Unit 6
APPLET

Introduction

- Applet are small applications that are accessed on an Internet server, transported over the Internet, automatically installed, and run as part of a web document.
- After an applet arrives on the client, it has limited access to resources so that it can produce a graphical user interface and run various computations without introducing the risk of viruses or breaching data integrity.

```
import java.awt.*;
import java.applet.*;
public class SimpleApplet extends Applet{
    public void paint(Graphics g){
            g.drawString(“A simple Applet”,20,20);
    }
    }
```

- After you enter the source code for SimpleApplet, compile in the same way that you have been compiling programs.
- However, running SimpleApplet involves a different process. In fact, there are two ways in which you can run an applet:
  - Executing the applet within a Java-compatible web browser.
  - Using an applet viewer, such as the standard tool, appletviewer.
- An applet viewer executes your applet in a window. This is generally the fastest and easiest way to test your applet.
- One way to execute an applet in a web browser is to write a short HTML text file that contains a tag that loads the applet.
- At the time of this writing, Oracle recommends using

- However, a more convenient method exists that you can use to speed up testing.
- Simply include a comment at the head of your Java source code file that contains the APPLET tag.
- By doing so, your code is documented with a prototype of the necessary HTML statements, and you can test your compiled applet merely by starting the applet viewer with your Java source code file.
- With this approach, you can quickly iterate through applet development by using these three steps:
  - 1. Edit a Java source file.
  - 2. Compile your program.
  - 3. Execute the applet viewer, specifying the name of your applet's source file.
- The applet viewer will encounter the APPLET tag within the comment and execute your applet.

Few things of Applets

- Applets do not need a main( ) method.
- Applets must be run under an applet viewer or a Java-compatible browser.
- Applets use the interface provided by a GUI framework.

Two Types of Applets

- It is important to state at the outset that there are two varieties of applets based on Applet.
- The first are those based directly on the Applet class
- These applets use the Abstract Window Toolkit (AWT) to provide the graphical user interface (or use no GUI at all).
- This style of applet has been available since Java was first created.
- The second type of applets are those based on the Swing class JApplet, which inherits Applet.
- Swing applets use the Swing classes to provide the GUI.
- Swing offers a richer and often easier-to-use user interface than does the AWT.

The Applet Class

- Applet provides all necessary support for applet execution, such as starting and stopping.
- It also provides methods that load and display images, and methods that load and play audio clips.
- Applet extends the AWT class Panel.
- In turn, Panel extends Container, which extends Component.
- These classes provide support for Java’s window-based, graphical interface.
- Thus, Applet provides all of the necessary support for window-based activities.

Applet Architecture

- An applet waits until an event occurs.
- The runtime system notifies the applet about an event by calling an event handler that has been provided by the applet.
- Once this happens, the applet must take appropriate action and then quickly return.
- Instead, the user interacts with the applet as he or she wants, when he or she wants.
- These interactions are sent to the applet as events to which the applet must respond.
- For example, when the user clicks the mouse inside the applet’s window, a mouse-clicked event is generated.
- If the user presses a key while the applet’s window has input focus, a keypress event is generated.
- When the user interacts with one of these controls, an

