



**FPT POLYTECHNIC**

# LẬP TRÌNH JAVA

## **Bài 7: Applets**

---

[www.poly.edu.vn](http://www.poly.edu.vn)

---



**Điểm danh**

## Nhắc lại bài trước

- Khái niệm Genegics
- Ưu điểm Genegics
- Tạo class generic và method
- Giới hạn kiểu dữ liệu
- Các ký hiệu đại diện
- Generic method
- Generic Interface
- Một số hạn chế



## Nội dung bài học

- Khái niệm Applets
- Sự khác nhau giữa Applets và Applications
- Vòng đời của applet
- Một số phương thức của class Graphics
- Tạo một applet
- Sử dụng tham số trong Applets
- Xử lý sự kiện
- Một số interface và class xử lý sự kiện
- Một số ví dụ



# Khái niệm Applets

- Applet là một chương trình Java được nhúng vào trang html và được chạy trên trình duyệt web.
- Tất cả các applet đều là các subclass của `java.applet.Applet`.



# Sự khác nhau giữa Applets và Applications

## Applets

- Applet là một ứng dụng nhỏ được dùng chủ yếu cho việc hiển thị trên web.
- Các Applet phải kế thừa từ class `java.applet.Applet`
- Applets được chạy trên trình duyệt web.

## Applications

- Application - Một ứng dụng được thiết kế để hoạt động như một chương trình độc lập.
- Không có hạn chế như vậy cho một ứng dụng
- Các application sẽ được chạy bởi Java virtual machine.

# Sự khác nhau giữa Applets và Applications

## Applets

- Phương thức `init()` sẽ được thực hiện đầu tiên trong Applet.
- Không cần thiết phải có phương thức `main()` trong một applet.

## Applications

- Phương thức `main()` sẽ được thực hiện đầu tiên trong application.
- Trong một application, bắt buộc phải có phương thức `main()` trong một class là `public`.

# Sự khác nhau giữa Applets và Applications

## Applets

- Sử dụng các phương thức trong package `java.awt` để hiển thị kết quả trong cửa sổ AWT.
- Có một số việc chỉ có thể làm ở application thông thường nhưng không thể làm trong một applet.

## Applications

- Để hiển thị kết quả, sử dụng phương thức `System.out.println()` của package `java.lang` (tự động import)
- Tất cả những công việc làm trong applet đều có thể làm được trong một application.



# Vòng đời của applet

- Vòng đời của một đối tượng được tính từ khi nó được khởi tạo đến khi nó bị hủy bỏ.
- Một applet định nghĩa cấu trúc của nó từ bốn sự kiện xảy ra trong suốt quá trình thực thi.
- Đối với mỗi sự kiện, một phương thức được tự động được gọi.

