Chapter 10: Applets and Advanced Graphics

- The `Applet` Class
- The `<applet>` HTML Tag
- Passing Parameters to Applets
- Conversions Between Applications and Applets
- Running a Program as an Applet and as an Application
- Handling the Mouse Event
- Handling the Keyboard Event
- Model dynamic behavior using sequence diagrams and statecharts diagrams
- Advanced Layout (CardLayout and `GridBagLayout` and using No Layout Manager) (Optional)

Applets on Web

- Dynamic interaction
- Live animation
- Client-side computing

- Applets are designed to run in web-browser, or appletviewer
- Java programming technologies can be used in development of applet
- Does not have a main() method

The Applet Class

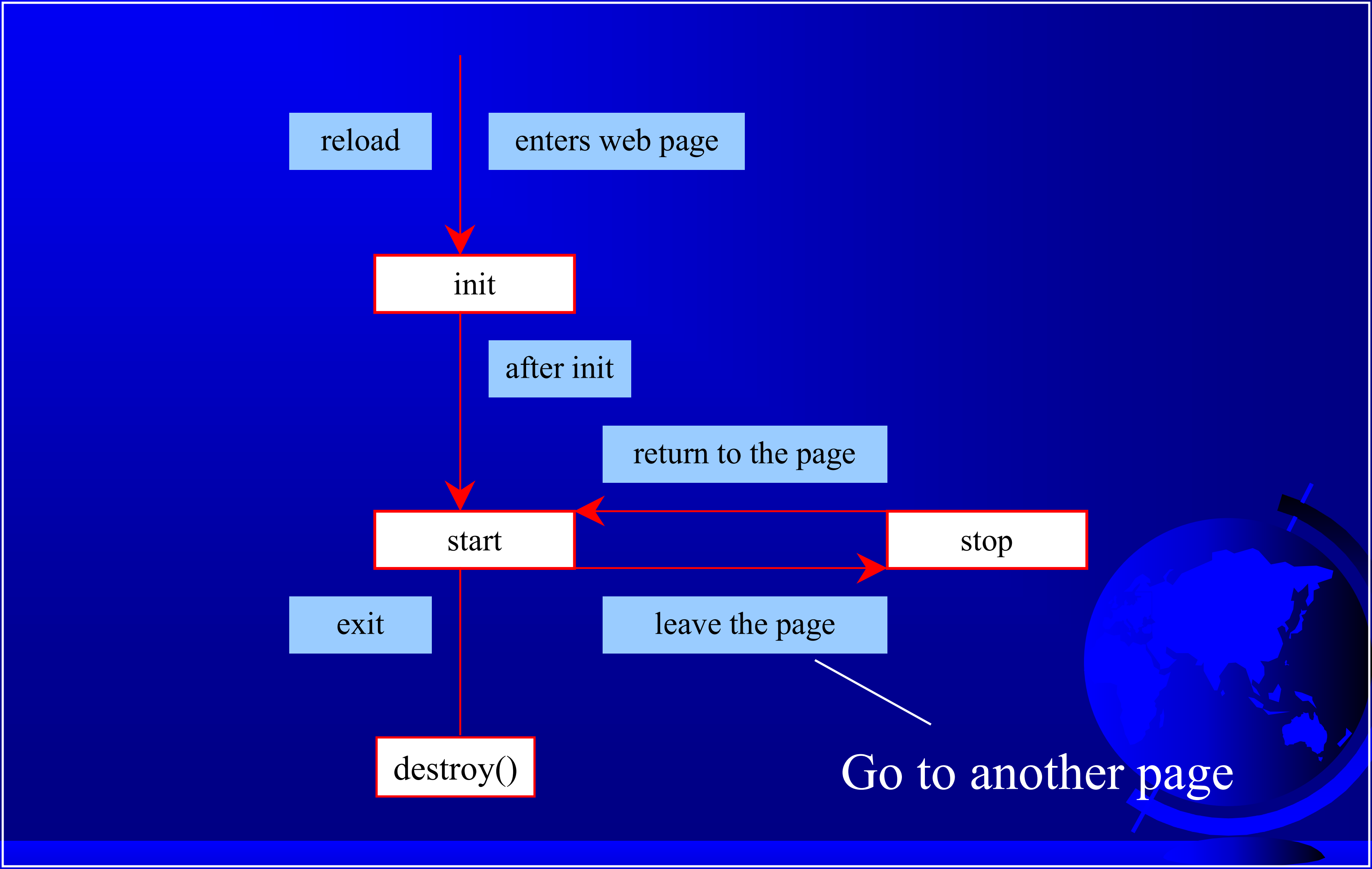
```
public class MyApplet
                extends java.applet.JApplet
{
   public void init()          ——— Initially load or reload
   { ... }
   public void start()         ——— Visit or re-visit the web page
   { ... }
   public void stop()          ——— applet become inactive
   { ... }
   public void destroy()       ——— Web browser exits
   { ... }
   //your other methods
}
```

The `Applet` Class

```
import java.applet.*;


public class MyApplet extends JApplet
{
   public void init()
   { ... }
   public void start()
   { ... }
   public void stop()
   { ... }
   public void destroy()
   { ... }
   //your other methods
}
```

Defaulted doing nothing
But may be customer zed

Browser Calling Applet Methods



The `init()` Method

Invoked when the applet is first loaded and again if the applet is reloaded.

The subclass of applet should override this method if the subclass has an initialization to perform.

Common functions implemented in this method include creating threads (chapter 13), loading images (chapter14), setting up user-interface components (later in this chapter), and getting parameters from the `<applet>` tag in the HTML page (later in this chapter).

The `start()` Method

Invoked after the `init()` method is executed; also called whenever the applet becomes active again after a period of inactivity (for example, when the user returns to the page containing the applet after surfing other Web pages).

A subclass of applet overrides this method if it has any operation that needs to be performed.

Functionality might include restarting threads (for example, to resume an animation) or simply telling the applet to run again.

The `stop()` Method

The opposite of the `start()` method, which is called when the user moves back to the page containing the applet; the `stop()` method is invoked when the user moves off the page.

A subclass of applet overrides this method if it has any operation that needs to be performed each time the web page containing the applet is no long visible.

When the user leaves the page, any threads the applet has started—but not completed—will continue to run.

The `destroy()` Method

Invoked when the browser exits normally to inform the applet that it is no longer needed and that it should release any resources it has allocated.

A subclass of applet overrides this method if it has any operation that needs to be performed before it is destroyed.

Usually, you will not need to override this method unless you need to release specific resources, such as threads that the applet created.

Applet and JApplet

Applet does not work well with Aswing components

JApplet (javax.swing.JApplet) is specially designed. It is a subclass of Applet class

By default, the content pane of Japplet uses BorderLayout

Example 10.1 Using Applets

- Objective: Compute mortgages. The applet enables the user to enter the annual interest rate, the number of years, and the loan amount. Click the Compute Mortgage button, and the applet displays the monthly payment and the total payment.

```
// MortgageApplet.java: Applet for computing mortgage payments
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.TitledBorder;

public class MortgageApplet extends JApplet
  implements ActionListener
{
  // Declare and create text fields for interest rate
  // year, loan amount, monthly payment, and total payment
  private JTextField jtfAnnualInterestRate = new JTextField();
  private JTextField jtfNumOfYears = new JTextField();
  private JTextField jtfLoanAmount = new JTextField();
  private JTextField jtfMonthlyPayment = new JTextField();
  private JTextField jtfTotalPayment = new JTextField();
```

```
// Declare and create a Compute Mortgage button
  private JButton jbtComputeMortgage =
          new JButton("Compute Mortgage");

  // Initialize user interface
  public void init()
  {
    // Set properties on the text fields
    jtfMonthlyPayment.setEditable(false);
    jtfTotalPayment.setEditable(false);

    // Right align text fields
    jtfAnnualInterestRate.setHorizontalAlignment(JTextField.RIGHT);
    jtfNumOfYears.setHorizontalAlignment(JTextField.RIGHT);
    jtfLoanAmount.setHorizontalAlignment(JTextField.RIGHT);
    jtfMonthlyPayment.setHorizontalAlignment(JTextField.RIGHT);
    jtfTotalPayment.setHorizontalAlignment(JTextField.RIGHT);
```

```
// Panel p1 to hold labels and text fields
JPanel p1 = new JPanel();
p1.setLayout(new GridLayout(5, 2));
p1.add(new Label("Annual Interest Rate"));
p1.add(jtfAnnualInterestRate);
p1.add(new Label("Number of Years"));
p1.add(jtfNumOfYears);
p1.add(new Label("Loan Amount"));
p1.add(jtfLoanAmount);
p1.add(new Label("Monthly Payment"));
p1.add(jtfMonthlyPayment);
p1.add(new Label("Total Payment"));
p1.add(jtfTotalPayment);
p1.setBorder(new
  TitledBorder("Enter interest rate, year and loan amount"));
```

```
// Panel p2 to hold the button
   JPanel p2 = new JPanel();
   p2.setLayout(new FlowLayout(FlowLayout.RIGHT));
   p2.add(jbtComputeMortgage);

   // Add the components to the applet
   getContentPane().add(p1, BorderLayout.CENTER);
   getContentPane().add(p2, BorderLayout.SOUTH);

   // Register listener
   jbtComputeMortgage.addActionListener(this);
 }
```

```
// Handler for the "Compute Mortgage" button
 public void actionPerformed(ActionEvent e)
  {if (e.getSource() == jbtComputeMortgage)
    {// Get values from text fields
      double interest =
        (Double.valueOf(jtfAnnualInterestRate.getText())).doubleValue();
      int year =
        (Integer.valueOf(jtfNumOfYears.getText())).intValue();
      double loan =
        (Double.valueOf(jtfLoanAmount.getText())).doubleValue();
      // Create a mortgage object
      Mortgage m = new Mortgage(interest, year, loan);
      // Display monthly payment and total payment
      jtfMonthlyPayment.setText(String.valueOf(m.monthlyPayment()));
      jtfTotalPayment.setText(String.valueOf(m.totalPayment()));
    }
  }
}
```

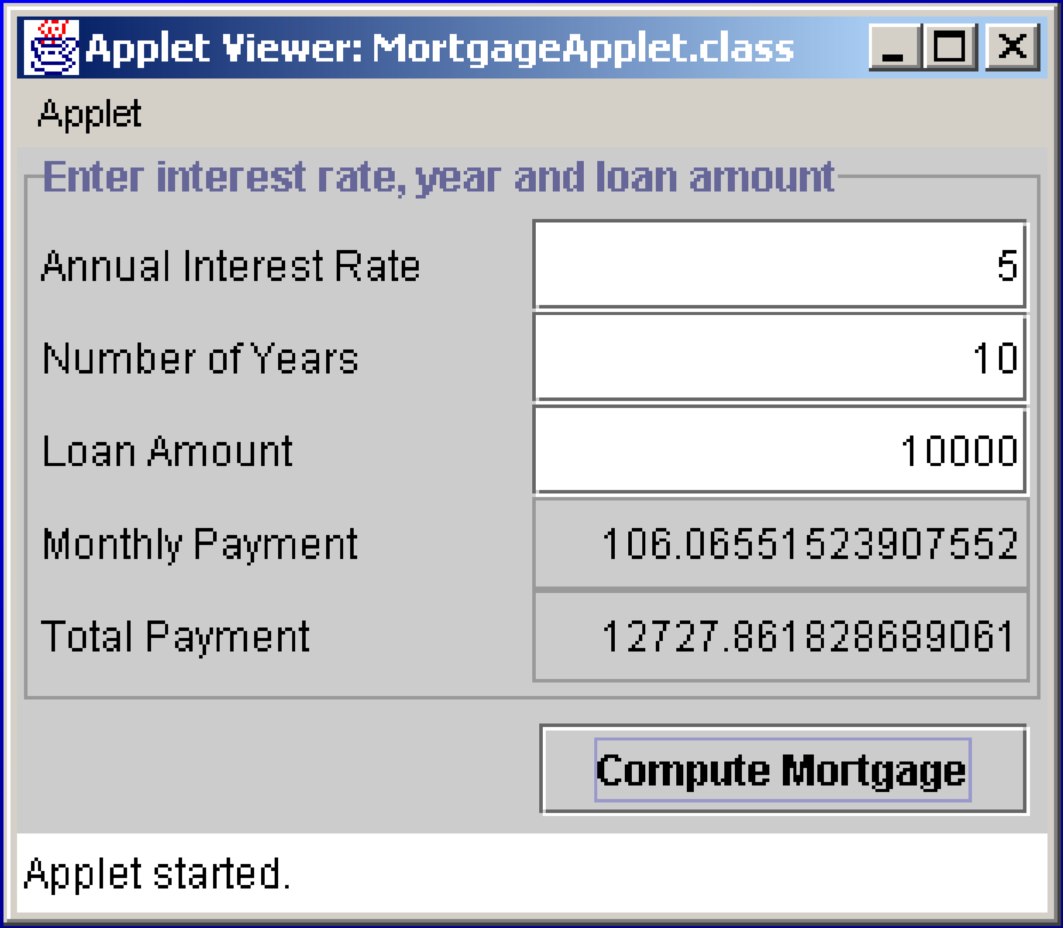
```
// Mortgage.java: Encapsulate mortgage information
public class Mortgage
{
  private double annualInterestRate;
  private int numOfYears;
  private double loanAmount;

  // Default constructor
  public Mortgage()
  {
  }

  // Construct a mortgage with specified annual interest rate,
  // number of years and loan amount
  public Mortgage(double annualInterestRate, int numOfYears,
    double loanAmount)
  {
```

```
this.annualInterestRate = annualInterestRate;
  this.numOfYears = numOfYears;
  this.loanAmount = loanAmount;
 }

 // Getter method for annualInterestRate
 public double getAnnualInterestRate()
 {
  return annualInterestRate;
 }

 // Setter method for annualInterestRate
 public void setAnnualInterestRate(double annualInterestRate)
 {
  this.annualInterestRate = annualInterestRate;
 }
```

```
// Getter method for numOfYears
 public int getNumOfYears()
 {
  return numOfYears;
 }

 // Setter method for numOfYears
 public void setNumOfYears(int numOfYears)
 {
  this.numOfYears = numOfYears;
 }

 // Getter method for loanAmount
 public double getLoanAmount()
 {
  return loanAmount;
 }
```

```
// Setter method for loanAmount
 public void setLoanAmount(double loanAmount)
 {
   this.loanAmount = loanAmount;
 }

 public double monthlyPayment()      // Find monthly payment
 {
   double monthlyInterestRate = annualInterestRate/1200;
   return loanAmount*monthlyInterestRate/
     (1 - (Math.pow(1/(1 + monthlyInterestRate), numOfYears*12)));
 }

 public double totalPayment()          // Find total payment
 {
   return monthlyPayment()*numOfYears*12;
 }
}
```

```
<APPLET CODE = "MortgageApplet.class"
WIDTH = 300 HEIGHT = 200
ALT = "You must have a JDK 1.2-enabled browser to
        view the applet">

</APPLET>
```