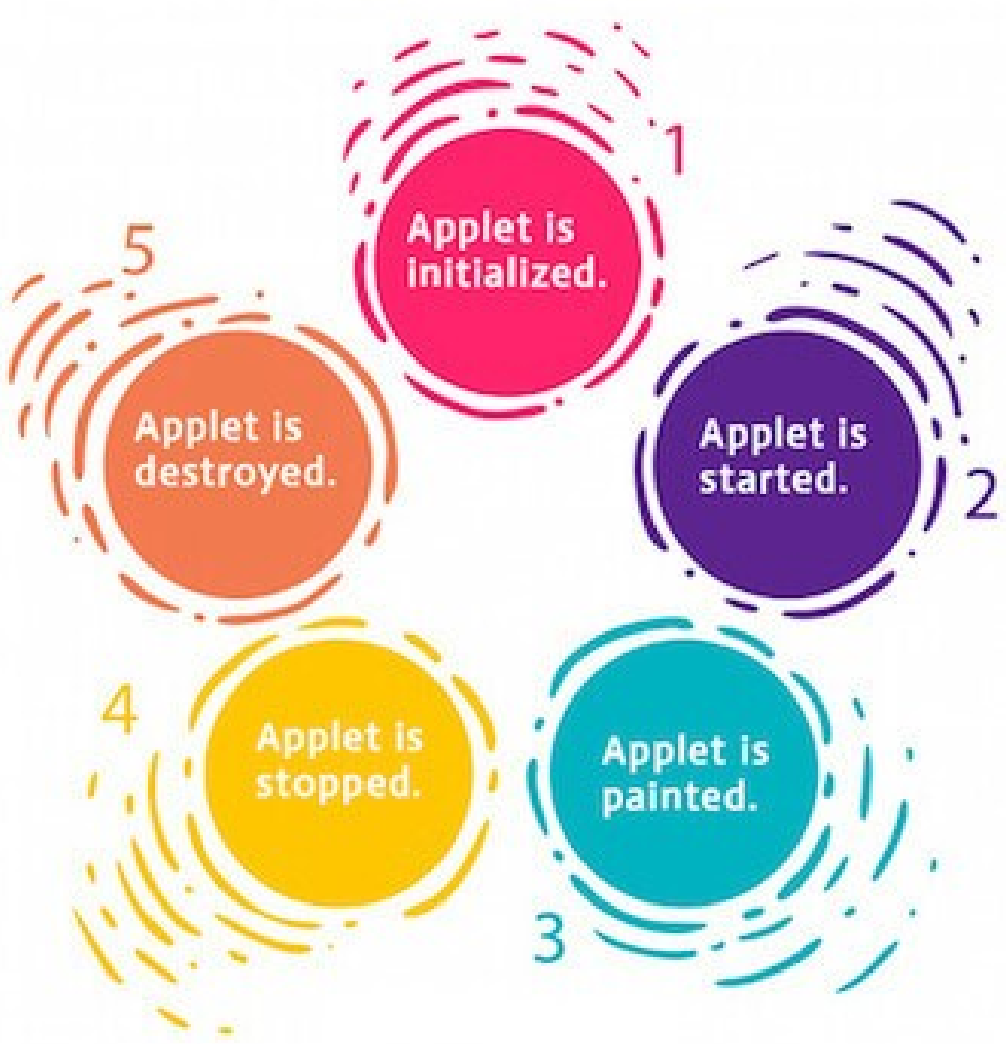
An Applet Skeleton

- All but the most trivial applets override a set of methods that provides the basic mechanism by which the browser or applet viewer interfaces to the applet and controls its execution.
- Four of these methods, init( ), start( ), stop( ), and destroy( ), apply to all applets and are defined by Applet.
- Default implementations for all of these methods are provided.
- Applets do not need to override those methods they do not use.
- However, only very simple applets will not need to define all of them.
- AWT-based will also often override the paint( ) method, which is defined by the AWT Component class.

```
import java.awt.*;
import java.applet.*;
/*
<applet code="AppletSkel" width=300 height=300>
</applet>
*/
public class AppletSkel extends Applet{
        public void init(){
                //initialiazation
        }
        public void start(){
                //start or resume execution
        }
```

```
    public void paint(){
        //redisplay contents of window
    }
    public void stop(){
        //suspends execution
    }
    public void destroy(){
        //perform shutdown activities
    }
}
```

Applet Lifecycle



- Applet Initialization and Termination It is important to understand the order in which the various methods shown in the skeleton are called.
- When an applet begins, the following methods are called, in this sequence:
- 1. init( )
- 2. start( )
- 3. paint( )
- When an applet is terminated, the following sequence of method calls takes place:
- 1. stop( )
- 2. destroy( )

- init( )
  - The init( ) method is the first method to be called.
  - This is where you should initialize variables.
  - This method is called only once during the run time of your applet.
- start( )
  - The start( ) method is called after init( ).
  - It is also called to restart an applet after it has been stopped. Whereas init( ) is called once—the first time an applet is loaded—start( ) is called each time an applet's HTML document is displayed onscreen.
  - So, if a user leaves a web page and comes back, the applet resumes execution at start( ).