# Vòng đời của applet

*Các phương thức đó là:*

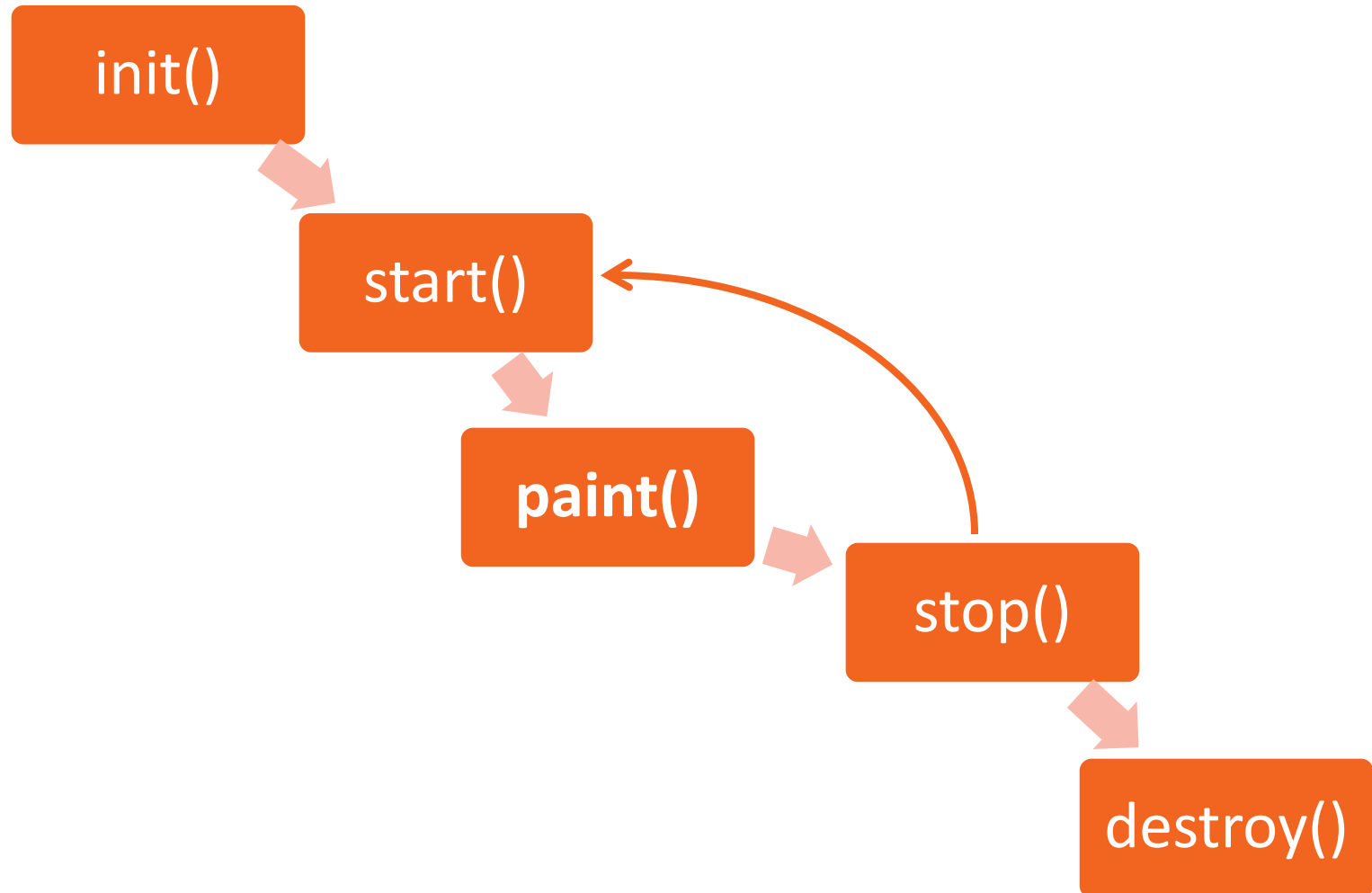
- **init()** : Khởi tạo applet.
- **start()** : Bắt đầu applet sau khi khởi tạo.
- **stop()** : Dừng applet đang thực thi.
- **destroy()** : Hủy bỏ applet.
- **paint()** : Hiển thị dòng văn bản, tô vẽ các hình.

# Vòng đời của applet

*Các phương thức đó là:*

- **update (Graphics):** Khi gọi phương thức `repaint()`, phương thức `update(Graphics g)` sẽ tự động được gọi.
- **showStatus(String):** Hiển thị dòng trạng thái ở cuối cửa sổ applet.
- **public String getParameter(String):** Trả về giá trị của tham số String.

