else if(disp instanceof Form) {

    // file đang được play nếu form hiển thị
    // và người dùng có thể đang muốn pause hoặc stop item
    try {

      if(command == stopCommand) {

        player.close(); // đóng player
        display.setCurrent(itemList); // hiển thị lại danh sách
        form.removeCommand(startCommand); // xóa start command
        form.addCommand(pauseCommand); // thêm pause command

      } else if(command == pauseCommand) { // nếu đang dừng

        player.stop(); // dừng media
        form.removeCommand(pauseCommand); // xóa pause command
        form.addCommand(startCommand); // thêm start command
      } else if(command == startCommand) { // nếu đang bắt đầu

        player.start(); // play bắt đầu từ điểm dừng lúc pause
        form.removeCommand(startCommand);
        form.addCommand(pauseCommand);
      }
    } catch(Exception e) {
      System.err.println(e);
    }
  }
```

```
/* Tạo đối tượng Player và play media */
private void playMedia(String locator, String key) throws Exception {

  // xác định đường dẫn các tập tin cần play
  String file = locator.substring(
    locator.indexOf("file://") + 6,
    locator.length());

  // tạo player
  // tải player như là một tài nguyên và sử dụng thông tin của player
  // từ itemsInfo hashtable
  player = Manager.createPlayer(
      getClass().getResourceAsStream(file), (String)itemsInfo.get(key));

  // listener để quản lý các sự kiện của player như starting, closing, v.v...
  player.addPlayerListener(this);

  player.setLoopCount(-1); // ply repeat
  player.prefetch(); // tìm nạp trước
  player.realize(); // nhận dạng

  player.start(); // và play
}

/* Kiểm soát các sự kiện của Player */
public void playerUpdate(Player player, String event, Object eventData) {

  // nếu sự kiện player đã bắt đầu, hiển thị form
  // nhưng chỉ khi sự kiện dữ liệu hiển thị sự kiện liên quan đến player
  // đã bắt đầu, như khi sự kiện STARTED. Chú ý rằng eventData
  // hiển thị thời gian bắt đầu sự kiện
  if(event.equals(PlayerListener.STARTED) &&
    new Long(0L).equals((Long)eventData)) {

    display.setCurrent(form);
  } else if(event.equals(PlayerListener.CLOSED)) {

    form.deleteAll(); // hủy control trong form
  }
 }
}
```

1.4 Lớp Manager

Ví dụ sau hướng dẫn sử dụng lớp manager để lấy thông tin về các kiểu file mà giao thức mà lớp manager hỗ trợ

```
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.StringItem;
import javax.microedition.media.Manager;
import javax.microedition.midlet.MIDlet;

public class MediaInformationMIDlet extends MIDlet implements CommandListener {
  private Form mInformationForm;

  public void startApp() {
    if (mInformationForm == null) {
      mInformationForm = new Form("Loại file và giao thức");
      String[] contentTypes = Manager.getSupportedContentTypes(null);
      for (int i = 0; i < contentTypes.length; i++) {
        String[] protocols = Manager.getSupportedProtocols(contentTypes[i]);
        for (int j = 0; j < protocols.length; j++) {
          StringItem si = new StringItem(contentTypes[i] + ": ", protocols[j]);
          mInformationForm.append(si);
        }
      }
      Command exitCommand = new Command("Exit", Command.EXIT, 0);
      mInformationForm.addCommand(exitCommand);
      mInformationForm.setCommandListener(this);
    }

    Display.getDisplay(this).setCurrent(mInformationForm);
  }


    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

    public void commandAction(Command c, Displayable s) {
      notifyDestroyed();
    }
  }
```

1.5 Bài tập

1.5.1 Hiển thị hình ảnh

Viết chương trình hiển thị hình ảnh trên Form

Hướng dẫn:

-Tạo đối tượng image với đường dẫn là file .png (file ảnh được lưu cùng cấp với file mã nguồn)

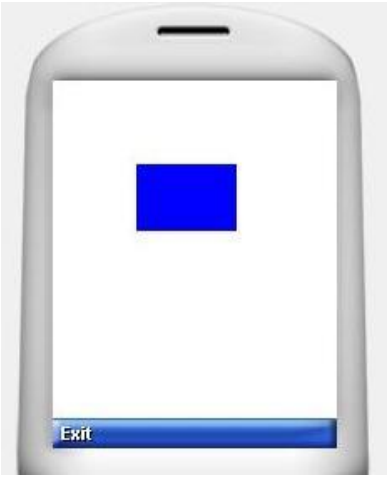
-Tạo đối tượng ImageItem từ image trên và chèn vào form

-Hiển thị ra màn hình

