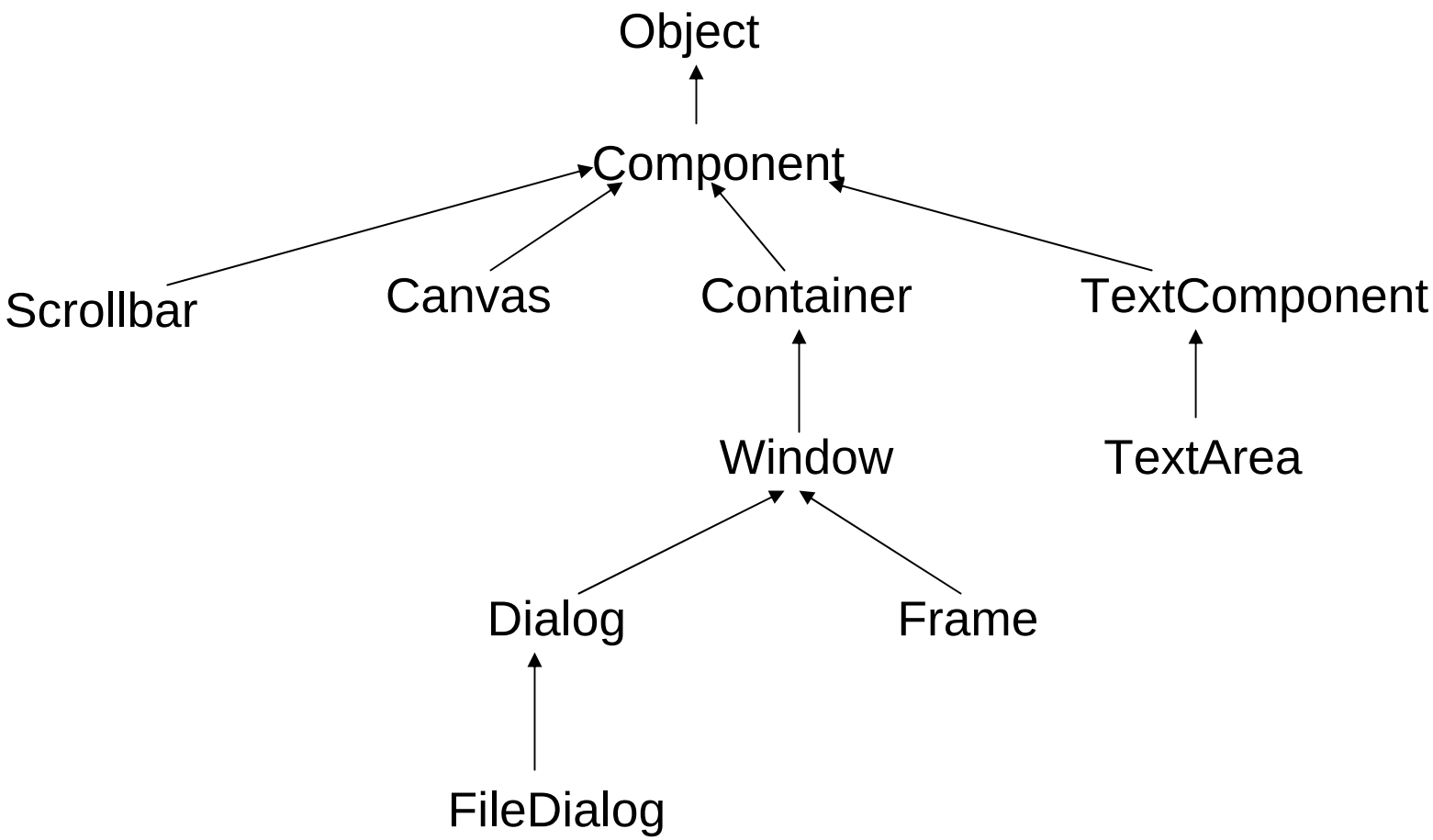
Lập trình Java cơ bản

Cao Đức Thông - Trần Minh Tuấn
cdthong@ifi.edu.vn, tmtuan@ifi.edu.vn

Bài 5. GUI nâng cao

- TextArea, Canvas
- CardLayout, GridBagLayout
- Frame, Menu, Dialog
- Scrollbar và ScrollPane
- Giới thiệu các thành phần Swing
- Phương pháp thiết kế MVC
- Bài tập

Các thành phần được giới thiệu



Vùng văn bản (TextArea)

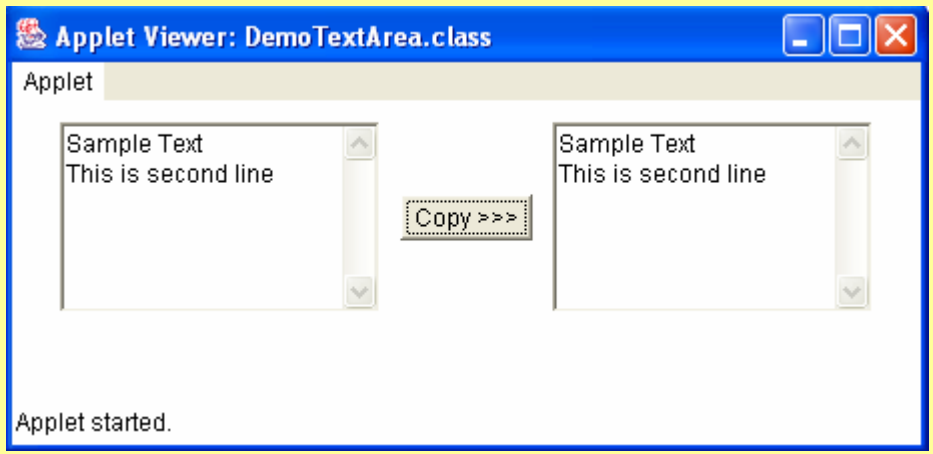
- Cho phép người dùng nhập vào nhiều dòng văn bản.
- Tạo đối tượng TextArea
  - TextArea();
  - TextArea(int rows, int columns);
  - TextArea(String s);
  - TextArea(String s, int rows, int columns);
- Các phương thức khác giống như TextField

Vùng văn bản (TextArea)

```
// Cac import can thiet...
public class DemoTextArea extends Applet implements ActionListener
{
    private TextArea textArea1, textArea2;
    private Button copy;

    public void init()
    {
        textArea1 = new TextArea("Sample Text", 5, 20);
        textArea2 = new TextArea(5, 20);
        copy = new Button("Copy >>>");
        setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));
        copy.addActionListener(this);
        add(textArea1);
        add(copy);
        add(textArea2);
    }
```

Vùng văn bản (TextArea)

```
    public void actionPerformed(ActionEvent event)
    {
        textArea2.setText(textArea1.getText());
    }
}
```



Khung vẽ (Canvas)

- Khung vẽ là một vùng chuyên để vẽ đồ hoạ, nó không bị che bởi các thành phần giao diện khác.
- Khung vẽ có thể xử lý các sự kiện giống như Applet.
- Để sử dụng khung vẽ, cần tạo một lớp khác dẫn xuất từ Canvas và cài đặt nạp chồng phương thức *paint()*.
- Nên gọi *setSize* cho khung vẽ. Toạ độ vẽ là (0,0) tính trong khung vẽ.