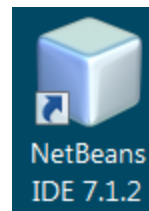
# Vòng đời của applet



# Vòng đời của applet

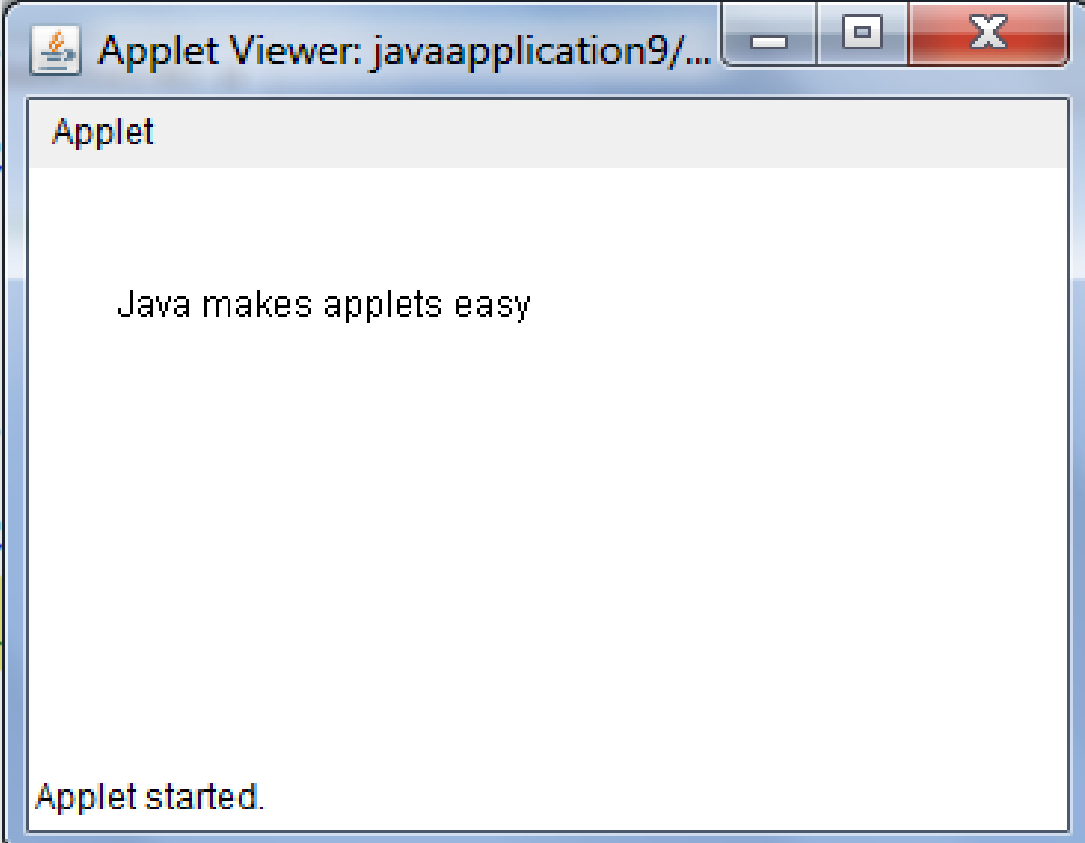
*Để chạy một applet thì:*

- Dùng file javac.exe dịch ra file .class
- Để chạy file .class thì có 2 cách:
  - Appletviewer
  - Trình duyệt web
- IDE Netbeans đã hỗ trợ việc dịch và chạy file applet.



# Tạo một applet

```
15 public class NewApplet extends Applet {  
16     private String str;  
17     @Override  
18     public void paint(Graphics g) {  
19         g.drawString(str, 100, 100);  
20     }  
21     @Override  
22     public void start() {  
23         str = "Java makes applets easy";  
24     }  
25 }
```

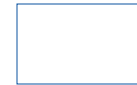


The image shows a code editor with a Java class named `NewApplet` that extends `Applet`. The class has a private `String` variable `str`. It overrides the `paint` method to draw the string `str` at coordinates (100, 100) and the `start` method to initialize `str` with the text "Java makes applets easy". A preview window titled "Applet Viewer: javaapplication9/..." is overlaid on the code, showing the rendered output of the applet, which is the text "Java makes applets easy". The status bar at the bottom of the preview window indicates "Applet started."

# Một số phương thức của class Graphics

**g.drawString**("Hello", 20, 20);      Hello

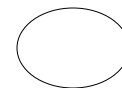
**g.drawRect**(x, y, width, height);