Enter interest rate, year and loan amount

Annual Interest Rate	5
Number of Years	10
Loan Amount	10000
Monthly Payment	106.06551523907552
Total Payment	12727.861828689061

Applet started.

Writing Applets

- Always extends the `JApplet` class, which is a subclass of `Applet` for Swing components.
- Override `init()`, `start()`, `stop()`, and `destroy()` if necessary. By default, these methods are empty.
- Add your own methods and data if necessary.
- Applets are always embedded in an HTML page.

Running Applets in Java Plug-In

Why to Use Java Plug-In?

Java Plug-in enables Web browsers to run Java applets consistently on all the platforms.

How to Use Java Plug-In

- Convert the HTML file to a new HTML file using the HTMLConverter Utility. The new HTML file contains the tags for invoking the Java Plug-In.

- If the Plug-In is not installed, the new HTML file automatically downloads it from the Sun JavaSoft Web site.

Running applets in Java enabled Web browsers

- HotJava
- Netscape supports JDK 1.0
- IE supports JDK 1.0
- Vendors fail behind the rapid development of JDK
- Sun's Java plug-in

The <applet> HTML Tag

```
<applet
   code=classfilename.class
   width=applet_viewing_width_in_pixels
   height=applet_viewing_height_in_pixels
   [archive=archivefile]
   [codebase=applet_url]
   [vspace=vertical_margin]
   [hspace=horizontal_margin]
   [align=applet_alignment]
   [alt=alternative_text]
>
<param name=param_name1
   value=param_value1>
</applet>
```

Passing Parameters to Applets

```
<applet
  code = "DisplayMessage.class"
  width = 200
  height = 50>
<param name=MESSAGE value="Welcome
    to Java">
<param name=X value=20>
<param name=Y value=20>
alt="You must have a Java-enabled
browser to view the applet"
</applet>
```

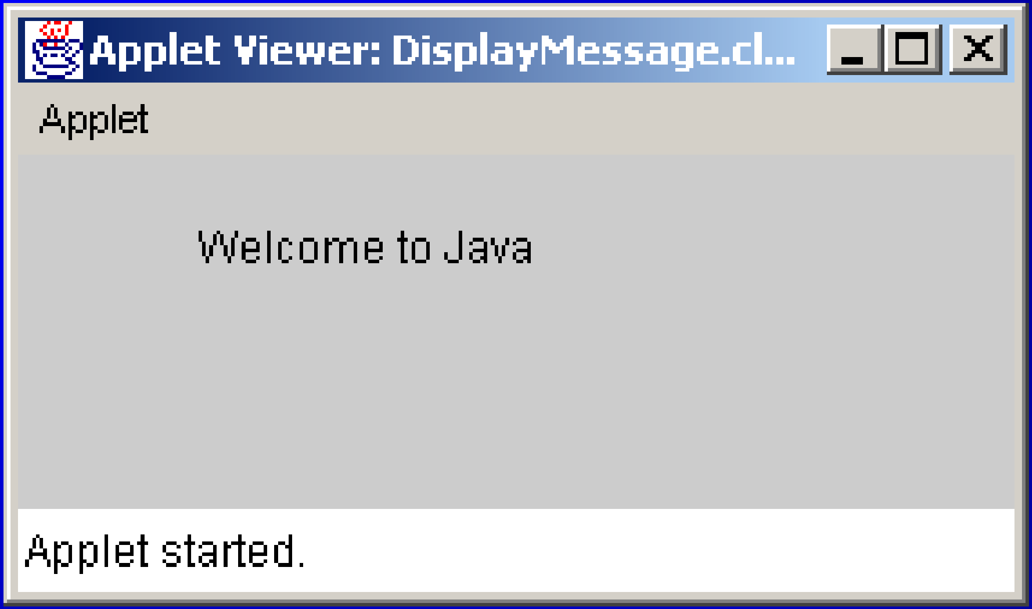
Example 10.2 Passing Parameters to Java Applets

- Objective: Display a message at a specified location. The message and the location (x, y) are obtained from the HTML source.

```
// DisplayMessage.java: Display a message on a panel in the applet
import javax.swing.*;

public class DisplayMessage extends JApplet
{
  private String message = "A defualt message"; // Message to display
  private int x = 20; // Default x coordinate
  private int y = 20; // Default y coordinate

  // Initialize the applet
  public void init()
  {
    // Get parameter values from the HTML file
    message = getParameter("MESSAGE");
    x = Integer.parseInt(getParameter("X"));
    y = Integer.parseInt(getParameter("Y"));
```

```
// Create a message panel
    MessagePanel messagePanel = new MessagePanel(message);
    messagePanel.setXCoordinate(x);
    messagePanel.setYCoordinate(y);

    // Add the message panel to the applet
    getContentPane().add(messagePanel);
  }
}
```

```
<html>
<head>
<title>Passing Parameters to Java Applets</title>
</head>
<body>
This applet gets a message from the HTML page and displays it.
<p>
<applet
        code = "DisplayMessage.class"
        width = 200
        height = 50
        alt="You must have a Java-enabled browser to view the applet">
        <param name=MESSAGE value="Welcome to Java">
        <param name=X value=50>
        <param name=Y value=30>
</applet>
</body>
</html>
```



Welcome to Java

Applet started.

Conversions Between Applications and Applets

- Conversions between applications and applets are simple and easy.
- You can always convert an applet into an application.
- You can convert an application to an applet as long as security restrictions are not violated.

Applets into Applications

- Steps:
  - Eliminate the html page. If the applet gets parameters from html page, you can handle them either by command line or set by using assignment
  - Derive the main class named NewClass, for example, use JFrame instead of using JApplet
  - Write a constructor in the new class to contain the code in the init() and the start method

Applets into Applications

– Add a main method, as follows

```
public static void main()
{
NewClass from =new NewClass();
frame.resize(width,height);
frame.setVisible(true);
}
```

– Add a title for the window using the setTitle() method

Example 10.3 Converting Applets into Applications

- Objective: Convert `MortgageApplet` (from Example 10.1) to a Java application.

```
// MortgageApplet.java: Applet for computing mortgage payments
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.TitledBorder;                 JFrame
                           MorgageApplication
                        ↗                          ↗

public class MortgageApplet extends JApplet
                                  implements ActionListener
{
  // Declare and create text fields for interest rate
  // year, loan amount, monthly payment, and total payment
  private JTextField jtfAnnualInterestRate = new JTextField();
  private JTextField jtfNumOfYears = new JTextField();
  private JTextField jtfLoanAmount = new JTextField();
  private JTextField jtfMonthlyPayment = new JTextField();
  private JTextField jtfTotalPayment = new JTextField();
```

```
// Declare and create a Compute Mortgage button
  private JButton jbtComputeMortgage =
          new JButton("Compute Mortgage");

                                        MortgageApplication()

  public void init()
  {
    // Set properties on the text fields
    jtfMonthlyPayment.setEditable(false);
    jtfTotalPayment.setEditable(false);

    // Right align text fields
    jtfAnnualInterestRate.setHorizontalAlignment(JTextField.RIGHT);
    jtfNumOfYears.setHorizontalAlignment(JTextField.RIGHT);
    jtfLoanAmount.setHorizontalAlignment(JTextField.RIGHT);
    jtfMonthlyPayment.setHorizontalAlignment(JTextField.RIGHT);
    jtfTotalPayment.setHorizontalAlignment(JTextField.RIGHT);
```

```
// Panel p1 to hold labels and text fields
JPanel p1 = new JPanel();
p1.setLayout(new GridLayout(5, 2));
p1.add(new Label("Annual Interest Rate"));
p1.add(jtfAnnualInterestRate);
p1.add(new Label("Number of Years"));
p1.add(jtfNumOfYears);
p1.add(new Label("Loan Amount"));
p1.add(jtfLoanAmount);
p1.add(new Label("Monthly Payment"));
p1.add(jtfMonthlyPayment);
p1.add(new Label("Total Payment"));
p1.add(jtfTotalPayment);
p1.setBorder(new
  TitledBorder("Enter interest rate, year and loan amount"));
```

```
// Panel p2 to hold the button
   JPanel p2 = new JPanel();
   p2.setLayout(new FlowLayout(FlowLayout.RIGHT));
   p2.add(jbtComputeMortgage);

   // Add the components to the applet
   getContentPane().add(p1, BorderLayout.CENTER);
   getContentPane().add(p2, BorderLayout.SOUTH);

   // Register listener
   jbtComputeMortgage.addActionListener(this);
 }
```

```
// Add a main method
 public static void main(String[] args)
 {
   MortgageApplication frame = new MortgageApplication();
   frame.setTitle("Mortgage Application");
   frame.setSize(400, 200);
   // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.setVisible(true);
 }
```

```
// Handler for the "Compute Mortgage" button
 public void actionPerformed(ActionEvent e)
 {if (e.getSource() == jbtComputeMortgage)
   {// Get values from text fields
     double interest =
       (Double.valueOf(jtfAnnualInterestRate.getText())).doubleValue();
     int year =
       (Integer.valueOf(jtfNumOfYears.getText())).intValue();
     double loan =
       (Double.valueOf(jtfLoanAmount.getText())).doubleValue();
     // Create a mortgage object
     Mortgage m = new Mortgage(interest, year, loan);
     // Display monthly payment and total payment
     jtfMonthlyPayment.setText(String.valueOf(m.monthlyPayment()));
     jtfTotalPayment.setText(String.valueOf(m.totalPayment()));
   }
 }
}
```

```
// MortgageApplication.java: Application for
// computing mortgage payments
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.TitledBorder;

public class MortgageApplication extends JFrame
  implements ActionListener
{
  // Declare and create text fields for interest rate
  // year, loan amount, monthly payment, and total payment
  private JTextField jtfAnnualInterestRate = new JTextField(10);
  private JTextField jtfNumOfYears = new JTextField(10);
  private JTextField jtfLoanAmount = new JTextField(10);
  private JTextField jtfMonthlyPayment = new JTextField(10);
  private JTextField jtfTotalPayment = new JTextField(10);
```