Khung vẽ (Canvas)

```
// Cac import can thiet...
public class DemoCanvas extends Applet implements ActionListener
{
    private Button rectButton;
    private Button circleButton;
    private MyCanvas canvas;

    public void init()
    {
        setLayout(new BorderLayout());
        rectButton = new Button("Draw Rectangle");
        circleButton = new Button("Draw Circle");
        rectButton.addActionListener(this);
        circleButton.addActionListener(this);
        Panel panel = new Panel();
        panel.add(rectButton);
        panel.add(circleButton);
```

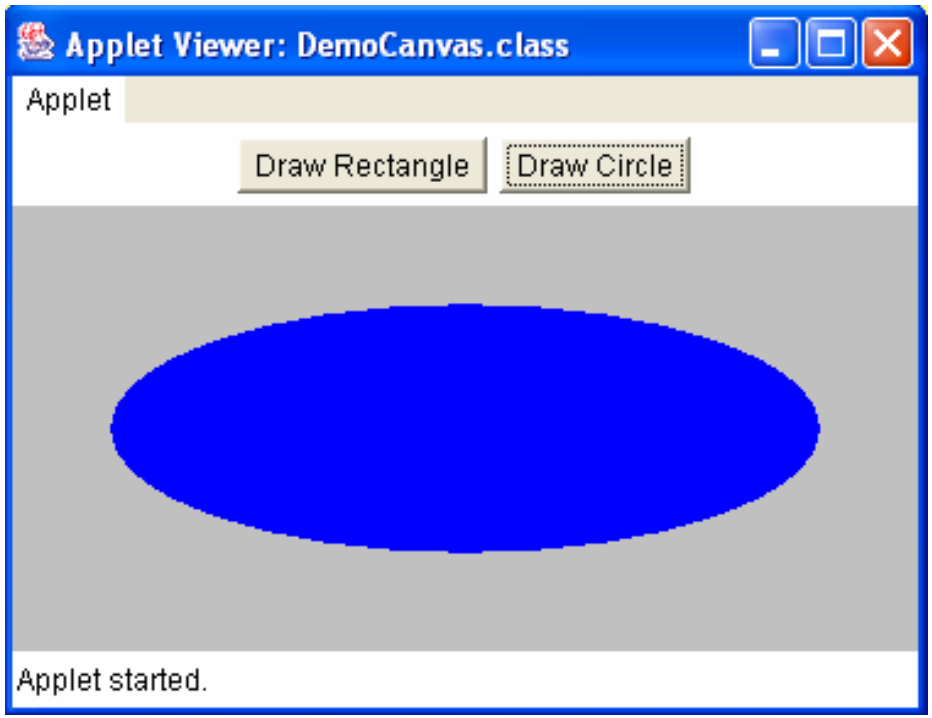
Khung vẽ (Canvas)

```
        canvas = new MyCanvas();
        canvas.setBackground(Color.lightGray);
        add(panel, BorderLayout.NORTH);
        add(canvas, BorderLayout.CENTER);
    }

    public void actionPerformed(ActionEvent event)
    {
        if (event.getSource() == rectButton)
            canvas.draw(1);
        else if (event.getSource() == circleButton)
            canvas.draw(2);
     }
}
```

Khung vẽ (Canvas)

```
class MyCanvas extends Canvas
{
    private int shape;
    public void paint(Graphics g)
    {
        Dimension size = getSize();
        g.setColor(Color.BLUE);
        if (shape == 1)
            g.fillRect(40, 40, size.width-80, size.height-80);
        else if (shape == 2)
            g.fillOval(40, 40, size.width-80, size.height-80);
    }

    public void draw(int shape)
    {
        this.shape = shape;
        repaint();
    }
}
```

Khung vẽ (Canvas)

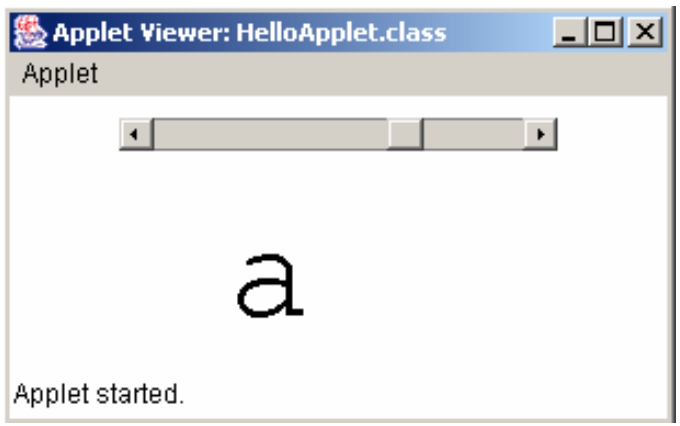


Thanh trượt (Scrollbar)

- Thanh trượt cho phép điều chỉnh giá trị trong một khoảng nhất định
- Để nghe sự kiện trên thanh trượt cần cài đặt giao tiếp *AdjusmentListener*.
  - Nạp chồng *adjustmentValueChanged()*

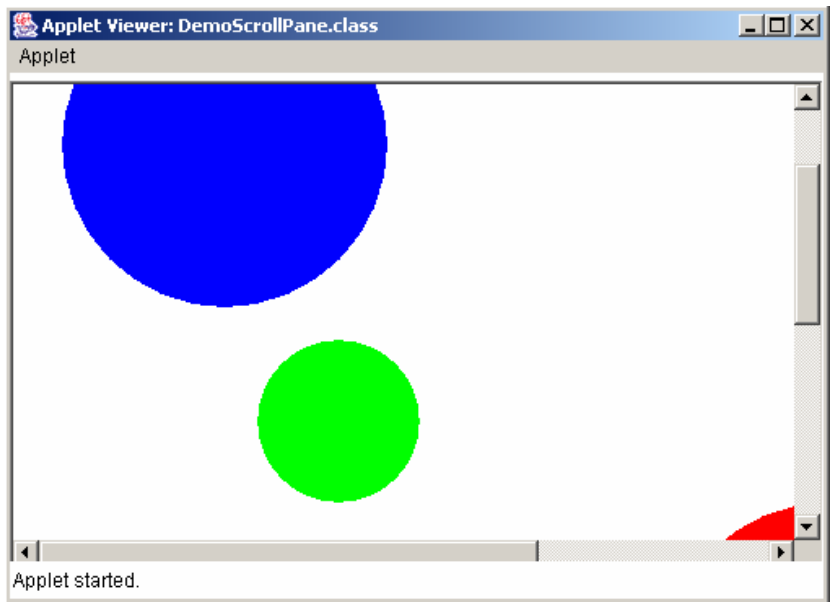
Thanh trượt (Scrollbar)

- Bài tập: Viết chương trình cho phép điều khiển font chữ thông qua thanh trượt. Khi thanh trượt thay đổi thì cỡ chữ hiển thị (drawString) thay đổi theo.