**g.fillRect**(x, y, width, height);



**g.drawOval**(x, y, width, height);



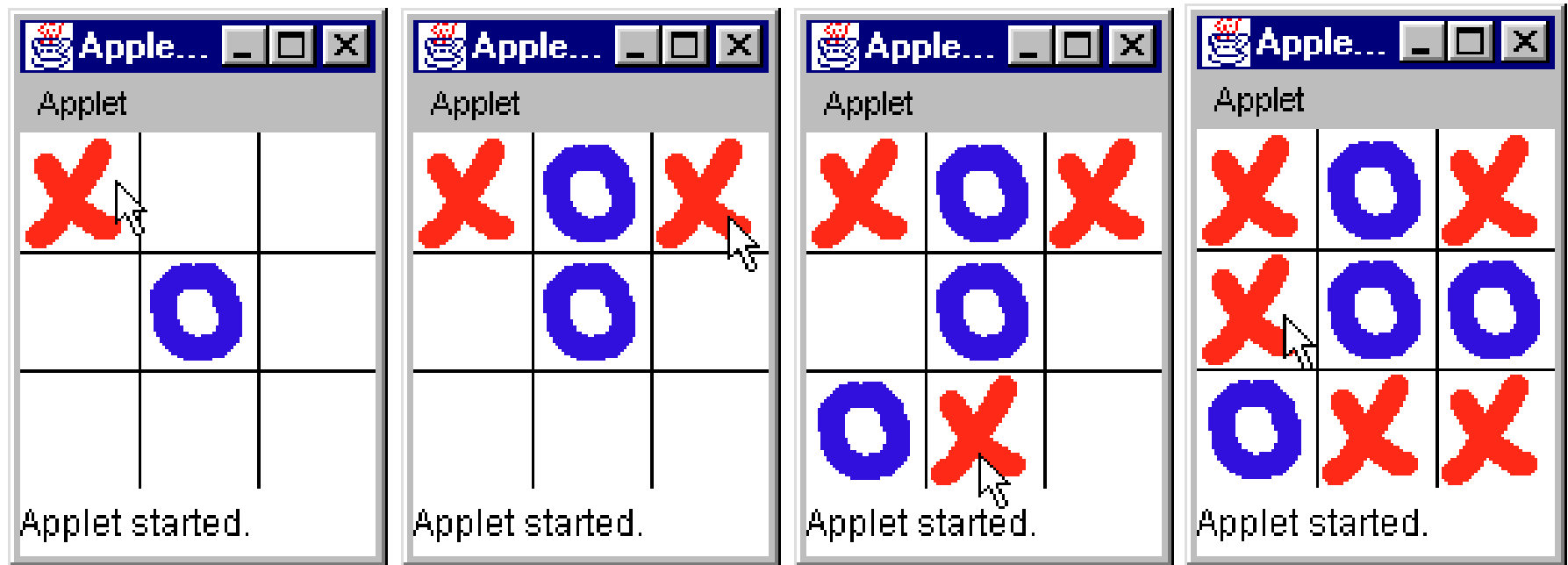
**g.fillOval**(x, y, width, height);