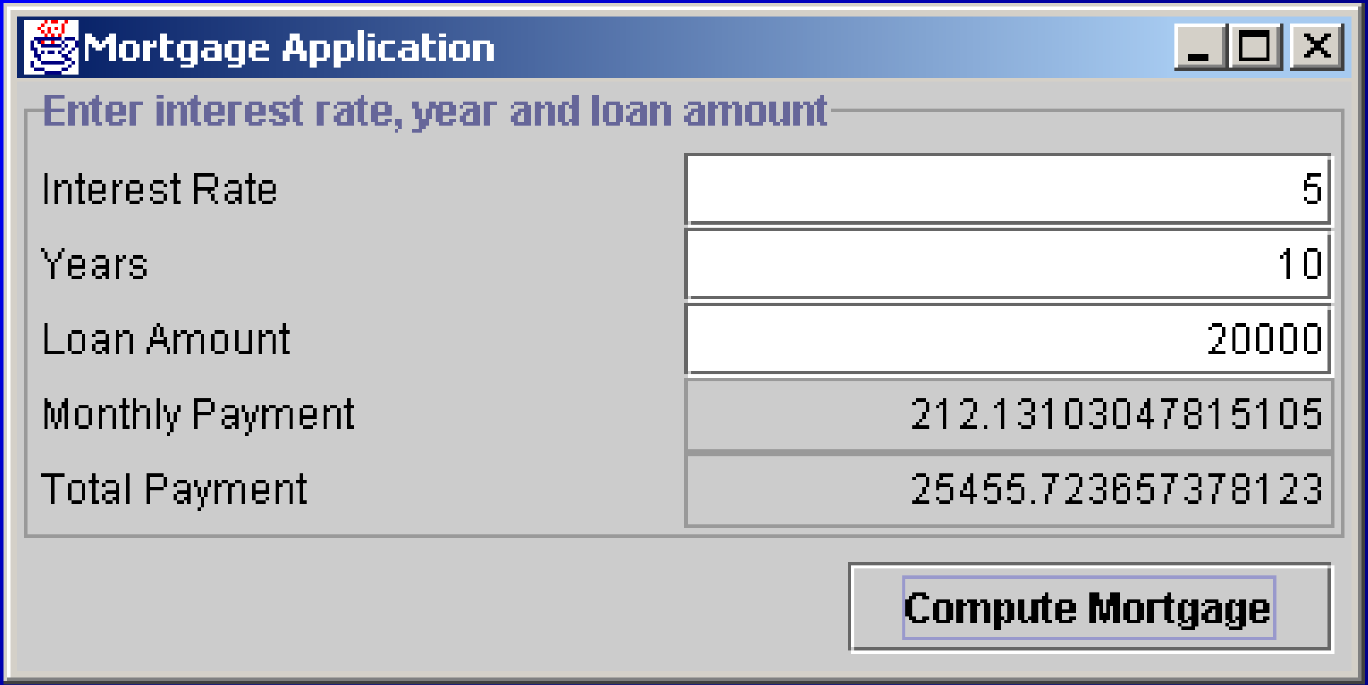
```
// Declare and create a Compute Mortgage button
private JButton jbtComputeMortgage =
        new JButton("Compute Mortgage");
// Constructor (replacing the init() method in the applet)
public MortgageApplication()
{
  // Set properties on the text fields
  jtfMonthlyPayment.setEditable(false);
  jtfTotalPayment.setEditable(false);

  // Right align text fields
  jtfAnnualInterestRate.setHorizontalAlignment(JTextField.RIGHT);
  jtfNumOfYears.setHorizontalAlignment(JTextField.RIGHT);
  jtfLoanAmount.setHorizontalAlignment(JTextField.RIGHT);
  jtfMonthlyPayment.setHorizontalAlignment(JTextField.RIGHT);
  jtfTotalPayment.setHorizontalAlignment(JTextField.RIGHT);
```

```
// Panel p1 to hold labels and text fields
JPanel p1 = new JPanel();
p1.setLayout(new GridLayout(5,2));
p1.add(new Label("Interest Rate"));
p1.add(jtfAnnualInterestRate);
p1.add(new Label("Years "));
p1.add(jtfNumOfYears);
p1.add(new Label("Loan Amount"));
p1.add(jtfLoanAmount);
p1.add(new Label("Monthly Payment"));
p1.add(jtfMonthlyPayment);
p1.add(new Label("Total Payment"));
p1.add(jtfTotalPayment);
p1.setBorder(new
  TitledBorder("Enter interest rate, year and loan amount"));
```

```
// Panel p2 to hold the button
   JPanel p2 = new JPanel();
   p2.setLayout(new FlowLayout(FlowLayout.RIGHT));
   p2.add(jbtComputeMortgage);

   // Add the components to the applet
   getContentPane().add(p1, BorderLayout.CENTER);
   getContentPane().add(p2, BorderLayout.SOUTH);

   // Register listener
   jbtComputeMortgage.addActionListener(this);
 }
```

```
// Add a main method
   public static void main(String[] args)
   {
     MortgageApplication frame = new MortgageApplication();
     frame.setTitle("Mortgage Application");
     frame.setSize(400, 200);
     // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     frame.setVisible(true);
   }
```

```
// Handler for the "Compute" button
  public void actionPerformed(ActionEvent e)
  {
    if (e.getSource() == jbtComputeMortgage)
    {
      // Get values from text fields
      double interest =
        (Double.valueOf(jtfAnnualInterestRate.getText())).doubleValue();
      int year =
        (Integer.valueOf(jtfNumOfYears.getText())).intValue();
      double loan =
        (Double.valueOf(jtfLoanAmount.getText())).doubleValue();
// Create a mortgage object
      Mortgage m = new Mortgage(interest, year, loan);

      // Display monthly payment and total payment
      jtfMonthlyPayment.setText(String.valueOf(m.monthlyPayment()));
      jtfTotalPayment.setText(String.valueOf(m.totalPayment()));
    }
  }
}
```



Enter interest rate, year and loan amount

Interest Rate	5
Years	10
Loan Amount	20000
Monthly Payment	212.1310304781505
Total Payment	25455.72365737812 3

Converting Applications to Applets

- If the conversion does not violate the security constraints imposed on applets, we can convert applications to applets, by the following steps
  - Create a html page and set getParameter in the init() method
  - Derive the main class from JApplet instead of JFrame
  - Replace the application's constructor by the init method
  - Eliminate the main() method
  - Eliminate setTitle() method

Example 10.4 Converting Applications into Applets

- Objective: Convert a Java application that performs simple numeric calculations (`MenuDemo`, from Example 9.11) into a Java applet.

```java
// AppletMenuDemo.java: Use menus in an applet
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class AppletMenuDemo extends JApplet
  implements ActionListener
{
  // Text fields for Number 1, Number 2, and Result
  private JTextField jtfNum1, jtfNum2, jtfResult;

  // Buttons "Add", "Subtract", "Multiply" and "Divide"
  private JButton jbtAdd, jbtSub, jbtMul, jbtDiv;

  // Menu items "Add", "Subtract", "Multiply","Divide" and "Close"
  private JMenuItem jmiAdd, jmiSub, jmiMul, jmiDiv, jmiClose;
```

```
// Initialize the applet
public void init()
{
  // Create menu bar
  JMenuBar jmb = new JMenuBar();

  // Add menu "Operation" to menu bar
  JMenu operationMenu = new JMenu("Operation");
  operationMenu.setMnemonic('O');
  jmb.add(operationMenu);

  // Add menu "Exit" in menu bar
  JMenu exitMenu = new JMenu("Exit");
  exitMenu.setMnemonic('E');
  jmb.add(exitMenu);
```

```
// Add menu items with mnemonics to menu "Operation"
   operationMenu.add(jmiAdd= new JMenuItem("Add", 'A'));
   operationMenu.add(jmiSub = new JMenuItem("Subtract", 'S'));
   operationMenu.add(jmiMul = new JMenuItem("Multiply", 'M'));
   operationMenu.add(jmiDiv = new JMenuItem("Divide", 'D'));
   exitMenu.add(jmiClose = new JMenuItem("Close", 'C'));
   // Set keyboard accelerators
   jmiAdd.setAccelerator(
     KeyStroke.getKeyStroke(KeyEvent.VK_A,
                                                ActionEvent.CTRL_MASK));
   jmiSub.setAccelerator(
     KeyStroke.getKeyStroke(KeyEvent.VK_S,
                                                  ActionEvent.CTRL_MASK));
   jmiMul.setAccelerator(
     KeyStroke.getKeyStroke(KeyEvent.VK_M,
                                               ActionEvent.CTRL_MASK));
```

```
 jmiDiv.setAccelerator(
     KeyStroke.getKeyStroke(KeyEvent.VK_D,
                         ActionEvent.CTRL_MASK));

// Panel p1 to hold text fields and labels
    JPanel p1 = new JPanel();
    p1.setLayout(new FlowLayout());
    p1.add(new JLabel("Number 1"));
    p1.add(jtfNum1 = new JTextField(3));
    p1.add(new JLabel("Number 2"));
    p1.add(jtfNum2 = new JTextField(3));
    p1.add(new JLabel("Result"));
    p1.add(jtfResult = new JTextField(4));
    jtfResult.setEditable(false);
```

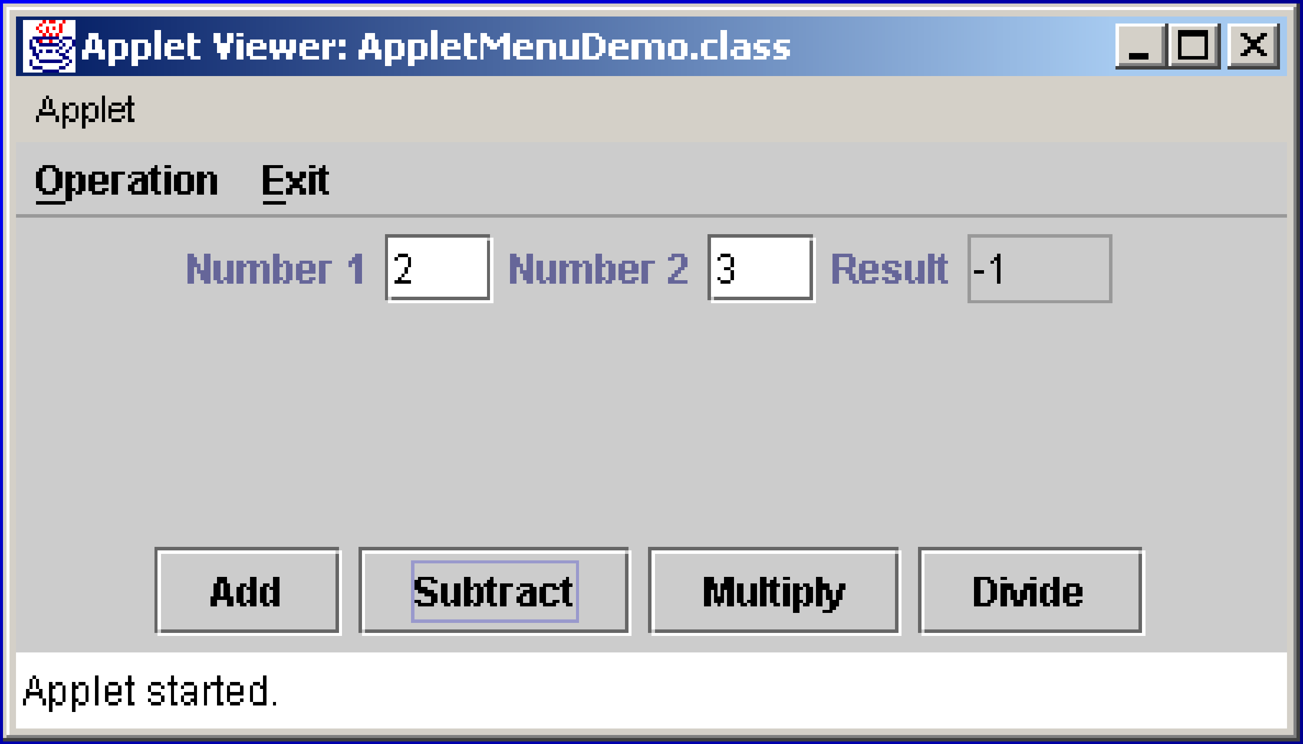
```
// Panel p2 to hold buttons
    JPanel p2 = new JPanel();
    p2.setLayout(new FlowLayout());
    p2.add(jbtAdd = new JButton("Add"));
    p2.add(jbtSub = new JButton("Subtract"));
    p2.add(jbtMul = new JButton("Multiply"));
    p2.add(jbtDiv = new JButton("Divide"));

    // Add menu bar to the applet
    setJMenuBar(jmb);

    // Add panels to the applet
    getContentPane().add(p1, BorderLayout.CENTER);
    getContentPane().add(p2, BorderLayout.SOUTH);
```

```
// Register listeners
  jbtAdd.addActionListener(this);
  jbtSub.addActionListener(this);
  jbtMul.addActionListener(this);
  jbtDiv.addActionListener(this);
  jmiAdd.addActionListener(this);
  jmiSub.addActionListener(this);
  jmiMul.addActionListener(this);
  jmiDiv.addActionListener(this);
  jmiClose.addActionListener(this);
 }

 // Handle ActionEvent from buttons and menu items
 public void actionPerformed(ActionEvent e)
 {
  String actionCommand = e.getActionCommand();
```

```
// Handling button events
   if (e.getSource() instanceof JButton)
   {
     if ("Add".equals(actionCommand))
       calculate('+');
     else if ("Subtract".equals(actionCommand))
       calculate('-');
     else if ("Multiply".equals(actionCommand))
       calculate('*');
     else if ("Divide".equals(actionCommand))
       calculate('/');
   }
   else if (e.getSource() instanceof JMenuItem)
   {
     // Handling menu item events
     if ("Add".equals(actionCommand))
       calculate('+');
```

```
else if ("Subtract".equals(actionCommand))
      calculate('-');
    else if ("Multiply".equals(actionCommand))
      calculate('*');
    else if ("Divide".equals(actionCommand))
      calculate('/');
    else if ("Close".equals(actionCommand))
      System.out.println("Can't close applet");
  }
 }

 // Calculate and show the result in jtfResult
 private void calculate(char operator)
 {
   // Obtain Number 1 and Number 2
   int num1 = (Integer.parseInt(jtfNum1.getText().trim()));
   int num2 = (Integer.parseInt(jtfNum2.getText().trim()));
   int result = 0;
```

```
    // Perform selected operation
    switch (operator)
    {
      case '+': result = num1 + num2;
                break;
      case '-': result = num1 - num2;
                break;
      case '*': result = num1 * num2;
                break;
      case '/': result = num1 / num2;
    }

    // Set result in jtfResult
    jtfResult.setText(String.valueOf(result));
  }
}
```


Applet Viewer: AppletMenuDemo.class
Applet
Operation
Exit
Number 1
2
Number 2
3
Result
-1
Add
Subtract
Multiply
Divide
Applet started.