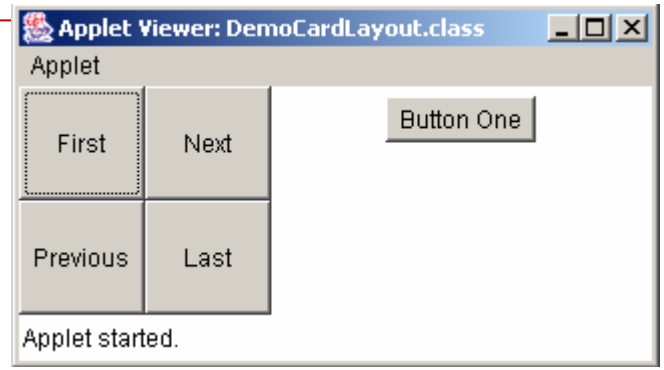
Khung cuộn (ScrollPane)

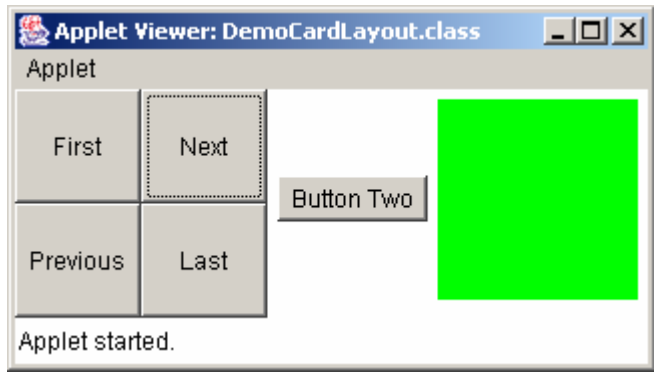
- Khung cuộn là một container cho phép chứa thành phần GUI có kích thước lớn hơn chính nó.
- Bài tập: Viết chương trình cho phép vẽ trong một canvas có độ rộng lớn hơn kích thước của applet. Đặt canvas vào trong một scroll pane.

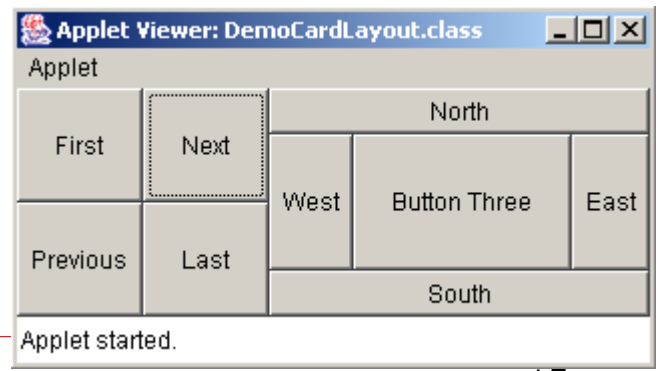


Bố cục nâng cao

- CardLayout
  - Sắp xếp các thành phần giống như các lá bài. Tại mỗi thời điểm chỉ lá bài đầu tiên được hiển thị.
  - Mỗi lá bài thường là một Panel và trên đó có thể dùng bất kỳ một bố cục nào.







Bố cục nâng cao

- GridBagLayout
  - Sắp xếp các thành phần trong một lưới giống như GridLayout.
  - Các thành phần có thể có kích thước khác nhau.
- Null Layout
  - Dùng lệnh *setLayout(null);*
  - Phải đặt vị trí và kích thước cho các thành phần thông qua các hàm: *setLocation, setSize, setBounds*.

Khung chứa Frame

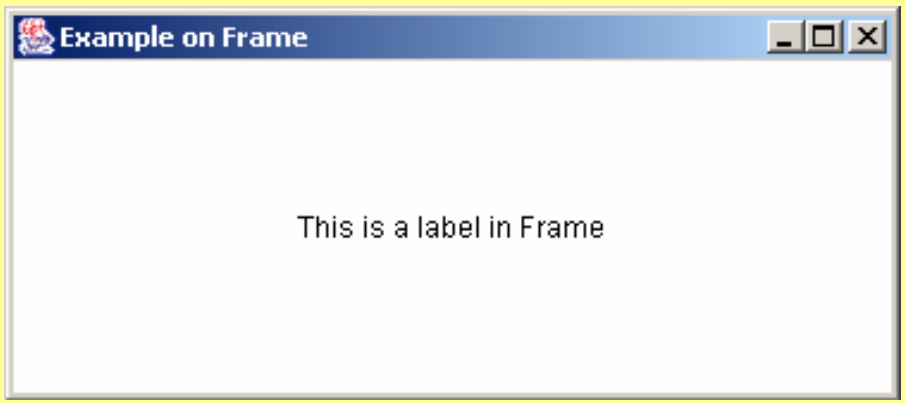
- Frame được dùng để xây dựng các ứng dụng GUI chạy độc lập.
- Frame là một cửa sổ có thanh tiêu đề và các đường biên. Bố cục mặc định của Frame là BorderLayout.
- Frame kế thừa từ Window, nó có thể nghe các sự kiện xảy ra trên cửa sổ khi cài đặt giao tiếp *WindowListener*.
- Các ứng dụng độc lập thường tạo ra cửa sổ kế thừa từ lớp Frame.

Ví dụ về Frame

```
import java.awt.*;
import java.awt.event.*;

public class DemoFrame
{
    public static void main(String[] args)
    {
        Frame frame = new Frame(“Example on Frame”);
        Label label = new Label("This is a label in Frame",
                                Label.CENTER);
        frame.add(label, BorderLayout.CENTER);
        frame.setSize(500,500);
        frame.setVisible(true);
        frame.addWindowListener(new MyWindowListener());
    }
}
```

Ví dụ về Frame

```
// Lop nghe doc lap (external listener)
class MyWindowListener extends WindowAdapter
{
    public void windowClosing(WindowEvent event)
    {
        System.exit(0);
    }
}
```