**g.setColor**(Color.red);

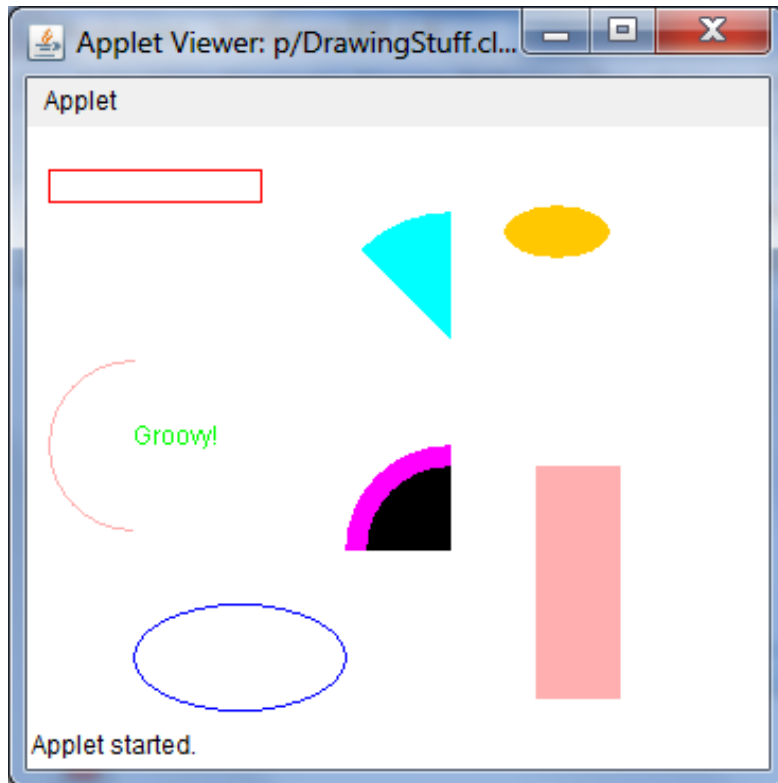


# Một số ví dụ Applet

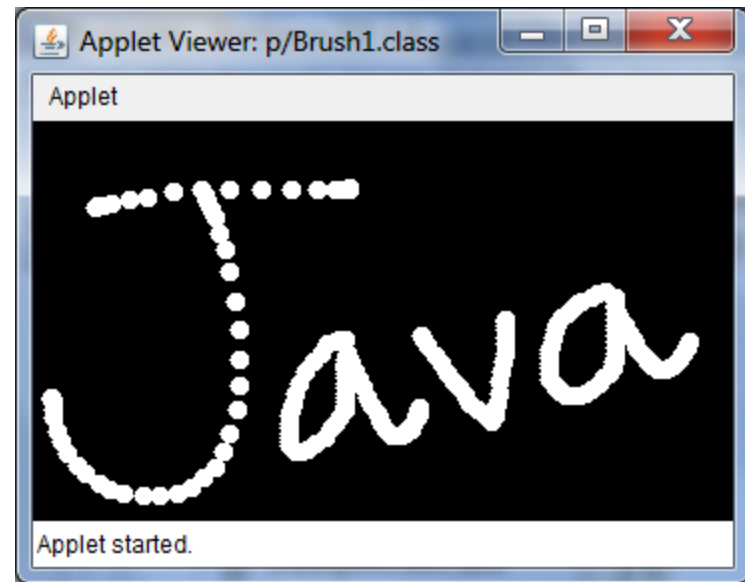
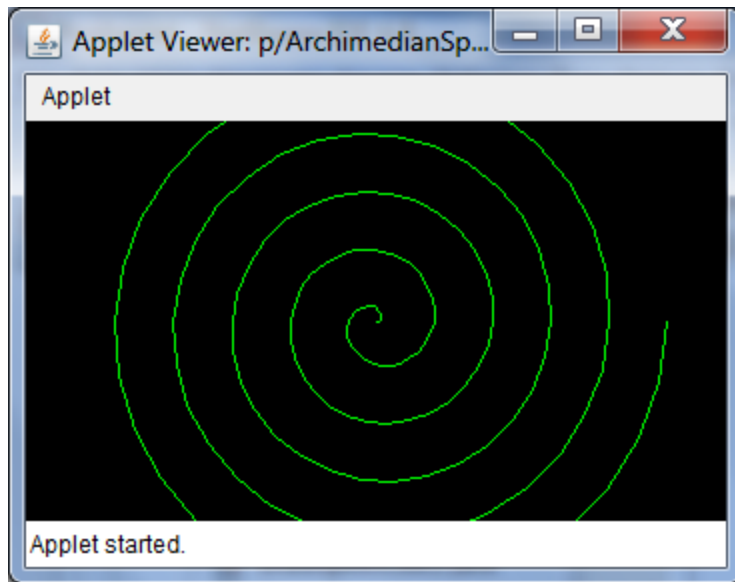




# Một số ví dụ Applet



# Một số ví dụ Applet



# Sử dụng tham số trong Applets

## FontExample.java

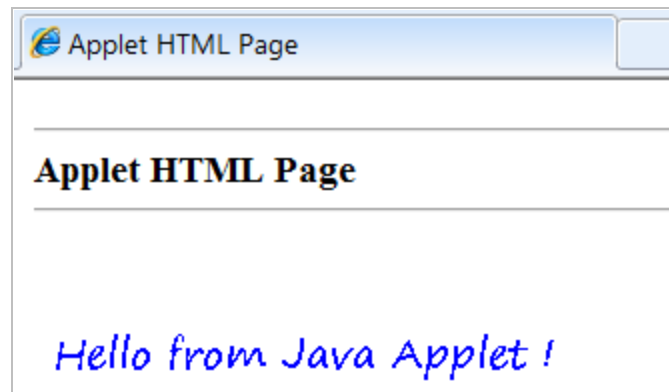
```
public class FontExample extends Applet {  
    public void paint(Graphics g) {  
  
        String myFont = getParameter("font");  
        String myString = getParameter("string");  
        int mySize = Integer.parseInt(getParameter("size"));  
  
        Font f = new Font(myFont, Font.PLAIN, mySize);  
        g.setFont(f);  
        g.setColor(Color.BLUE);  
        g.drawString(myString, 10, 50); //cột, hàng  
  
    }  
}
```