```
try {
    Image im = Image.createImage("/image_UIT.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");
}
```

1.5.2 Vẽ các đối tượng cơ bản

Viết chương trình vẽ hình chữ nhật lên màn hình



Hướng dẫn:

-Tạo lớp RectangleExample thừa kế từ Midlet. Trong đó có đối tượng canvas thuộc lớp MyCanvas (được định nghĩa bên dưới)

```
//  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);
    }


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

     public void pauseApp(){}
     public void destroyApp( boolean unconditional ){}
     public void exitMIDlet(){
         destroyApp(true);
         notifyDestroyed();
     }
}
```

-Tạo lớp MyCanvas thừa kế từ Canvas (lớp khung vẽ) và giao diện CommandListener (để xử lý nút Exit)

-Phương thức constructor của MyCanvas được tạo từ một đối tượng RectangleExample (để truyền vào chính đối tượng đã gọi MyCanvas)

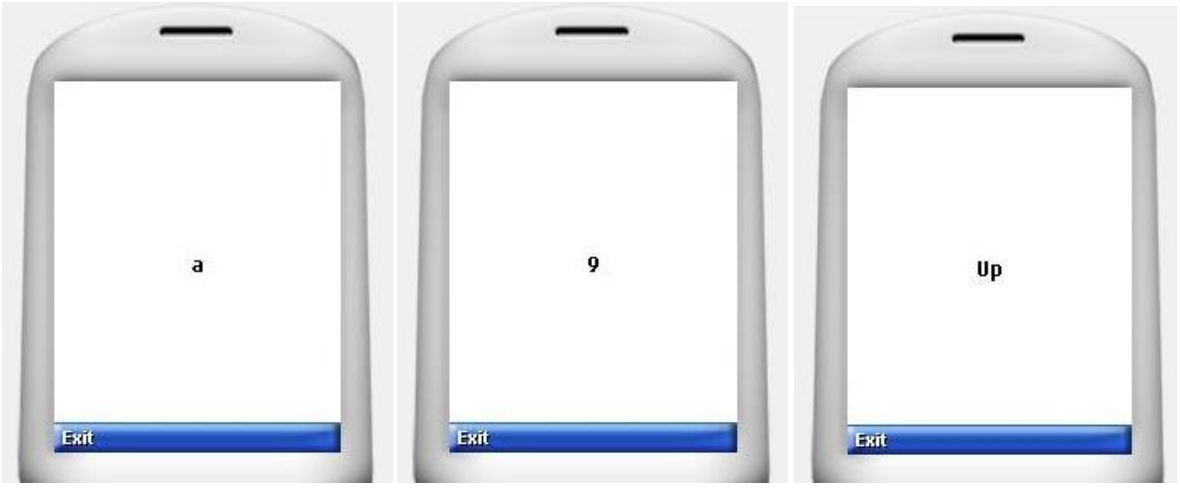
-Trong phương thức paint của canvas, dùng các hàm của lớp Graphics để vẽ và tô màu cho hình chữ nhật

```
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);
    }
```

```
    public void paint(Graphics graphics){
        graphics.setColor(255,255,255);
        graphics.fillRect(0, 0, getWidth(), getHeight());
        graphics.setColor(0, 0, 255);
        graphics.fillRect(50,50, 60, 40);
    }

    public void commandAction(Command command, Displayable s){
        if (command == exit){
            filledRectangleExample.exitMIDlet();
        }
    }
}
```

1.5.3 Xử lý sự kiện phím

Viết chương trình xử lý sự kiện phím, hiển thị tên của phím người dùng vừa nhấn



Hướng dẫn:

-Xây dựng lớp KeyCodes thừa kế từ Midlet, với giao diện là một Canvas (KeyCodeCamvas)

```
public class KeyCodes extends MIDlet{
    private Display  display;       // The display
    private KeyCodeCanvas canvas;   // Canvas

    public KeyCodes() {
        display = Display.getDisplay(this);
        canvas  = new KeyCodeCanvas(this);
    }
```

-Xây dựng lớp KeyCodeCanvas thừa kế từ Canvas và CommandListener với 3 thuộc tính: command Exit để thoát midlet, string keyText tên của phím được nhấn, KeyCodes midlet là đối tượng gọi canvas

-Phương thức constructor KeyCodeCanvas, khởi tạo từ 1 đối tượng KeyCodes:

```
/*-------------------------------------------------
* 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);
    }
```

-Trong hàm paint của lớp KeyCodeCanvas, thực hiện việc hiển thị keyText ra màn hình

```
/*-------------------------------------------------
* 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);
      }
  }
```

-Phương thức keyPressed của canvas xử lý sự kiện nhấn phím, dùng hàm getKeyName để lấy tên phím được nhấn. Thực hiện repaint để gọi phương thức paint

```
/*-------------------------------------------------
* Key code event handling
*-------------------------------------------------*/
    protected void keyPressed(int keyCode)  {
      keyText = getKeyName(keyCode);
      repaint();
    }
```