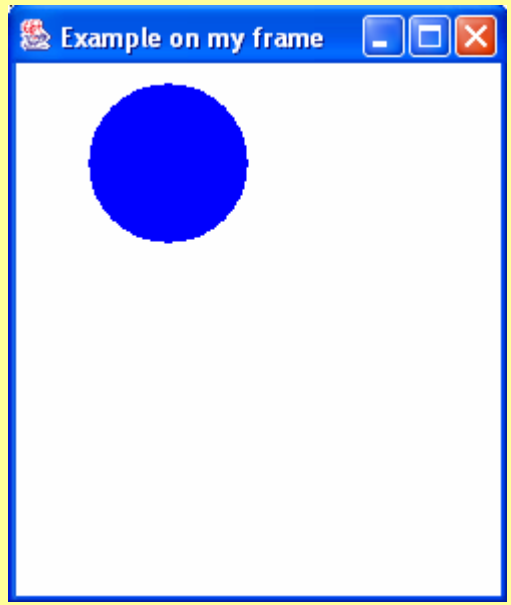


Ví dụ về Frame

```
import java.awt.*;
import java.awt.event.*;

public class DemoFrame2
{
    public static void main(String[] args)
    {
        MyFrame myFrame = new MyFrame("Example on my frame");
        myFrame.setSize(250, 300);
        myFrame.setVisible(true);
        myFrame.addWindowListener(new WindowAdapter()
            {// Lop nghe noi khong ten (anonymous inner class listener)
                public void windowClosing(WindowEvent e)
                {
                    System.exit(0);
                }
            });
    }
}
```

Ví dụ về Frame

```
class MyFrame extends Frame
{
    public MyFrame(String title)
    {
        super(title);
    }

    public void paint(Graphics g)
    {
        g.setColor(Color.BLUE);
        g.fillOval(40, 40, 80, 80);
    }
}
```



Cài đặt listener

- Lớp nghe độc lập: Lớp nghe sự kiện là một lớp khác với lớp chứa giao diện.
- Lớp nghe nội không tên: Lớp nghe không có tên, chỉ định nghĩa phương thức xử lý sự kiện (actionPerformed)
- Lớp nghe nội có tên: Lớp nghe được khai báo nằm trong một lớp khác.
  - Lớp nội có thể truy xuất các phương thức, dữ liệu của lớp chứa nó (outer class)
  - Dùng lớp nghe nội có tên là một kĩ thuật phổ biến.

Khung chứa Frame

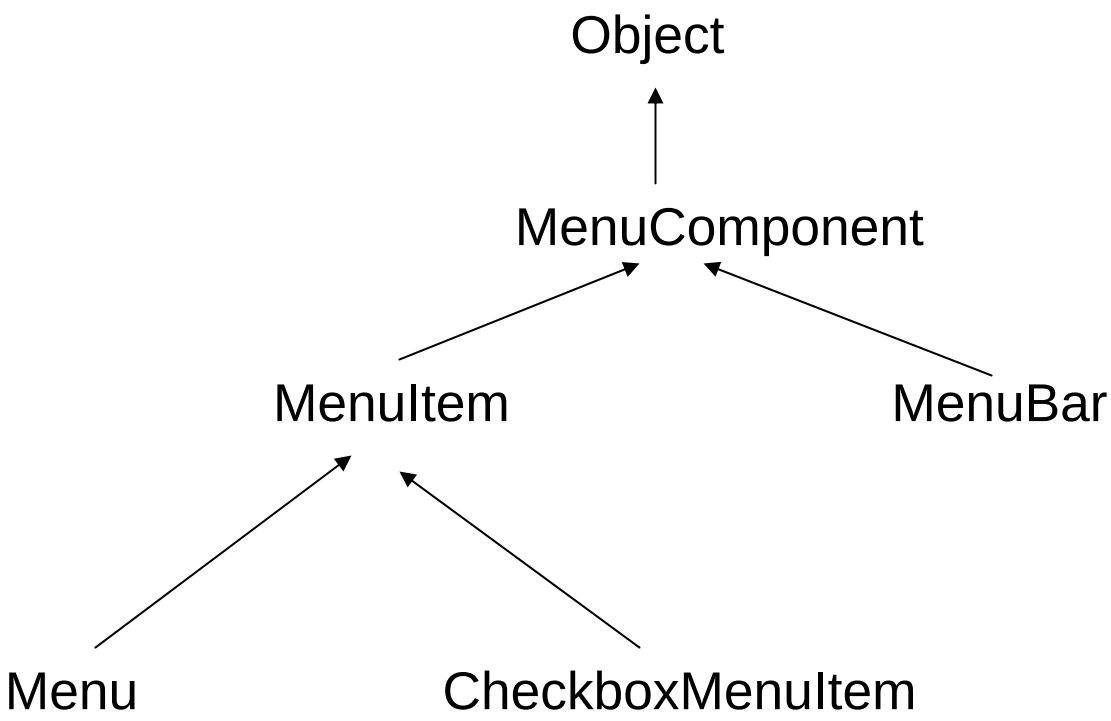
- Tạo các thành phần GUI và xử lý sự kiện trong Frame cũng giống như trong Applet.
- Chú ý:
  - Frame không có các phương thức init, start… như trong Applet.
  - Các ứng dụng độc lập dùng Frame phải có hàm main và được chạy trực tiếp bằng lệnh java
  - Cần có lệnh *setSize*, *setVisible(true)* để có thể hiển thị Frame.
  - Ở cuối chương trình nên có lệnh: System.exit(0);