# Sử dụng tham số trong Applets

## FontExample.html

```
<P>  
  <APPLET codebase="classes" code="p/FontExample.class"  
    width=350 height=200>  
    <param name="font" value="Segoe Print">  
    <param name="string" value="Hello from Java Applet !">  
    <param name="size" value="20">  
  </APPLET>  
</P>
```

Kết quả



# Xử lý sự kiện

Có 3 thành phần chính trong xử lý sự kiện:

- **Events** (sự kiện): Một sự kiện là một sự thay đổi trạng thái của một đối tượng.
- **Events Source**: Là một đối tượng tạo ra sự kiện.
- **Listeners**: Là một đối tượng lắng nghe xem có sự kiện nào xảy ra hay không. Một listener sẽ nhận được thông báo khi một sự kiện được kích hoạt.

# Xử lý sự kiện

- Một sự kiện sẽ được tự động tạo ra khi có sự tương tác giữa người dùng với các đối tượng trên giao diện đồ họa, ví dụ như các thao tác:
  - Di chuyển chuột
  - Kích chuột vào một nút lệnh
  - Dê chuột trên thanh cuộn
  - Ấn một phím, . . .
- Chương trình được thiết kế sẽ tự động phát hiện các sự kiện và có các phản ứng thích hợp với các sự kiện đó.

# Xử lý sự kiện

*Quá trình xử lý sự kiện:*

- Một đối tượng nguồn tạo ra một sự kiện và gửi nó cho một hoặc nhiều bộ “lắng nghe” (listeners).
- Listener chỉ đơn giản là đợi đến khi nó nhận được một sự kiện.
- Listener sẽ xử lý các sự kiện và sau đó quay trở về.

# Một số interface và class xử lý sự kiện

Class Event	Mô tả	Interface Listener
ActionEvent	Khi click vào button, chọn một mục từ menu và trong list (danh sách)	ActionListener
MouseEvent	Các sự kiện khi dịch chuyển, click,vào/ra, ấn/nhả chuột.	MouseListener
KeyEvent	Khi nhập dữ liệu từ bàn phím	KeyListener
ItemEvent	Khi thay đổi các giá trị của textarea hoặc textfield.	ItemListener
WindowEvent	Khi cửa sổ được kích hoạt/không kích hoạt, mở/đóng	WindowListener



# Các ví dụ

## *Ví dụ 1: Sử dụng interface ActionListener*

```
public class TestEvents extends Applet implements ActionListener{

    private int count = 0;
    TextField text;
    Button buttonInc, buttonDec;

    @Override
    public void init() {
        text = new TextField("" + count);
        add(text);
        buttonInc = new Button("Increment");
        add(buttonInc);
        buttonDec = new Button("Decrement");
        add(buttonDec);
    }
}
```

Thực thi interface  
ActionListener để có thể  
xử lý sự kiện.