Running a Program as an Applet and as an Application

- You can implement a main() method in an applet that will run as an application or as an applet using the same program.
- Usually it starts with an applet. You can add a main method, in which an object of the applet class is created.
- Technically

Running a Program as an Applet and as an Application

```
public static void main(String[] args)
 {
   // Create a frame
   JFrame frame = new JFrame("DisplayMessageApp");

   // Create an instance of the applet
   DisplayMessageApp applet = new DisplayMessageApp();

   // It runs as an application
   applet.isStandalone = true;
// Get parameters from the command line
    applet.getCommandLineParameters(args);

    // Add the applet instance to the frame
    frame.getContentPane().add(applet, BorderLayout.CENTER);

    // Invoke init() and start()
    applet.init();
    applet.start();

    // Display the frame
    frame.setSize(300, 300);
    // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
```

Example 10.5
Running a Program as an Applet and as an Application

- Objective: Modify `MortgageApplet` to enable it to run both as an applet and as an application.

```
// DisplayMessageApp.java:
// The program can run as an applet or application
import javax.swing.*;
import java.awt.BorderLayout;
import java.awt.Font;

public class DisplayMessageApp extends JApplet
{
  private String message = "A default message"; // Message to display
  private int x = 20; // Default x coordinate
  private int y = 20; // Default y coordinate

  // Determine if it is application
  private boolean isStandalone = false;
```

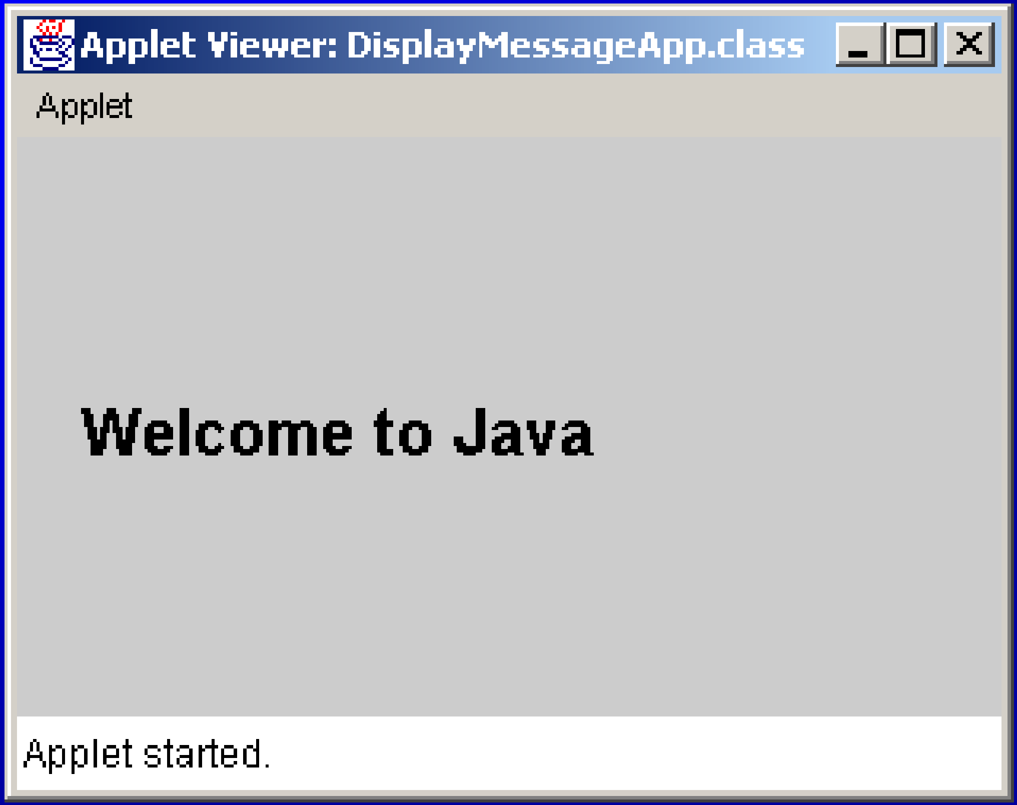
```
// Initialize the applet
 public void init()
 {
  if (!isStandalone)
  {
   // Get parameter values from the HTML file
   message = getParameter("MESSAGE");
   x = Integer.parseInt(getParameter("X"));
   y = Integer.parseInt(getParameter("Y"));
  }

  // Create a message panel
  MessagePanel messagePanel = new MessagePanel(message);
  messagePanel.setFont(new Font("SansSerif", Font.BOLD, 20));
  messagePanel.setXCoordinate(x);
  messagePanel.setYCoordinate(y);
```

```
// Add the message panel to the applet
   getContentPane().add(messagePanel);
 }

 // Main method with three arguments:
 // args[0]: x coordinate
 // args[1]: y coordinate
 // args[2]: message
 public static void main(String[] args)
 {
   // Create a frame
   JFrame frame = new JFrame("DisplayMessageApp");

   // Create an instance of the applet
   DisplayMessageApp applet = new DisplayMessageApp();
```

```
// It runs as an application
  applet.isStandalone = true;

  // Get parameters from the command line
  applet.getCommandLineParameters(args);

  // Add the applet instance to the frame
  frame.getContentPane().add(applet, BorderLayout.CENTER);

  // Invoke init() and start()
  applet.init();
  applet.start();
  // Display the frame
  frame.setSize(300, 300);
  // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  frame.setVisible(true);
 }
```

```
// Get command line parameters
 public void getCommandLineParameters(String[] args)
 {
  // Check usage and get x, y and message
  if (args.length != 3)
  {
   System.out.println(
    "Usage: java DisplayMessageApp x y message");
   System.exit(0);
  }
  else
  {
   x = Integer.parseInt(args[0]);
   y = Integer.parseInt(args[1]);
   message = args[2];
  }
 }
}
```

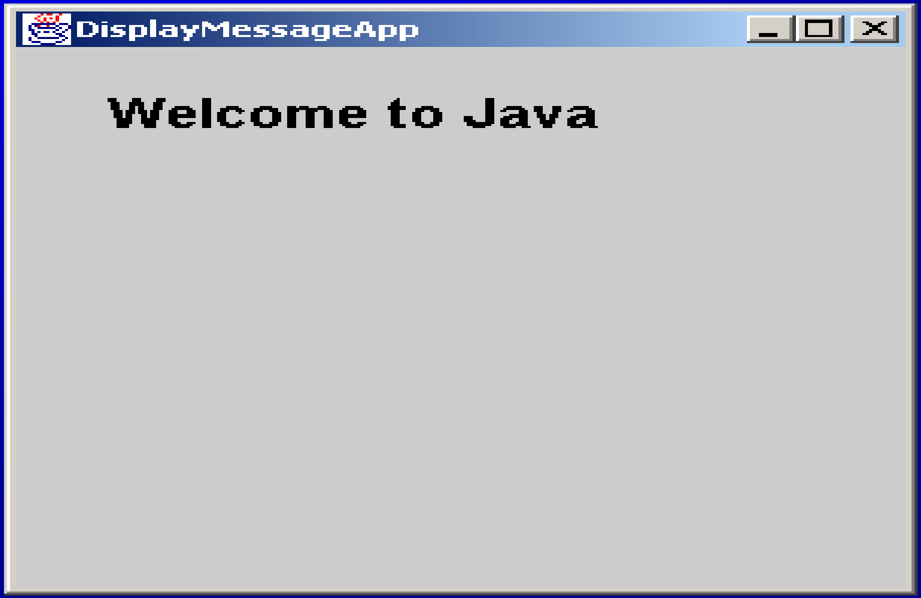
```
<html><head>
<title>Passing Parameters to Java Applets</title></head>
<body>
This applet gets a message from the HTML page and displays it.
<p>
<applet
  code = "DisplayMessageApp.class"
  width = 200
  height = 200
  alt="You must have a Java 2 browser to view the applet"
                              VIEWASTEXT>
  <param name=MESSAGE value="Welcome to Java">
  <param name=X value=20>
  <param name=Y value=100>
</applet>
</body>
</html>
```



```
Welcome to Java
```

```
Applet started.
```

```
java DisplayMessageApp 30 40 "Welcome to Java"
```



DisplayMessageApp

Welcome to Java

Mouse Events

- Mouse event is generated whenever a mouse is clicked, released, moved, or dragged on a component.
- Mouse event object can capture the event such as number of clicks and mouse location.

Handling Mouse Events

- Java provides two listener interfaces, `MouseListener` and `MouseMotionListener`, to handle mouse events.
- The `MouseListener` listens for actions such as when the mouse is pressed, released, entered, exited, or clicked.
- The `MouseMotionListener` listens for actions such as dragging or moving the mouse.

Mouse Events Handlers

- MouseEntered(MouseEvent e) and mouseExit(MouseEvent e) handlers
  - are invoked when a mouse enters a component or exit the component (such as a mouse enter a button)
- mousePressed(MouseEvent e) and mouseReleased(MouseEvent e) handlers
  - Are invoked when a mouse is pressed or released.
- mouseClicked(MouseEvent e) handles
  - Is invoked when a mouse is pressed and then released

Mouse Events Handlers

- mouseMoved(MouseEvent e) handler
  - Is invoked when the mouse is moved without a button being pressed.
- mouseMoved(MouseEvent e) handler
  - Is invoked when the mouse is moved with a button pressed.

Mouse Events Handlers

- Point class is often used for handling mouse events, Point(intx, int y)
- Point class has a method called move(int x, int y) which can help relocate mouse location.

Mouse Events Handlers

- MouseEvent has the following methods which can be used when mouse event occurs.
  - public int getClick(): return the number of clicks
  - public Point getPoint(): return x and y coordinates
  - public int getX(): return x coordinate
  - public int getY(): return y coordinate

Mouse Events Handlers

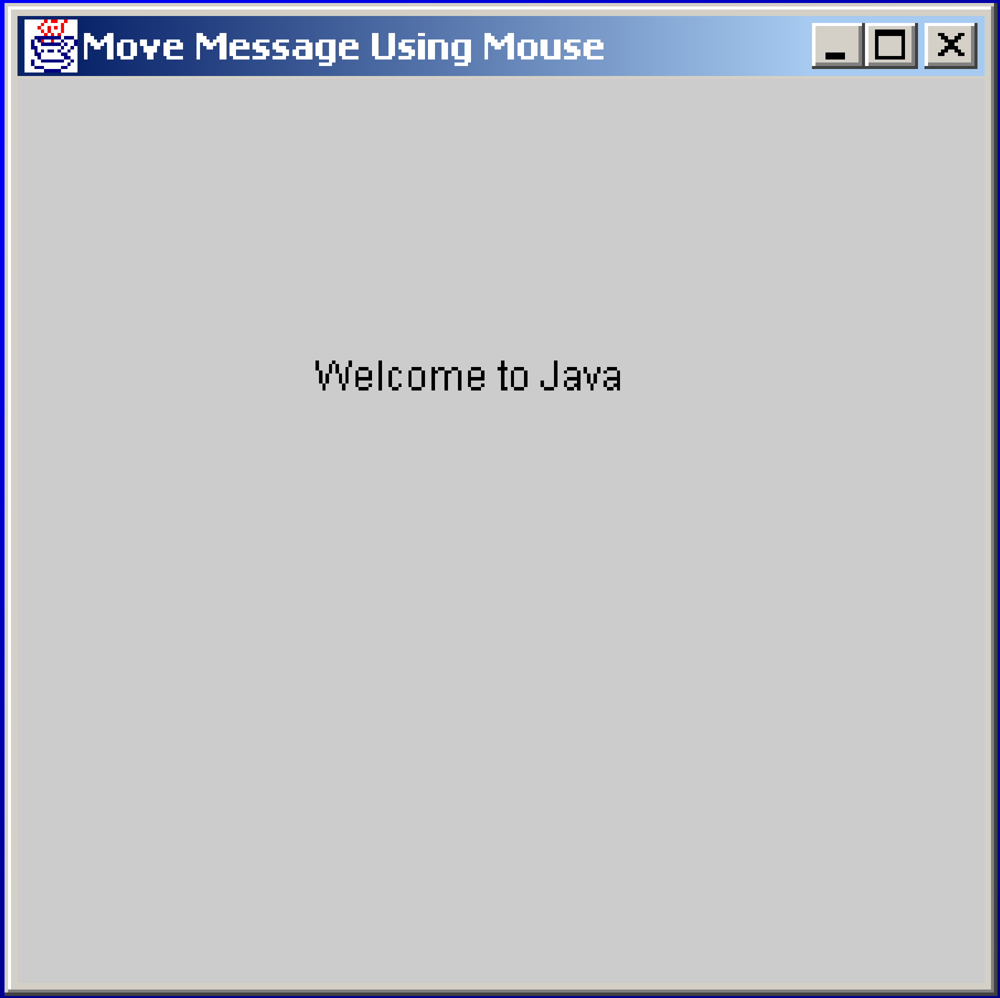
- Since MouseEvent is inherited from InputEvent class, you can use the following methods
  - public long getWhen(): indicating when the event occurred
  - public boolean isAltDown(): check whether the Alt key is down on the event
  - public boolean isControlDown: check whether the Control key is down on the event
  - public boolean isMetaDown(): check whether the right mouse button is pressed
  - Boolean isShiftDown(): check whether the Shift key is down on the event