Bài tập tại lớp

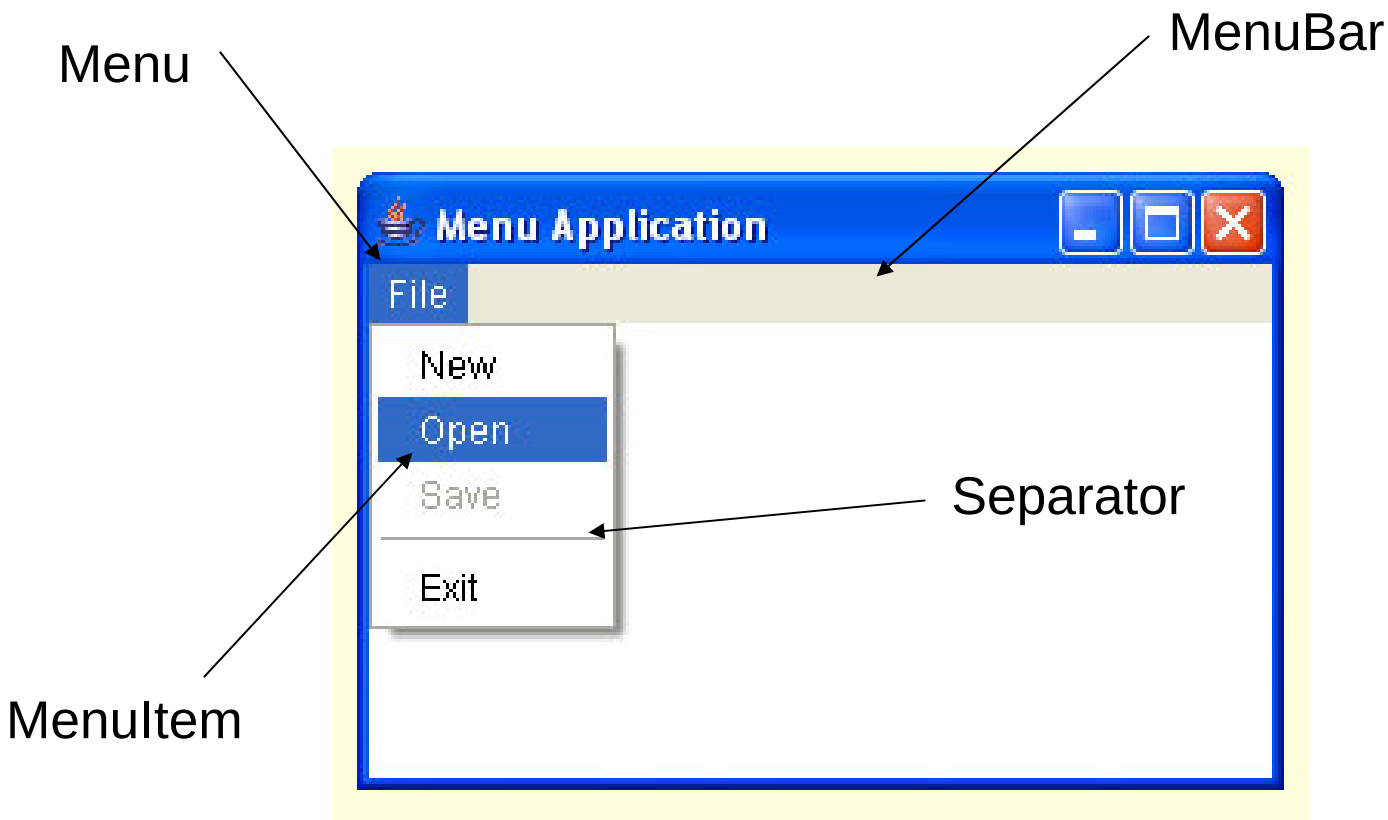
- Bài 1: Viết lại chương trình Tính Tổng 2 số sử dụng Frame.
- Bài 2: Mở rộng bài 1 để khi người dùng đóng cửa sổ thì sẽ xuất hiện một thông báo xác nhận việc đóng.

  Dùng *JOptionPane.showConfirmDialog*

Thực đơn (Menu)



Thực đơn (Menu)



Thực đơn (Menu)

- Tạo thực đơn
  - Tạo và gắn MenuBar vào cửa sổ
    - MenuBar menuBar = new MenuBar();
    - myFrame.setMenuBar(menuBar);
  - Tạo Menu và gắn vào MenuBar
    - Menu fileMenu = new Menu("File");
    - menuBar.add(fileMenu);
  - Tạo MenuItem và gắn vào Menu
    - MenuItem openItem = new MenuItem("Open");
    - fileMenu.add(openItem);
  - Tạo đường phân cách
    - fileMenu.addSeparator();

Thực đơn (Menu)

- Xử lý sự kiện trên các MenuItem
  - Đối tượng nghe các MenuItem phải cài đặt giao tiếp ActionListener
- *Tham khảo thêm về*
  - *CheckboxMenuItem*
  - *PopupMenu*

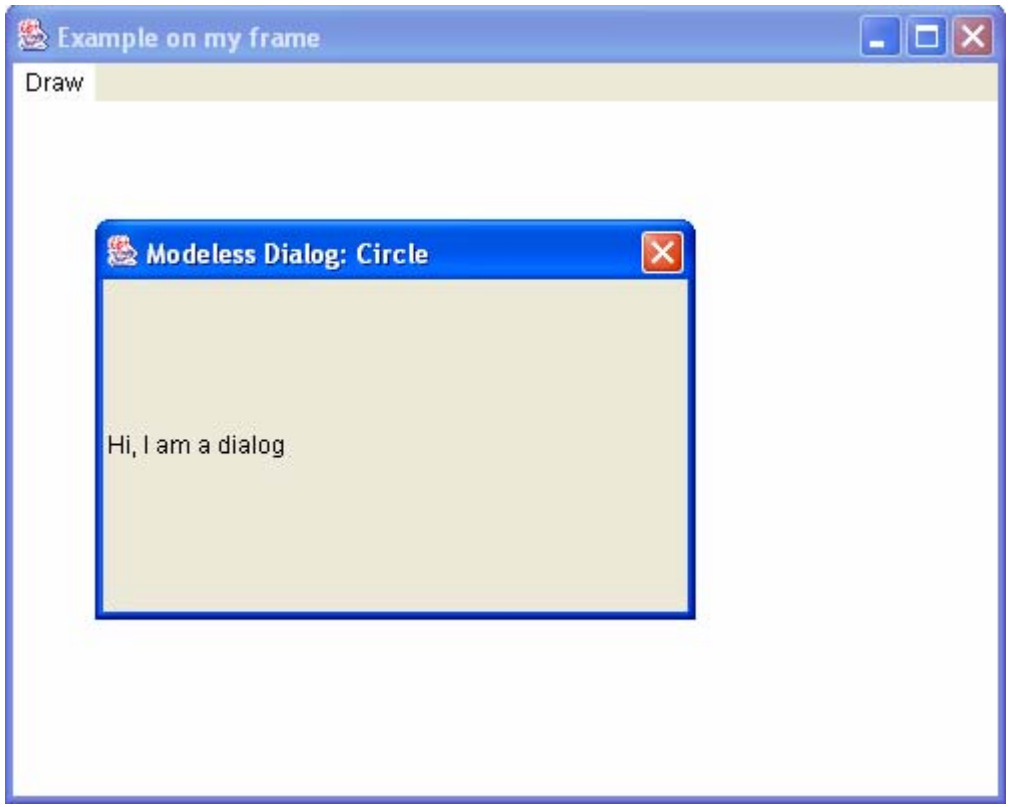
Hộp hội thoại (Dialog)

- Dialog cũng là một cửa sổ, thường dùng để nhập hoặc hiển thị thông tin với người dùng.
- Hai loại hộp thoại
  - Modal: Phải đóng hộp thoại trước khi chuyển sang cửa sổ khác.
  - Modaless: Có thể giữ nguyên hộp thoại và chuyển sang cửa sổ khác.

Hộp hội thoại (Dialog)

- Dialog kế thừa từ lớp Window, nó có bố cục mặc định là BorderLayout.
- Hộp thoại có thể chứa các thành phần GUI và xử lý các sự kiện như một cửa sổ bình thường.