Đăng ký listener

## Các ví dụ

Anonymous  
inner class

```
buttonInc.addActionListener(this);
buttonDec.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        text.setText("" + --count);
    }
});

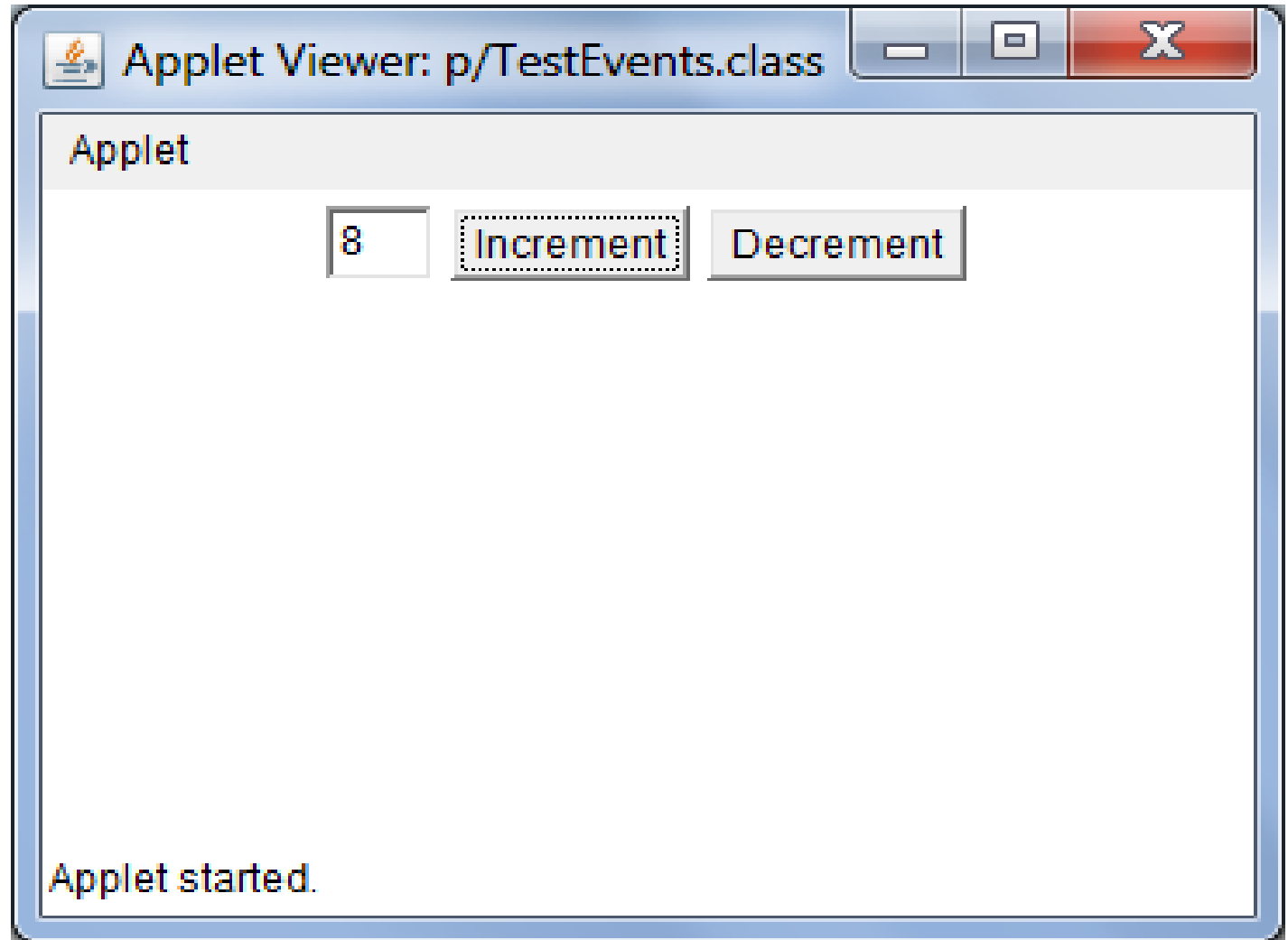
public void actionPerformed(ActionEvent ae) {
    if (ae.getSource() == buttonInc) {
        text.setText("" + ++count);
    }
}
```

Thực thi **actionPerformed()**  
của interface **ActionListener**

# Các ví dụ

*Ví dụ 1: Sử dụng interface ActionListener*

Kết quả:



# Các ví dụ

## Ví dụ 2: Sử dụng interface *MouseListener*

```
public class Mousey extends Applet implements MouseListener {  
    int x1, y1, x2, y2;  
  
    public void init() {  
        setLayout(new FlowLayout());  
        setBounds(100, 100, 300, 300);  
        addMouseListener(this);  
        this.setVisible(true);  
    }  
  
    public void mouseClicked(MouseEvent e) {}  
    public void mouseMoved(MouseEvent e) {}  
    public void mouseEntered(MouseEvent e) {}  
    public void mouseDragged(MouseEvent e) {}  
    public void mouseExited(MouseEvent e) {}  
}
```

Thực thi interface *MouseListener* để có thể xử lý sự kiện.