Example 10.6
Moving Message Using Mouse

- Objective: Create a program to display a message in a panel. You can use the mouse to move the message. The message moves as the mouse drags and is always displayed at the mouse point.

```
// MoveMessageDemo.java: Move a message in a panel
// by dragging the mouse
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MoveMessageDemo extends JApplet
{
  // Initialize the applet
  public void init()
  {
    // Create a MoveMessagePanel instance for drawing a message
    MoveMessagePanel p =
          new MoveMessagePanel("Welcome to Java");
    // Place the message panel in the frame
    getContentPane().setLayout(new BorderLayout());
    getContentPane().add(p);
  }
```

```
// This main method enables the applet to run as an application
  public static void main(String[] args)
  {
    // Create a frame
    JFrame frame = new JFrame("Move Message Using Mouse");

    // Create an instance of the applet
    MoveMessageDemo applet = new MoveMessageDemo();

    // Add the applet instance to the frame
    frame.getContentPane().add(applet, BorderLayout.CENTER);

    // Invoke init() and start()
    applet.init();
    applet.start();
    // Display the frame
    frame.setSize(300, 300);
    // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }

  // MoveMessagePanel draws a message
  // This class is defined as inner class
  class MoveMessagePanel extends MessagePanel
                                          implements MouseMotionListener
  {
    // Construct a panel to draw string s
    public MoveMessagePanel(String s)
    {
      super(s);
      this.addMouseMotionListener(this);
    }
  // Tell the panel how to draw things
  public void paintComponent(Graphics g)
  {
    // Invoke the paintComponent method in the MessagePanel class
    super.paintComponent(g);
  }
  // Handler for mouse moved event
  public void mouseMoved(MouseEvent e)
  {    }
  // Handler for mouse dragged event
  public void mouseDragged(MouseEvent e) {
    // Get the new location and repaint the screen
    setXCoordinate(e.getX());
    setYCoordinate(e.getY());
    repaint();
  }
 }
}
```



Welcome to Java

Example 10.7
Handling Complex Mouse Events

- Objective: Create a program for drawing using a mouse. Draw by dragging with the left mouse button pressed; erase by dragging with the right button pressed.

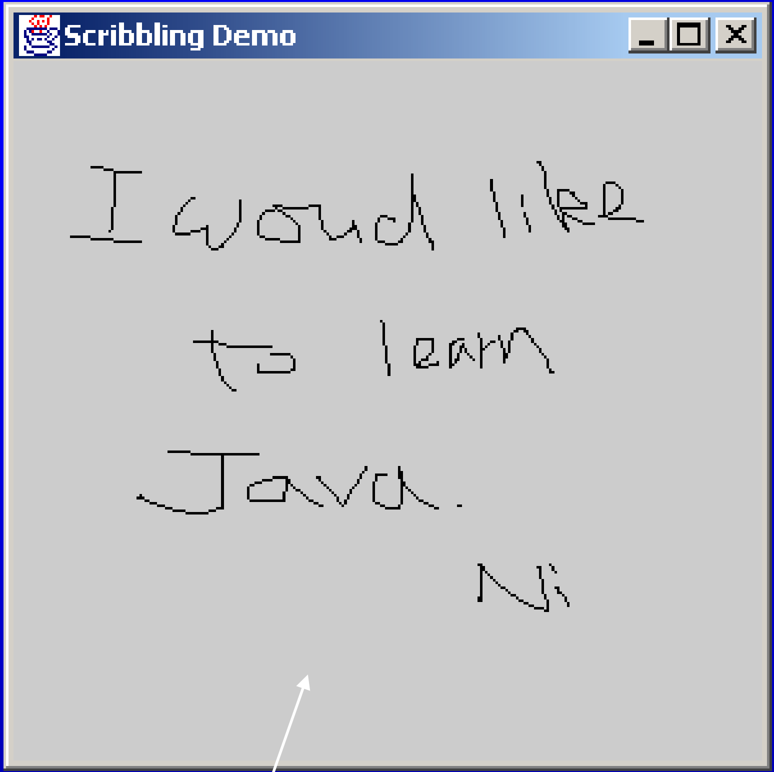
```
// ScribbleDemo.java: Scribble using mouse
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class ScribbleDemo extends JApplet
{
  // This main method enables the applet to run as an application
  public static void main(String[] args)
  {
    // Create a frame
    JFrame frame = new JFrame("Scribbling Demo");

    // Create an instance of the applet
    ScribbleDemo applet = new ScribbleDemo();
// Add the applet instance to the frame
  frame.getContentPane().add(applet, BorderLayout.CENTER);
  // Invoke init() and start()
  applet.init();
  applet.start();
  // Display the frame
  frame.setSize(300, 300);
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  frame.setVisible(true);
 }

 // Initialize the applet
 public void init()
 {
  // Create a PaintPanel and add it to the applet
  getContentPane().add(new ScribblePanel(), BorderLayout.CENTER);
 }
}
```

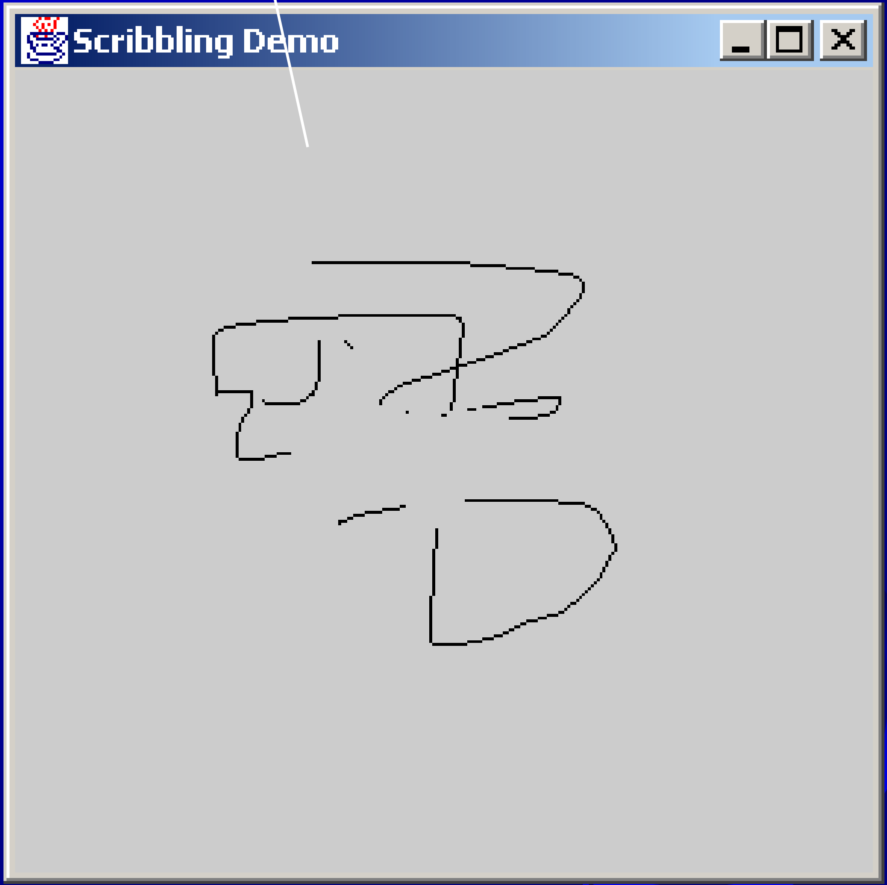
```
// ScribblePanel for scribbling using the mouse
class ScribblePanel extends JPanel
  implements MouseListener, MouseMotionListener
{
  final int CIRCLESIZE = 20; // Circle diameter used for erasing
  private Point lineStart = new Point(0, 0); // Line start point
  private Graphics g; // Create a Graphics object for drawing

  public ScribblePanel()
  {
    // Register listener for the mouse event
    addMouseListener(this);
    addMouseMotionListener(this);
  }

  public void mouseClicked(MouseEvent e)
  {
  }
```

```
public void mouseEntered(MouseEvent e)
{
}

public void mouseExited(MouseEvent e)
{
}

public void mouseReleased(MouseEvent e)
{
}

public void mousePressed(MouseEvent e)
{
 lineStart.move(e.getX(), e.getY());
}
```

```
public void mouseDragged(MouseEvent e)
{
  g = getGraphics(); // Get graphics context

  if (e.isMetaDown()) // Detect right button pressed
  {
    // Erase the drawing using an oval
    g.setColor(getBackground());
    g.fillOval(e.getX() - (CIRCLESIZE/2),
      e.getY() - (CIRCLESIZE/2), CIRCLESIZE, CIRCLESIZE);
  }
  else
  {
    g.setColor(Color.black);
    g.drawLine(lineStart.x, lineStart.y,
      e.getX(), e.getY());
  }
```

```
    lineStart.move(e.getX(), e.getY());

    // Dispose this graphics context
    g.dispose();
  }

  public void mouseMoved(MouseEvent e)
  {
  }
}
```





Right mouse button is used to erase.

Left mouse button is used to paint.

Handling Keyboard Events

To process a keyboard event, use the following handlers in the `KeyListener` interface:

- `keyPressed(KeyEvent e)`

  Called when a key is pressed.

- `keyReleased(KeyEvent e)`

  Called when a key is released.

- `keyTyped(KeyEvent e)`

  Called when a key is pressed and then released.

The `KeyEvent` Class

- Methods:

  `getKeyChar()` method: get the character of the key

  `getKeyCode()` method: get int value of the key

- Keys (Table 10.2):

Home	VK_HOME
End	VK_End
Page Up	VK_PGUP
Page Down	VK_PGDN
etc...	

Example 10.8
Keyboard Events Demo

- Objective: Display a user-input character. The user can also move the character up, down, left, and right using the arrow keys.