Ví dụ về Frame, Menu và Dialog



Ví dụ về Frame, Menu và Dialog

```
import java.awt.*;
import java.awt.event.*;

public class DemoFrame3
{
    public static void main(String[] args
    {
        MyFrame myFrame = new MyFrame("Example on my frame");
        myFrame.setSize(500, 400);
        myFrame.setVisible(true);
        myFrame.addWindowListener(new WindowAdapter()
          {
              public void windowClosing(WindowEvent e)
              {
                  System.exit(0);
              }
          });
     }
}
```

Ví dụ về Frame, Menu và Dialog

```
class MyFrame extends Frame implements ActionListener
{
    private MenuBar menuBar;
    private Menu menu;
    private MenuItem circleItem, rectItem;

    public MyFrame(String title)
    {
        super(title);
        menuBar = new MenuBar();     setMenuBar(menuBar);
        menu = new Menu("Draw");     menuBar.add(menu);

        circleItem = new MenuItem("Circle");
        rectItem = new MenuItem("Rectangle");
        menu.add(circleItem);
        menu.add(rectItem);
        circleItem.addActionListener(this);
        rectItem.addActionListener(this);
    }
```

Ví dụ về Frame, Menu và Dialog

```
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == circleItem)
        {   MyDialog dialog = new MyDialog(this,
                                    "Modeless Dialog: Circle", false);
        }
    }
}

class MyDialog extends Dialog
{
    MyDialog(Frame parent, String title, boolean isModel)
    {
        super(parent, title, isModel);
        add(new Label("Hi, I am a dialog"), BorderLayout.CENTER);
        setSize(300, 200);
        setVisible(true);
        addWindowListener(new MyDialogListener(this));
     }
};
```

Ví dụ về Frame, Menu và Dialog

```
// Co the dat lop nay lam lop noi (inner class) cua lop MyDialog
class MyDialogListener extends WindowAdapter
{
    Dialog dialog;

    MyDialogListener(Dialog dia)
    {
        dialog = dia;
    }

    public void windowClosing(WindowEvent e)
    {
        dialog.setVisible(false);
        dialog.dispose();
    }
}
```

Giới thiệu JFC

- JFC (Java Foundation Class)
  - Là thư viện lập trình giao diện đồ hoạ phát triển dựa trên thư viện AWT
  - JFC cung cấp khả năng tạo giao diện linh động, uyển chuyển hơn so với AWT
  - JFC có sẵn trong các phiên bản từ Jdk 1.2 trở đi.
  - Các lớp của JFC nằm trong gói javax.swing

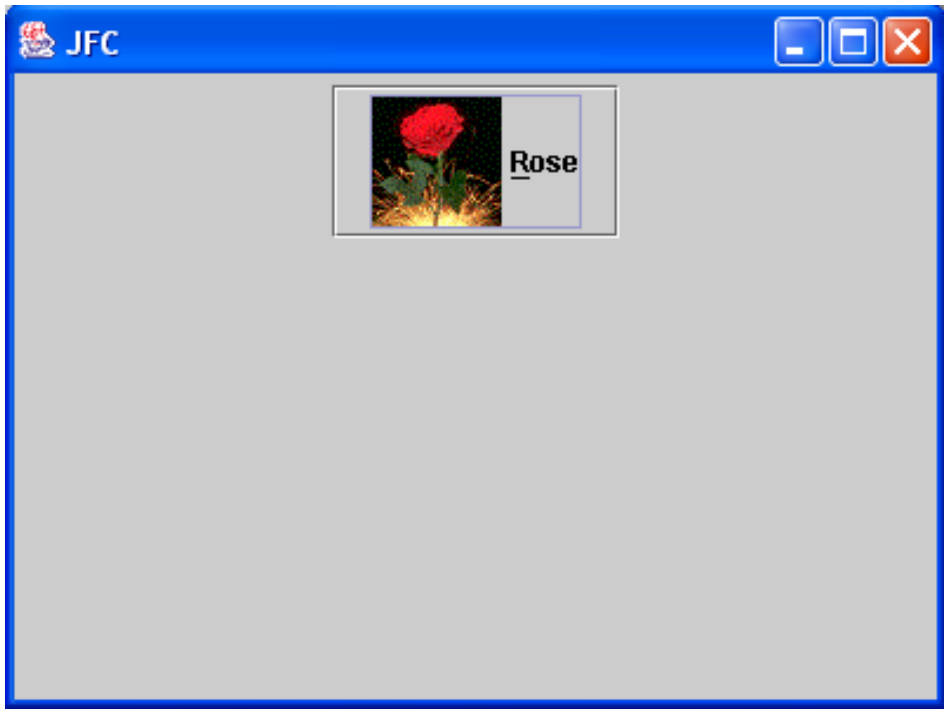
Các thành phần Swing

- Các thành phần GUI của Swing thường bắt đầu bởi chữ J:
  - JButton, JLabel, JTextArea, JFrame, JPanel, JCheckBox, JRadioButton, JList, JComboBox, JScrollPane…
  - Các thành phần mở rộng như: JTabbedPane, JProgressBar, JTable, JTree
- Việc xử lý sự kiện trên các thành phần Swing giống như trên các thành phần AWT.

Ví dụ về Swing

```
import java.awt.*;
import javax.swing.*;

public class HelloJFC
{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("JFC");
        Icon icon = new ImageIcon("rose.gif");
        JButton button = new JButton("Rose", icon);
        button.setMnemonic('R');
        button.setToolTipText("Button Rose");

        frame.getContentPane().setLayout(new FlowLayout());
        frame.getContentPane().add(button);
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.setSize(400, 300);
        frame.setVisible(true);
    }
}
```

Ví dụ về Swing

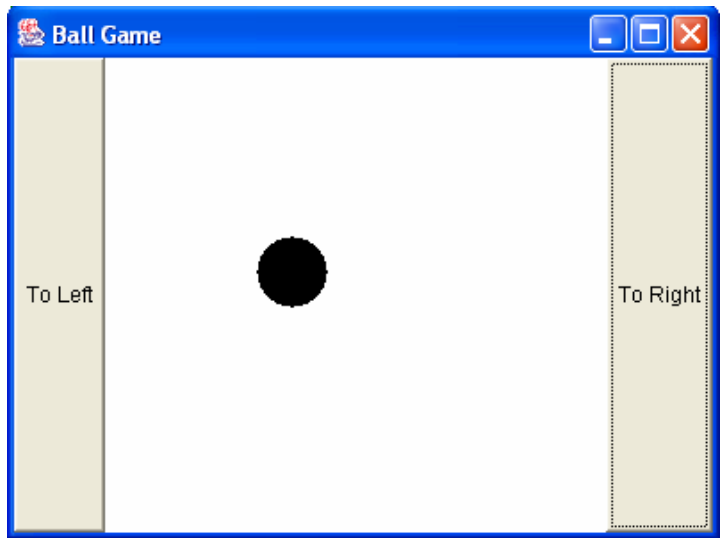


Thiết kế chương trình

- Các thành phần của chương trình
  - Dữ liệu của bài toán cần xử lý (Model)
  - Hiển thị dữ liệu của bài toán thông qua giao diện (View)
  - Điều khiển tương tác với người dùng (Controller)
- Ví dụ: Chương trình điều khiển quả bóng
  - Model: Dữ liệu về quả bóng
  - View: Giao diện hiển thị dữ liệu quả bóng
  - Controller: Điều khiển di chuyển quả bóng