# Các ví dụ

## Ví dụ 2: Sử dụng interface *MouseListener*

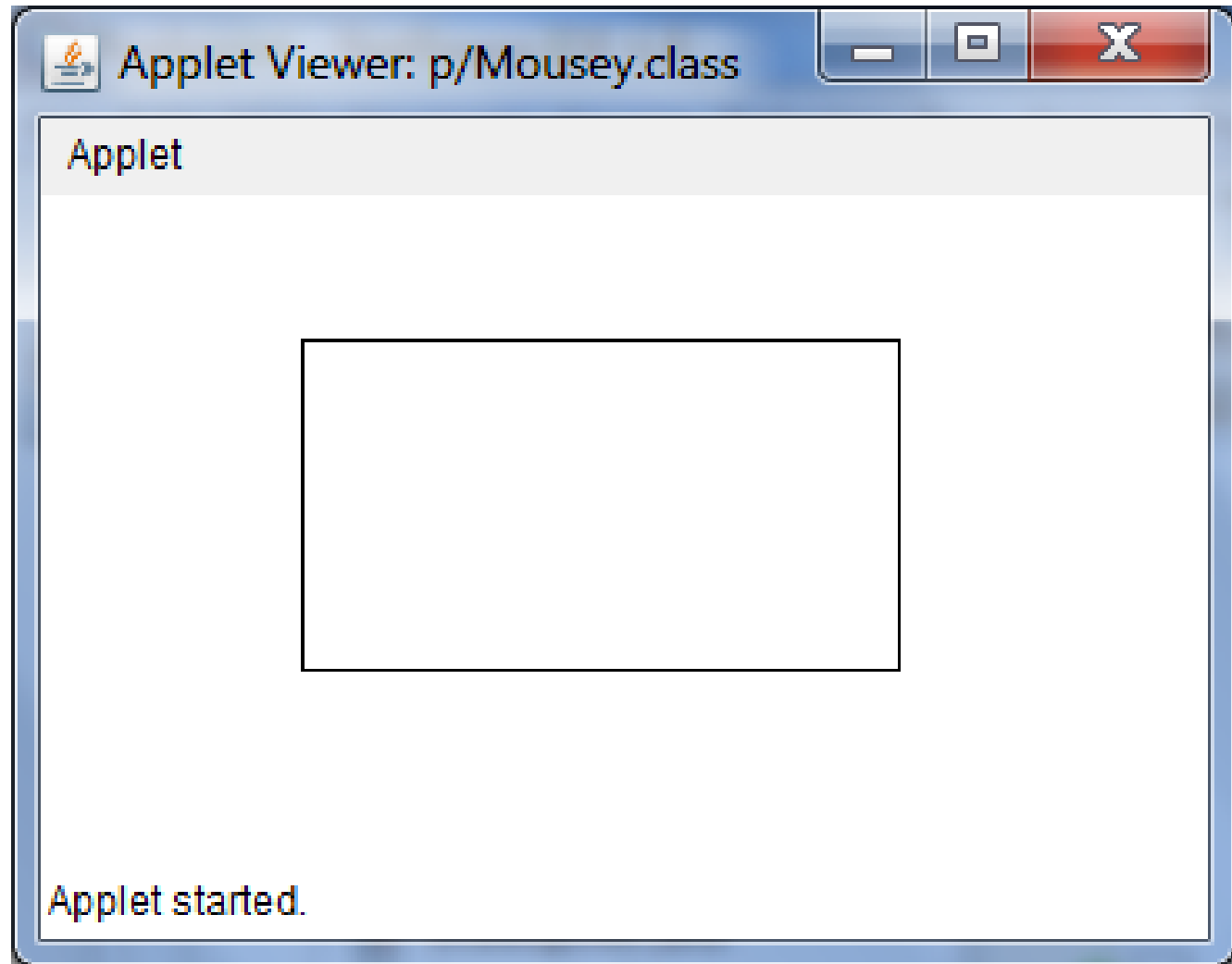
```
public void mousePressed(MouseEvent e) {  
    x1 = e.getX();  
    y1 = e.getY();  
}  
public void mouseReleased(MouseEvent e) {  
    x2 = e.getX();  
    y2 = e.getY();  
    repaint();  
}  
@Override  
public void paint(Graphics g) {  
    g.drawRect(x1, y1, x2 - x1, y2 - y1);  
    x2 = 0;  
    y2 = 0;  
}  
}
```

Vẽ hình chữ nhật khi rê chuột từ **mousePressed**(ấn) đến **mouseReleased**(thả)

# Các ví dụ

*Ví dụ 2: Sử dụng interface `MouseListener`*

Kết quả:



## Tổng kết bài học

- Khái niệm Applets
- Sự khác nhau giữa Applets và Applications
- Vòng đời của applet
- Một số phương thức của class Graphics
- Tạo một applet
- Sử dụng tham số trong Applets
- Xử lý sự kiện
- Một số interface và class xử lý sự kiện
- Một số ví dụ