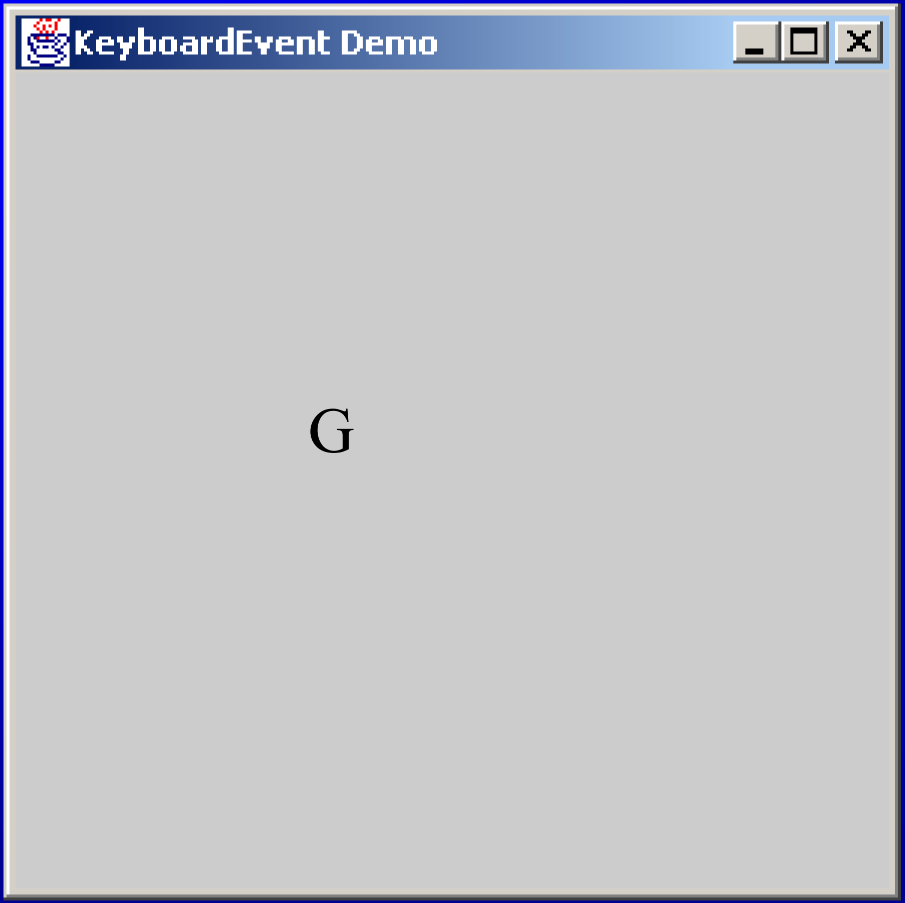
```
// KeyboardEventDemo.java: Receive key input
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class KeyboardEventDemo extends JApplet
{
  private KeyboardPanel keyboardPanel = new KeyboardPanel();

  // Main method used if run as an application
  public static void main(String[] args)
  {
    // Create a frame
    JFrame frame = new JFrame("KeyboardEvent Demo");

    // Create an instance of the applet
    KeyboardEventDemo applet = new KeyboardEventDemo();
  // Add the applet instance to the frame
  frame.getContentPane().add(applet, BorderLayout.CENTER);

  // Invoke init() and start()
  applet.init();
  applet.start();

  // Display the frame
  frame.setSize(300, 300);
  // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  frame.setVisible(true);

  // Set focus on the keyboardPanel
  applet.focus();
}
```

```
  // Initialize UI
  public void init()
  {
    // Add the keyboard panel to accept and display user input
    getContentPane().add(keyboardPanel);
    // Request focus
    focus();
  }

  // Set focus on the panel
  public void focus()
  {
    // It is required for receiving key input
    keyboardPanel.requestFocus();
  }
}
```

```
// KeyboardPanel for receiving key input
class KeyboardPanel extends JPanel implements KeyListener
{
  private int x = 100;
  private int y = 100;
  private char keyChar = 'A'; // Default key

  public KeyboardPanel()
  {
    addKeyListener(this); // Register listener
  }

  public void keyReleased(KeyEvent e)
  {
  }
```

```
public void keyTyped(KeyEvent e)
{
}

public void keyPressed(KeyEvent e)
{
  switch (e.getKeyCode())
  {
    case KeyEvent.VK_DOWN: y += 10; break;
    case KeyEvent.VK_UP: y -= 10; break;
    case KeyEvent.VK_LEFT: x -= 10; break;
    case KeyEvent.VK_RIGHT: x += 10; break;
    default: keyChar = e.getKeyChar();
  }
  repaint();
}
```

```
  // Draw the character
  public void paintComponent(Graphics g)
  {
    super.paintComponent(g);

    g.setFont(new Font("TimesRoman", Font.PLAIN, 24));
    g.drawString(String.valueOf(keyChar), x, y);
  }
}
```


KeyboardEvent Demo
G

Example 10.9
The TicTacToe Game

```
// TicTacToe.java: Play the TicTacToe game
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.LineBorder;

public class TicTacToe extends JApplet
{
  // Indicate which player has a turn, initially it is the X player
  private char whoseTurn = 'X';
```

```
// Create and initialize cells
  private Cell[][] cell =  new Cell[3][3];

  // Create and initialize a status label
  private JLabel jlblStatus = new JLabel("X's turn to play");

  // Initialize UI
  public void init()
  {
    // Panel p to hold cells
    JPanel p = new JPanel();
    p.setLayout(new GridLayout(3, 3, 0, 0));
    for (int i=0; i<3; i++)
      for (int j=0; j<3; j++)
        p.add(cell[i][j] = new Cell());
```

```
// Set line borders on the cells panel and the status label
   p.setBorder(new LineBorder(Color.red, 1));
   jlblStatus.setBorder(new LineBorder(Color.yellow, 1));

   // Place the panel and the label to the applet
   this.getContentPane().add(p, BorderLayout.CENTER);
   this.getContentPane().add(jlblStatus, BorderLayout.SOUTH);
 }

 // This main method enables the applet to run as an application
 public static void main(String[] args)
 {
   // Create a frame
   JFrame frame = new JFrame("Tic Tac Toe");

   // Create an instance of the applet
   TicTacToe applet = new TicTacToe();
```

```
  // Add the applet instance to the frame
  frame.getContentPane().add(applet, BorderLayout.CENTER);

  // Invoke init() and start()
  applet.init();
  applet.start();

  // Display the frame
  frame.setSize(300, 300);
  // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  frame.setVisible(true);
}
```

```
// Determine if the cells are all occupied
 public boolean isFull()
 {
   for (int i=0; i<3; i++)
     for (int j=0; j<3; j++)
       if (cell[i][j].getToken() == ' ')
         return false;

   return true;
 }
```

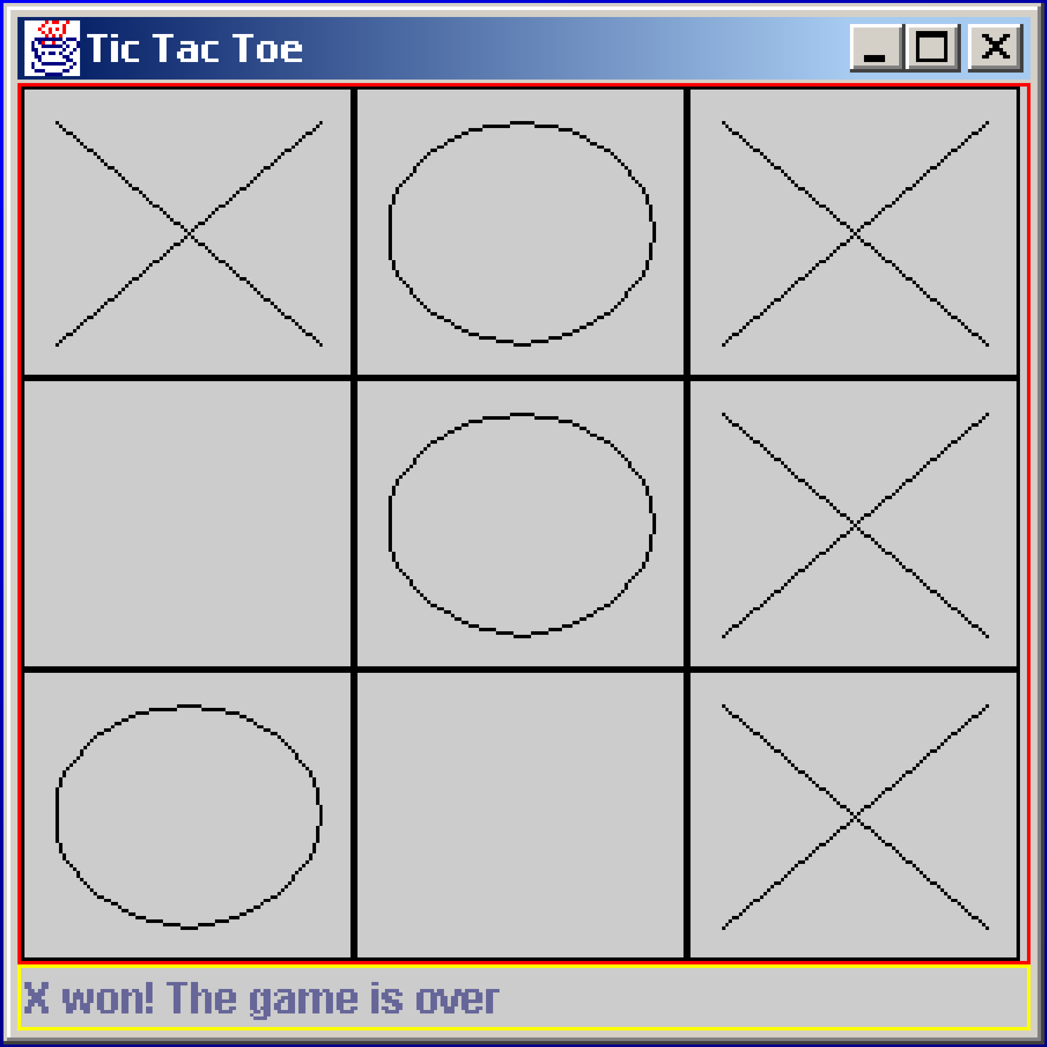
```
// Determine if the player with the specified token wins
 public boolean isWon(char token)
 {
   for (int i=0; i<3; i++)
     if ((cell[i][0].getToken() == token)
         && (cell[i][1].getToken() == token)
         && (cell[i][2].getToken() == token))
     {
       return true;
     }
   for (int j=0; j<3; j++)
     if ((cell[0][j].getToken() ==  token)
         && (cell[1][j].getToken() == token)
         && (cell[2][j].getToken() == token))
     {
       return true;
     }
```

```
  if ((cell[0][0].getToken() == token)
      && (cell[1][1].getToken() == token)
      && (cell[2][2].getToken() == token))
  {
    return true;
  }

  if ((cell[0][2].getToken() == token)
      && (cell[1][1].getToken() == token)
      && (cell[2][0].getToken() == token))
  {
    return true;
  }

  return false;
}
```

```
// An inner class for a cell
 public class Cell extends JPanel implements MouseListener
 {
   // Token used for this cell
   private char token = ' ';

   public Cell()
   {
     setBorder(new LineBorder(Color.black, 1)); // Set cell's border
     addMouseListener(this);  // Register listener
   }

   // The getter method for token
   public char getToken()
   {
     return token;
   }
```

```
// The setter method for token
    public void setToken(char c)
    {
      token = c;
      repaint();
    }

    // Paint the cell
    public void paintComponent(Graphics g)
    {
      super.paintComponent(g);

      if (token == 'X')
      {
        g.drawLine(10, 10, getSize().width-10, getSize().height-10);
        g.drawLine(getSize().width-10, 10, 10, getSize().height-10);
      }
```

```
  else if (token == 'O')
  {
    g.drawOval(10, 10, getSize().width-20, getSize().height-20);
  }
}

// Handle mouse click on a cell
public void mouseClicked(MouseEvent e)
{
  if (token == ' ') // If cell is not occupied
  {
    if (whoseTurn == 'X')  // If it is the X player's turn
    {
      setToken('X');  // Set token in the cell
      whoseTurn = 'O';  // Change the turn
jlblStatus.setText("O's turn");  // Display status
        if (isWon('X'))
          jlblStatus.setText("X won! The game is over");
        else if (isFull())
          jlblStatus.setText("Draw! The game is over");
      }
      else if (whoseTurn == 'O') // If it is the O player's turn
      {
        setToken('O'); // Set token in the cell
        whoseTurn = 'X';  // Change the turn
        jlblStatus.setText("X's turn"); // Display status
        if (isWon('O'))
          jlblStatus.setText("O won! The game is over");
        else if (isFull())
          jlblStatus.setText("Draw! The game is over");
      }
    }
  }
  public void mousePressed(MouseEvent e)
   {
    // TODO: implement this java.awt.event.MouseListener method;
   }
   public void mouseReleased(MouseEvent e)
   {
    // TODO: implement this java.awt.event.MouseListener method;
   }
   public void mouseEntered(MouseEvent e)
   {
    // TODO: implement this java.awt.event.MouseListener method;
   }
   public void mouseExited(MouseEvent e)
   {
    // TODO: implement this java.awt.event.MouseListener method;
   }
 }
}
```