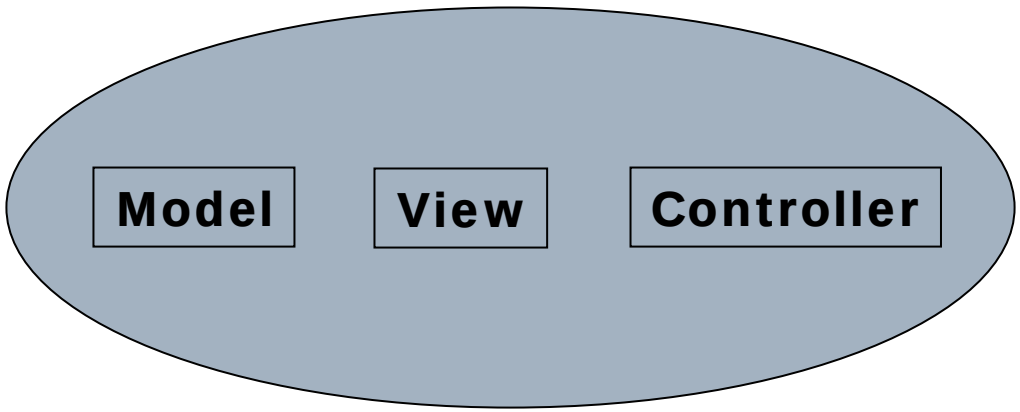
Thiết kế chương trình

- Model: Dữ liệu về bóng gồm toạ độ tâm (x,y) và bán kính bóng.
- View: Giao diện hiển thị gồm có hình quả bóng và 2 nút điều khiển.
- Controller: Khi ấn nút điều khiển thì quả bóng di chuyển.



Một số phương pháp thiết kế

- Big Blob
  - Tất cả Model, View, Controller đặt trong một một lớp duy nhất.



Ví dụ với Big Blob

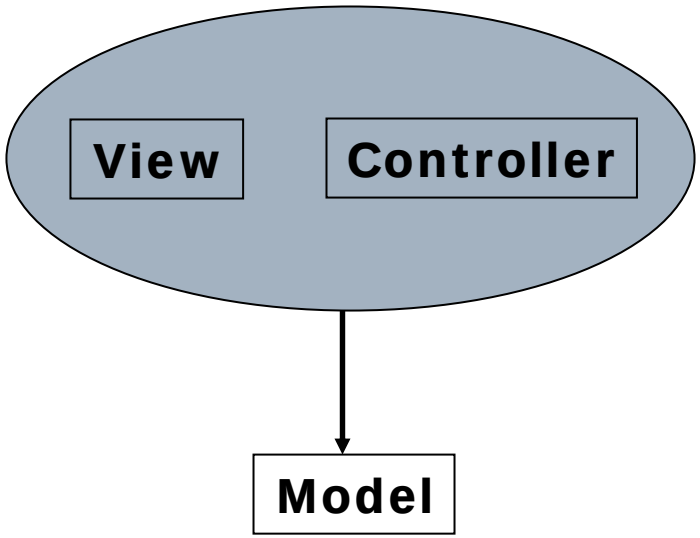
```
// file TestBall.java tao ra mot big blob
public class TestBall
{
    public static void main(String[] args)
    {
        MyBallFrame myFrame = new MyBallFrame("Ball Frame");
        myFrame.setSize(400, 300);
        myFrame.setVisible(true);
        ...
    }
}
```

Ví dụ với Big Blob

```
// MyBallFrame la mot big blob
// No chua ca model, view va controller
class MyBallFrame extends Frame implements ActionListener
{
    private int x, y, radius; // du lieu ve qua bong (model)
    private Button moveLeft, moveRight; // thanh phan GUI (view)

    ...
    moveLeft.addActionListener(this);
    moveRight.addActionListener(this);
    ...

    // xu ly su kien (controller)
    public void actionPerformed(ActionEvent event)
    ...
}
```

Một số phương pháp thiết kế

- Presentation-Model
  - Tách riêng Model và Presentation (gồm View + Controller)



Ví dụ với Presentation-Model

```
// file TestBall.java tao model va presentation
public class TestBall
{
    public static void main(String[] args)
    {
        // tao model
        BallModel myBall = new BallModel(50, 50, 20);

        // tao presentation
        BallPresentation myFrame = new BallPresentation(myBall);
        ...
    }
}
```

Ví dụ với Presentation-Model

```
// file BallPresentation.java chua view va controller
// No co mot thanh phan du lieu la model can xu ly
// Cach 1: Dung top-level listener
public class BallPresentation extends Frame implements ActionListener
{
    private BallModel ball;  // model can xu ly
    private Button moveLeft, moveRight;  // thanh phan GUI (view)
    ...
    moveLeft.addActionListener(this);
    moveRight.addActionListener(this);
    ...

    // xu ly su kien (controller)
    public void actionPerformed(ActionEvent event)
    ...
}
```

Ví dụ với Presentation-Model

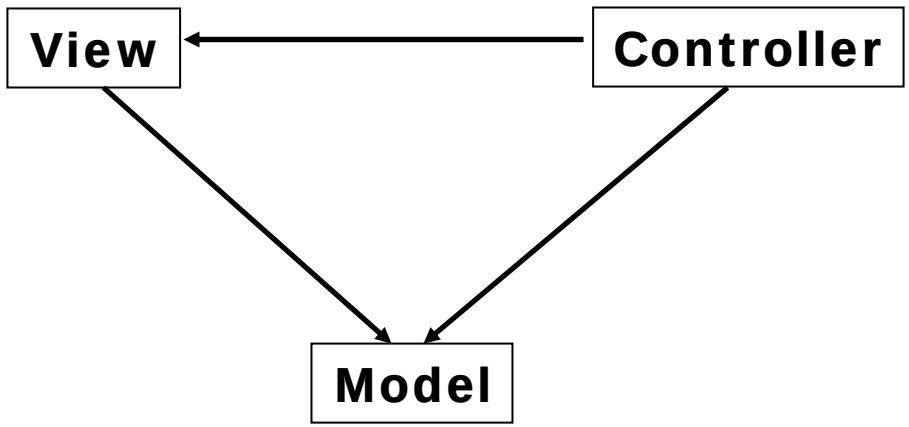
```
// file BallPresentation.java, cach 2: dung lop nghe la inner class
public class BallPresentation extends Frame
{
    private BallModel ball;  // model can xu ly
    private Button moveLeft, moveRight;  // thanh phan GUI (view)
    ...
    moveLeft.addActionListener(new ToLeftListener());
    moveRight.addActionListener(new ToRightListener());
    ...
    // xu ly su kien (controller)
    class ToLeftListener implements ActionListener
    {
        public void actionPerformed(ActionEvent event)
        {
            ball.moveLeft();
            repaint(); // goi phuong thuc cua lop outer
        }
    }
    ...
}
```

Ví dụ với Presentation-Model

```
// file BallModel.java chua du lieu va phuong thuc cua qua bong
// Model khong phu thuoc vao Presentation
public class BallModel
{
    private int x, y, radius;

    BallModel(int x, int y, int radius) ...
    void moveLeft() ...
    void moveRight() ...
    int getXCenter() ...
    int getYCenter() ...
    int getRadius() ...
}
```

Một số phương pháp thiết kế

- Model-View-Controller
  - Tách riêng Model, View và Controller



Model-View-Controller

- Ưu điểm
  - Các modul độc lập, dễ quản lý
  - Có thể dễ dàng tạo nhiều giao diện khác nhau cho cùng một chương trình
  - Dễ mở rộng chương trình

Ví dụ với MVC

```
// file TestBall.java tao model, view va controller
public class TestBall
{
    public static void main(String[] args)
    {
        // tao model
        BallModel myBall = new BallModel(50, 50, 20);

        // tao view
        BallView ballView = new BallView(myBall);

        // tao controller
        BallController ballController =
                                    new BallController(myBall, ballView);

        ballView.setVisible(true);
        ...
    }
}
```

Ví dụ với MVC

```
// file BallView.java
public class BallView extends Frame
{
    private BallModel ball;  // model can xu ly
    private Button moveLeft, moveRight;

    BallView(BallModel ballModel)
    {
        ball = ballModel;
    }

    public void paint(Graphics g)
    {
        g.fillOval(...);
    }
    // phuong thuc nay duoc goi boi controller
    public void addToLeftListener(ActionListener al)
    {
        buttLeft.addActionListener(al);
    }
    ...
}
```

Ví dụ với MVC

```
// file BallController.java
public class BallController
{
    private BallModel ball;  // model can xu ly
    private BallView view;  // view can xu ly
    ...
    BallController(BallModel ballModel, BallView ballView)
    {
        // nhan model va view can xu ly
        ball = ballModel;
        view = ballView;

        // dat lang nghe tren view
        view.addToLeftListener(new ToLeftListener());
        view.addToRightListener(new ToRightListener());
    }
```

Ví dụ với MVC

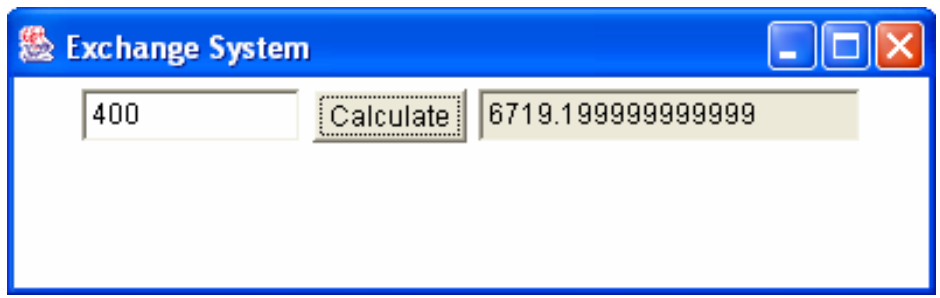
```
// file BallController.java (tiep theo)

    // xu ly su kien thong qua inner class
    class ToLeftListener implements ActionListener
    {
        public void actionPerformed(ActionEvent event)
        {
            ball.moveLeft();
            view.repaint();
        }
    }
    ...
}
```

Ví dụ với MVC

```
// file BallModel.java chua du lieu va phuong thuc cua qua bong
// Model khong phu thuoc vao View va Controller
public class BallModel
{
    private int x, y, radius;

    BallModel(int x, int y, int radius) ...
    void moveLeft() ...
    void moveRight() ...
    int getXCenter() ...
    int getYCenter() ...
    int getRadius() ...
}
```

Case study: Chương trình đổi tiền

- Viết chương trình cho phép tính số lượng tiền VNĐ tương ứng với một số lượng USD cho trước. Biết rằng 1 đôla = 16,798 đ.
- Mô tả giao diện như sau:



Case study: Chương trình đổi tiền

- ExchangeModel ?
- ExchangeView ?
- ExchangeController ?
- Cài đặt và thử nghiệm

Tài liệu tham khảo

- http://www.dickbaldwin.com/toc.htm
- http://leepoint.net/notes-java/index.html
- http://java.sun.com/developer/onlineTraining/GUI/Swing2/shortcourse.htm

Bài tập

1. Viết chương trình cho phép người dùng chọn một trong hai chế độ là Line và Point (dùng Checkbox), sau đó người dùng có thể dùng chuột để vẽ trong một Canvas nằm giữa màn hình (giống MS Paint).
2. Viết lại các chương trình liên quan tới Graphics sử dụng Frame.
3. Viết lại các chương trình liên quan tới thành phần giao diện GUI sử dụng Frame.

Bài tập

4. Viết chương trình tạo 3 menu trong một Frame như sau:

Colors	**Shapes**	**Help**
Red	Circle	About
Green	Rect	
Blue	Line	

Menu **Colors** cho phép chọn màu, menu **Shapes** cho phép chọn hình. Khi người dùng chọn menu và ấn nút *draw* trên Frame thì chương trình vẽ ra hình và màu được chọn. Menu **Help** – About hiển thị hộp thoại giới thiệu về chương trình.

Bài tập

5. Viết chương trình tính diện tích các hình: Tạo một Frame trong đó ở bên trái có 3 lựa chọn là Circle, Rectangle và Triangular (dùng Checkbox). Khi người dùng chọn một trong các hình thì ở bên phải sẽ hiển thị các ô nhập liệu tương ứng: Circle có một ô nhập là Radius, Rectangle có 2 ô nhập là Width và Height, Triangular có 3 ô nhập là 3 cạnh a, b, c. Sau đó người dùng chọn nút Compute thì chương trình tính và hiển thị kết quả lên màn hình. (Dùng CardLayout)

Bài tập

6. Viết chương trình tạo 2 menu item là *Nhập hàng* và *Bán hàng*. Khi người dùng chọn nhập hàng thì hiển thị Frame cho phép nhập vào tên hàng, số lượng, đơn giá (lưu thông tin này vào mảng). Khi người dùng chọn *Bán hàng* thì hiển thị Frame (hoặc Dialog) cho phép nhập vào tên người mua và cho phép chọn mua một trong số các mặt hàng có sẵn; sau đó hiển thị số tiền mà người đó phải trả. (Thiết kế theo MVC)