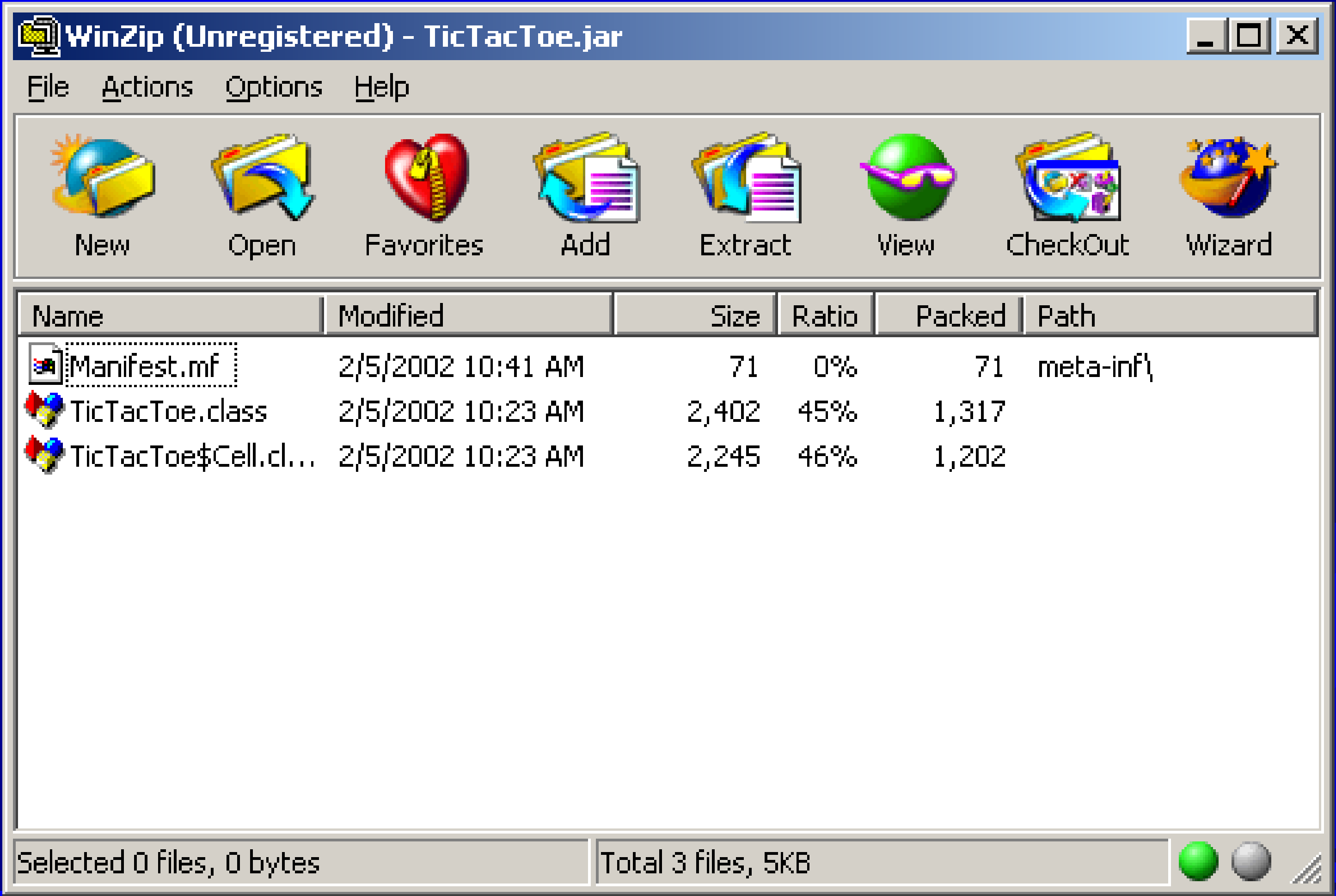
X won! The game is over

Package and Deploying Java Projects(Optional)

- Each Project may consist of many classes and supporting files.
- Need to provide end-user with all these files
- Archive file which can be used to group all the project files in a compressed file, called jar file
- It is based on ZIP file
- It can be deployed on an end-user's cmputer
- Jar command can be used to crate an archive file

```
jar –cf TicTacToc.jar TicTacToe.class TicTacToe$Cell.class
```

Package and Deploying Java Projects(Optional)



Name	Modified	Size	Ratio	Packed	Path
Manifest.mf	2/5/2002 10:41 AM	71	0%	71	meta-inf\
TicTacToe.class	2/5/2002 10:23 AM	2,402	45%	1,317	
TicTacToe$Cell.cl...	2/5/2002 10:23 AM	2,245	46%	1,202	

Selected 0 files, 0 bytes

Total 3 files, 5KB

Package and Deploying Java Projects(Optional)

- After compressed the file, it contains a file called Manifest.mf
- To run TicTacToe.jar file as an application, you need the following steps
  - Create a text file and save it as temp.mf
  - In the temp.mf, type in the following two lines

```
Main-Class: TicTacToe
Sealed: true
```

Package and Deploying Java Projects(Optional)

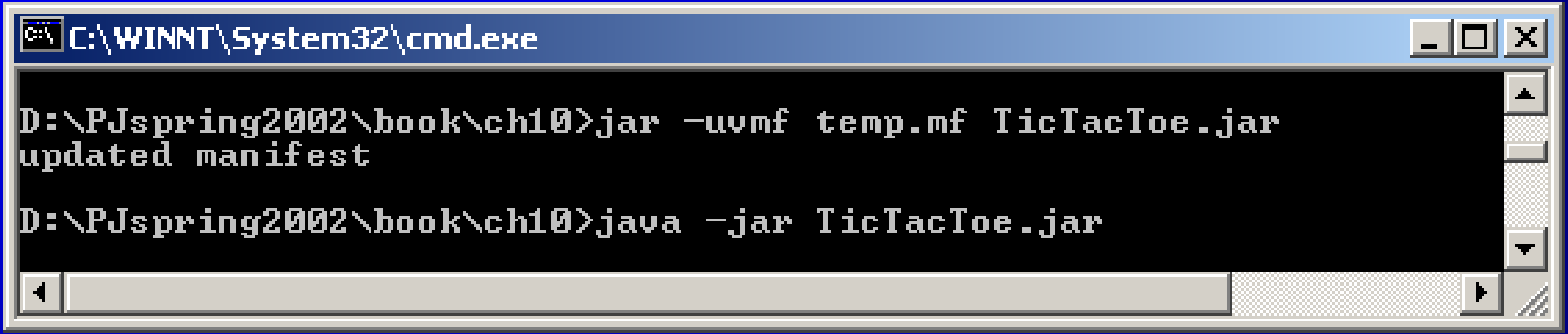
- Update Manifest.mf file, you need to use jar command as

```
jar –uvmf temp.mf TicTacToe.jar
```

- Now you can run the application, using

```
java –jar TicTacToe.jar
```

Package and Deploying Java Projects(Optional)



C:\WINNT\System32\cmd.exe

```
D:\PJspring2002\book\ch10>jar -uvmf temp.mf TicTacToe.jar
updated manifest

D:\PJspring2002\book\ch10>java -jar TicTacToe.jar
```

Package and Deploying Java Projects(Optional)

- To run TicTacToe as an applet, you need to modify the html file

```
<APPLET
CODE = "TicTacToe.class"
ARCHIVE="TicTacToe.jar"
WIDTH=400
HEIGHT=300
HSPACE=0
VSPACE=0
ALIGN=Middle>
</APPLET>
```

Advanced Layout

(Optional from here on)

- CardLayout
- GridBagLayout
- Using No Layout Manager

`CardLayout` (Optional from here on)

- The `CardLayout` manager arranges components in a queue of cards. You can only see one card at a time. To construct a `CardLayout`, simply use the constructor `CardLayout()`.

- To add a component in the `CardLayout` container, use the following method:

```
void add(Component com,
  String name)
```

`CardLayout` View Components

- `void first(container)`
- `void last(container)`
- `void next(container)`
- `void previous(container)`
- `void show(Container, String name)`

Example 10.10
Testing `CardLayout` Manager

- Objective: Create two panels in a frame. The first panel holds named components. The second panel uses buttons and a choice box to control which component is shown.

```java
// ShowCardLayout.java: Using CardLayout to display images
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ShowCardLayout extends JApplet
  implements ActionListener, ItemListener
{
  private CardLayout queue = new CardLayout();
  private JPanel cardPanel = new JPanel();
  private JButton jbtFirst, jbtNext, jbtPrevious, jbtLast;
  private JComboBox jcboImage;

  public void init()
  {
    // Use CardLayout for cardPanel
    cardPanel.setLayout(queue);
```

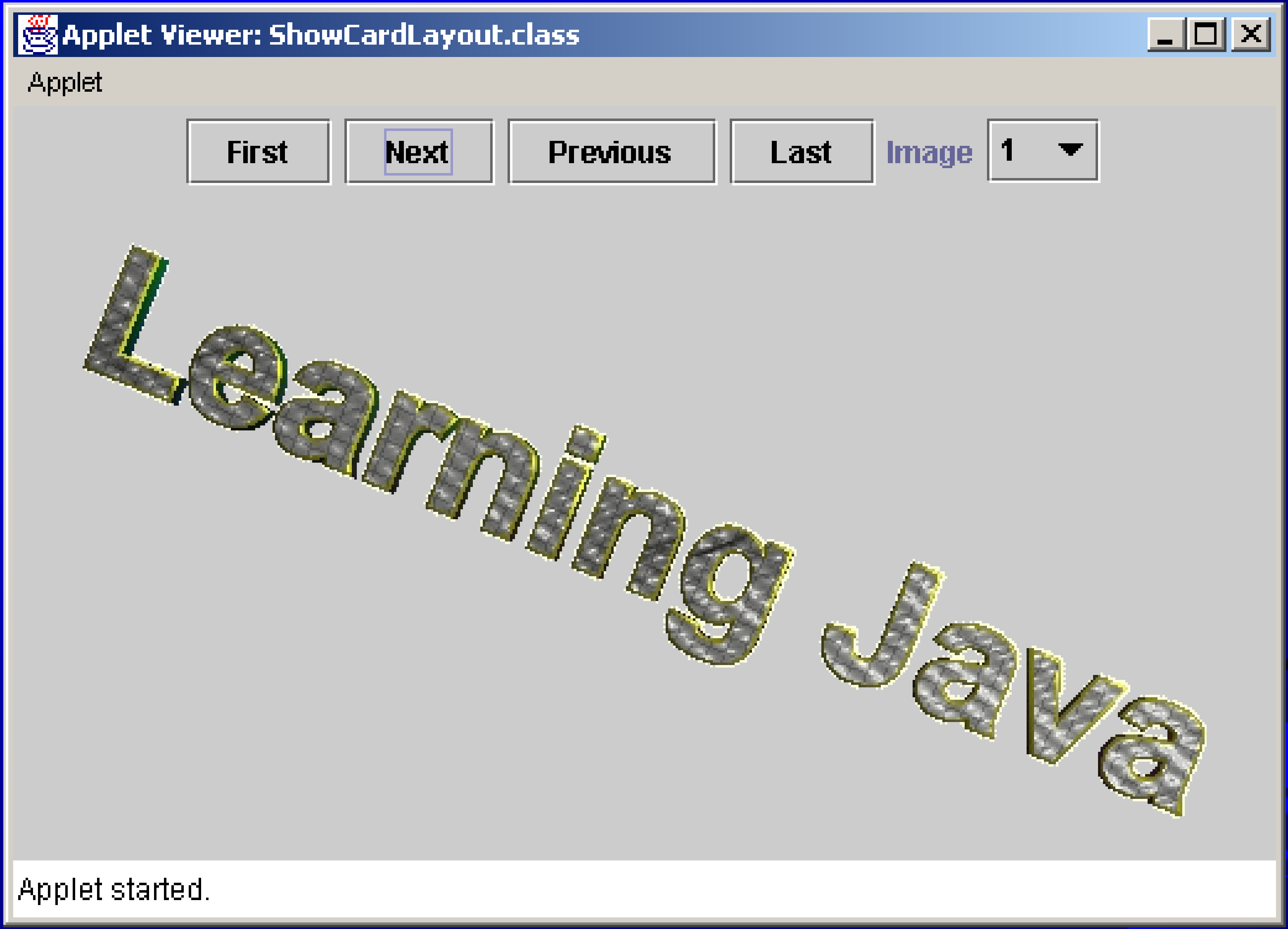
```
// Add 15 labels for displaying images into cardPanel
for (int i=1; i<=15; i++)
  cardPanel.add
    (new JLabel(new ImageIcon("images/L"+i+".gif")),
      String.valueOf(i));

// Panel p to hold buttons and a combo box
JPanel p = new JPanel();
p.add(jbtFirst = new JButton("First"));
p.add(jbtNext = new JButton("Next"));
p.add(jbtPrevious= new JButton("Previous"));
p.add(jbtLast = new JButton("Last"));
p.add(new JLabel("Image"));
p.add(jcboImage = new JComboBox());
```

```
// Initialize combo box items
   for (int i=1; i<=15; i++)
     jcboImage.addItem(String.valueOf(i));

   // Place panels in the frame
   getContentPane().add(cardPanel, BorderLayout.CENTER);
   getContentPane().add(p, BorderLayout.NORTH);

   // Register listeners with the source objects
   jbtFirst.addActionListener(this);
   jbtNext.addActionListener(this);
   jbtPrevious.addActionListener(this);
   jbtLast.addActionListener(this);
   jcboImage.addItemListener(this);
 }
```

```
// This main method enables the applet to run as an application
 public static void main(String[] args)
 {
   // Create a frame
   JFrame frame = new JFrame("CardLayout Demo");

   // Create an instance of the applet
   ShowCardLayout applet = new ShowCardLayout();

   // Add the applet instance to the frame
   frame.getContentPane().add(applet, BorderLayout.CENTER);

   // Invoke init() and start()
   applet.init();
   applet.start();
```

```
// Display the frame
   frame.setSize(300, 300);
   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.setVisible(true);
 }

 // Handle button actions
 public void actionPerformed(ActionEvent e)
 {
   String actionCommand = e.getActionCommand();
   if (e.getSource() instanceof JButton)
     if ("First".equals(actionCommand))
       // Show the first component in queue
       queue.first(cardPanel);
     else if ("Last".equals(actionCommand))
       // Show the last component in queue
       queue.last(cardPanel);
```

```
else if ("Previous".equals(actionCommand))
      // Show the previous component in queue
      queue.previous(cardPanel);
    else if ("Next".equals(actionCommand))
      // Show the next component in queue
      queue.next(cardPanel);
 }

 // Handle selection of combo box item
 public void itemStateChanged(ItemEvent e)
 {
   if (e.getSource() == jcboImage)
     // Show the component at specified index
     queue.show(cardPanel, (String)e.getItem());
 }
}
```



Applet started.

GridBagLayout

The `GridBagLayout` manager is the most flexible and the most complex. It is similar to the `GridLayout` manager in the sense that both layout managers arrange components in a grid. The components can vary in size, however, and can be added in any order in `GridBagLayout`.

Example 10.11: Using GridBagLayout Manager

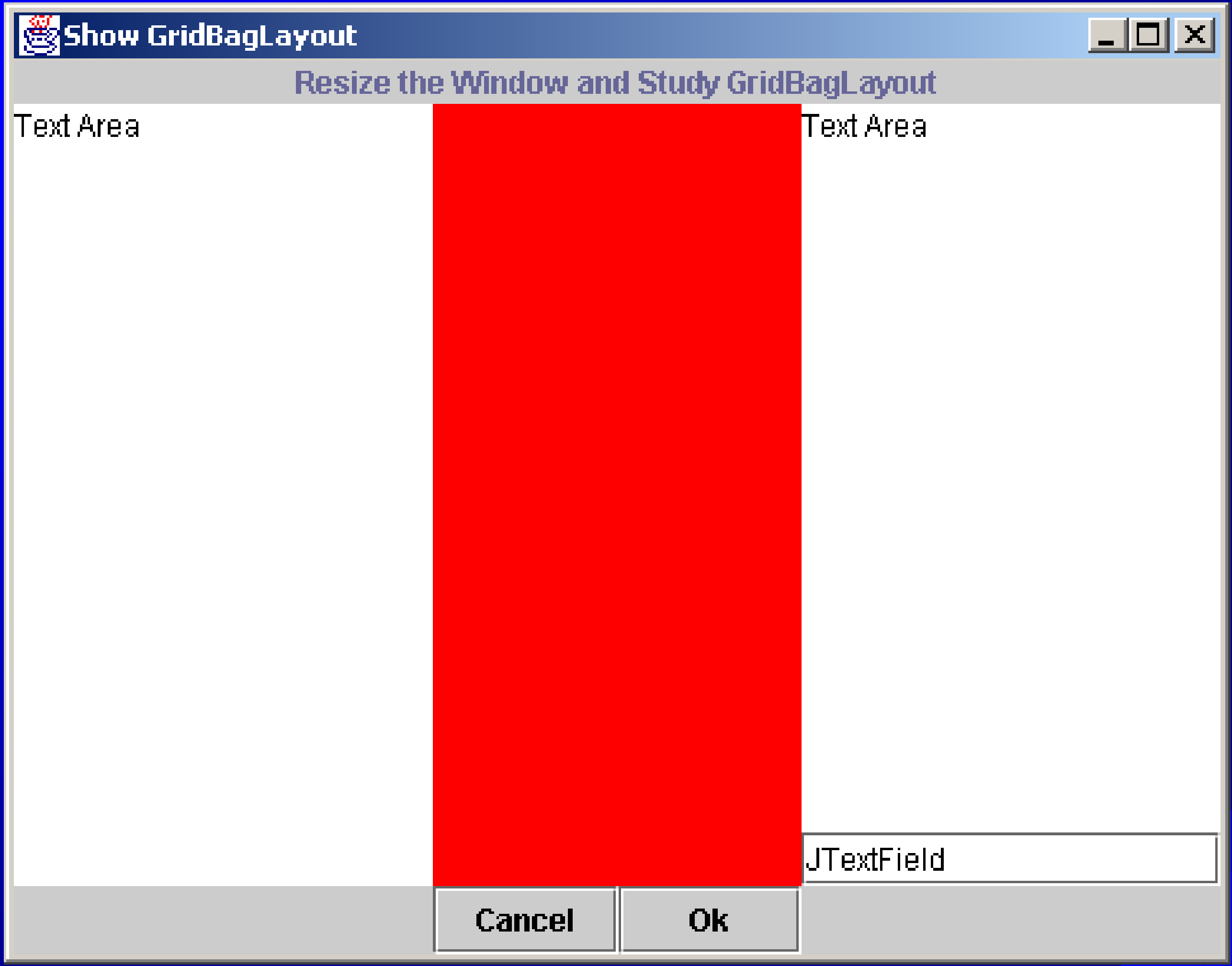
```
// ShowGridBagLayout.java: Using GridBagLayout
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ShowGridBagLayout extends JFrame
{
  private JLabel jlbl;
  private JTextArea jta1, jta2;
  private JTextField jtf;
  private JPanel jp;
  private JButton jbt1, jbt2;
  private GridBagLayout gbLayout;
  private GridBagConstraints gbConstraints;
```

```
// Main method
 public static void main(String[] args)
 {
  ShowGridBagLayout frame = new ShowGridBagLayout();
  frame.setSize(350,200);
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  frame.setVisible(true);
 }

 // Add a component to the conjtainer
 private void addComp(Component c, GridBagLayout gbLayout,
                      GridBagConstraints gbConstraints,
                      int row, int column, int numRows,
                      int numColumns, int weightx, int weighty)
 {
  // Set parameters
  gbConstraints.gridx = column;
gbConstraints.gridy = row;
  gbConstraints.gridwidth = numColumns;
  gbConstraints.gridheight = numRows;
  gbConstraints.weightx = weightx;
  gbConstraints.weighty = weighty;

  // Set constraints in the GridBagLayout
  gbLayout.setConstraints(c, gbConstraints);

  // Add component to the container
  getContentPane().add(c);
 }

 // Constructor
 public ShowGridBagLayout()
 {
  setTitle("Show GridBagLayout");
```

```
// Initialize UI components
   jlbl = new JLabel("Resize the Window and Study GridBagLayout",
                  JLabel.CENTER);
   jp = new JPanel();
   jta1 = new JTextArea("Text Area", 5, 15 );
   jta2 = new JTextArea("Text Area", 5, 15 );
   jtf = new JTextField("JTextField");
   jbt1 = new JButton("Cancel" );
   jbt2 = new JButton("Ok" );

   // Create GridBagLayout and GridBagConstraints object
   gbLayout = new GridBagLayout();
   gbConstraints = new GridBagConstraints();
   getContentPane().setLayout(gbLayout);
```

```
// Place JLabel to occupy row 0 (the first row)
   gbConstraints.fill = GridBagConstraints.BOTH;
   gbConstraints.anchor = GridBagConstraints.CENTER;
   addComp(jlbl, gbLayout, gbConstraints, 0, 0, 1, 4, 0, 0);

   // Place text area 1 in row 1 and 2, and column 0
   addComp(jta1, gbLayout, gbConstraints, 1, 0, 2, 1, 0, 0);

   // Place Panel in row 1 and 2, and column 1 and 2
   addComp(jp, gbLayout, gbConstraints, 1, 1, 2, 2, 100, 100);
   jp.setBackground(Color.red);

   // Place text area 2 in row 1 and column 3
   addComp(jta2, gbLayout, gbConstraints, 1, 3, 1, 1, 0, 100);

   // Place text field in row 2 and column 3
   addComp(jtf, gbLayout, gbConstraints, 2, 3, 1, 1, 0, 0);
```

```
    // Place JButton 1 in row 3 and column 1
    addComp(jbt1, gbLayout, gbConstraints, 3, 1, 1, 1, 0, 0);

    // Place JButton 2 in row 3 and column 2
    addComp(jbt2, gbLayout, gbConstraints, 3, 2, 1, 1, 0, 0);
  }
}
```


Show GridBagLayout
Resize the Window and Study GridBagLayout
Text Area
Text Area
JTextField
Cancel
Ok

Using No Layout Manager

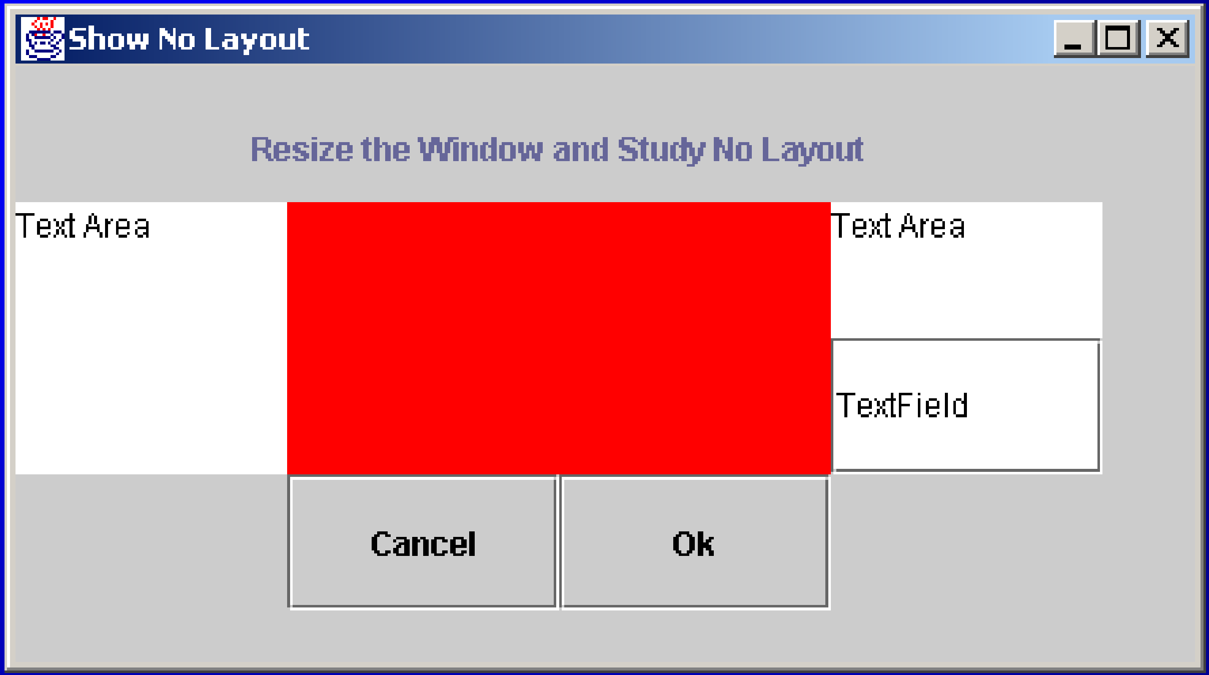
You can place components in a container without using any layout manager. In this case, the component must be placed using the component's instance method `setBounds()`.

Example 10.12: Using No Layout Manager

```
// ShowNoLayout.java: Place components
//   without using a layout manager
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ShowNoLayout extends JFrame
{
  private JLabel jlbl =
    new JLabel("Resize the Window and Study No Layout",
      JLabel.CENTER);;
  private JTextArea jta1 = new JTextArea("Text Area", 5, 10 );
  private JTextArea jta2 = new JTextArea("Text Area", 5, 10 );
  private JTextField jtf = new JTextField("TextField");
  private JPanel jp = new JPanel();
  private JButton jbt1 = new JButton("Cancel" );
  private JButton jbt2 = new JButton("Ok" );
private GridBagLayout gbLayout;
 private GridBagConstraints gbConstraints;

 public static void main(String[] args)
 {
   ShowNoLayout frame = new ShowNoLayout();
   frame.setSize(400,200);
   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.setVisible(true);
 }

 public ShowNoLayout()
 {
   setTitle("Show No Layout");

   // Set background color for the panel
   jp.setBackground(Color.red);
```

```
// Specify no layout manager
   getContentPane().setLayout(null);

   // Add components to frame
   getContentPane().add(jlbl);
   getContentPane().add(jp);
   getContentPane().add(jta1);
   getContentPane().add(jta2);
   getContentPane().add(jtf);
   getContentPane().add(jbt1);
   getContentPane().add(jbt2);
```

```
// Put components in the right place
   jlbl.setBounds(0, 10, 400, 40);
   jta1.setBounds(0, 50, 100, 100);
   jp.setBounds(100, 50, 200, 100);
   jta2.setBounds(300, 50, 100, 50);
   jtf.setBounds(300, 100, 100, 50);
   jbt1.setBounds(100, 150, 100, 50);
   jbt2.setBounds(200, 150, 100, 50);
  }
}
```


Show No Layout
Resize the Window and Study No Layout
Text Area
Text Area
TextField
Cancel
Ok