- paint( )
  - The paint( ) method is called each time an AWT-based applet’s output must be redrawn.
  - This situation can occur for several reasons. For example, the window in which the applet is running may be overwritten by another window and then uncovered.
  - Or the applet window may be minimized and then restored. paint( ) is also called when the applet begins execution.
  - Whatever the cause, whenever the applet must redraw its output, paint( ) is called.
  - The paint( ) method has one parameter of type Graphics. This parameter will contain the graphics context, which describes the graphics environment in which the applet is running. This context is used whenever output to the applet is required.

- stop( )
  - The stop( ) method is called when a web browser leaves the HTML document containing the applet—when it goes to another page, for example.
  - When stop( ) is called, the applet is probably running. You should use stop( ) to suspend threads that don’t need to run when the applet is not visible. You can restart them when start( ) is called if the user returns to the page.
- destroy( )
  - The destroy( ) method is called when the environment determines that your applet needs to be removed completely from memory.
  - At this point, you should free up any resources the applet may be using. The stop( ) method is

The HTML Applet Tag

- As mentioned earlier, at the time of this writing, Oracle recommends that the APPLET tag be used to manually start an applet when Java Network Launch Protocol is not used.
- An applet viewer will execute each APPLET tag that it finds in a separate window, while web browsers will allow many applets on a single page.
- So far, we have been using only a simplified form of the APPLET tag.
- Now it is time to take a closer look at it.

- <APPLET
- [CODEBASE =codebaseURL]
- CODE=A URL that points to the class of the applet
- [ALT=Alternate text to be displayed in case browser does not support applet]
- [NAME=Defines a unique name for the applet]
- WIDTH=pixels HEIGHT=pixels
- [ALIGN=alignment] deprecated
- [VSPACE=pixels][HSPACE=pixels] Amount of white space to be inserted above and below the object. deprecated
- [<PARAM NAME=AttributeName VALUE=AttributeValue>]
- [<PARAM NAME=AttributeName VALUE=AttributeValue>]
- .....
- [HTML displayed in the absence of java]
- [/APPLET>

Passing Parameters to Applets

- As just discussed, the APPLET tag allows you to pass parameters to your applet.
- To retrieve a parameter, use the getParameter( ) method. It returns the value of the specified parameter in the form of a String object.
- Thus, for numeric and boolean values, you will need to convert their string representations into their internal formats.
- Here is an example that demonstrates passing parameters:

Example 1

```
import java.applet.Applet;
import java.awt.Graphics;
/*
<applet code="UseParam.class" width="300"
height="300">
<param name="msg" value="Welcome to applet">

</applet>
*/
public class UseParam extends Applet {
    public void paint(Graphics g) {
        String str = getParameter("msg");
        g.drawString(str, 50, 50);
    }
```

Example 2

```
import java.applet.*;
import java.awt.*;
/*
 <applet code="ParamDemo.class" width="300" height="100">
 <param name="fontName" value="Courier">
 <param name="fontSize" value="14">
 <param name="Leading" value="2">
 <param name="accountEnabled" value="true">
 </applet>
 */
public class ParamDemo extends Applet {

    String fontName;
    int fontSize;
    float leading;
    boolean active;
```

```
 //Initialize the string to be displayed
    public void start() {
        fontName = getParameter("fontName");
        fontSize =Integer.parseInt(getParameter("fontSize"));
        leading = Float.parseFloat(getParameter("leading"));
        active =
Boolean.parseBoolean(getParameter("accountEnabled"));
    }

//Display parameters
    public void paint(Graphics g) {
        g.drawString("Font name:" + fontName, 0, 10);
        g.drawString("Font size:" + fontSize, 0, 26);
        g.drawString("Leading:" + leading, 0, 42);
        g.drawString("Account Active:" + active, 0, 58);
    }
}
```

Practices

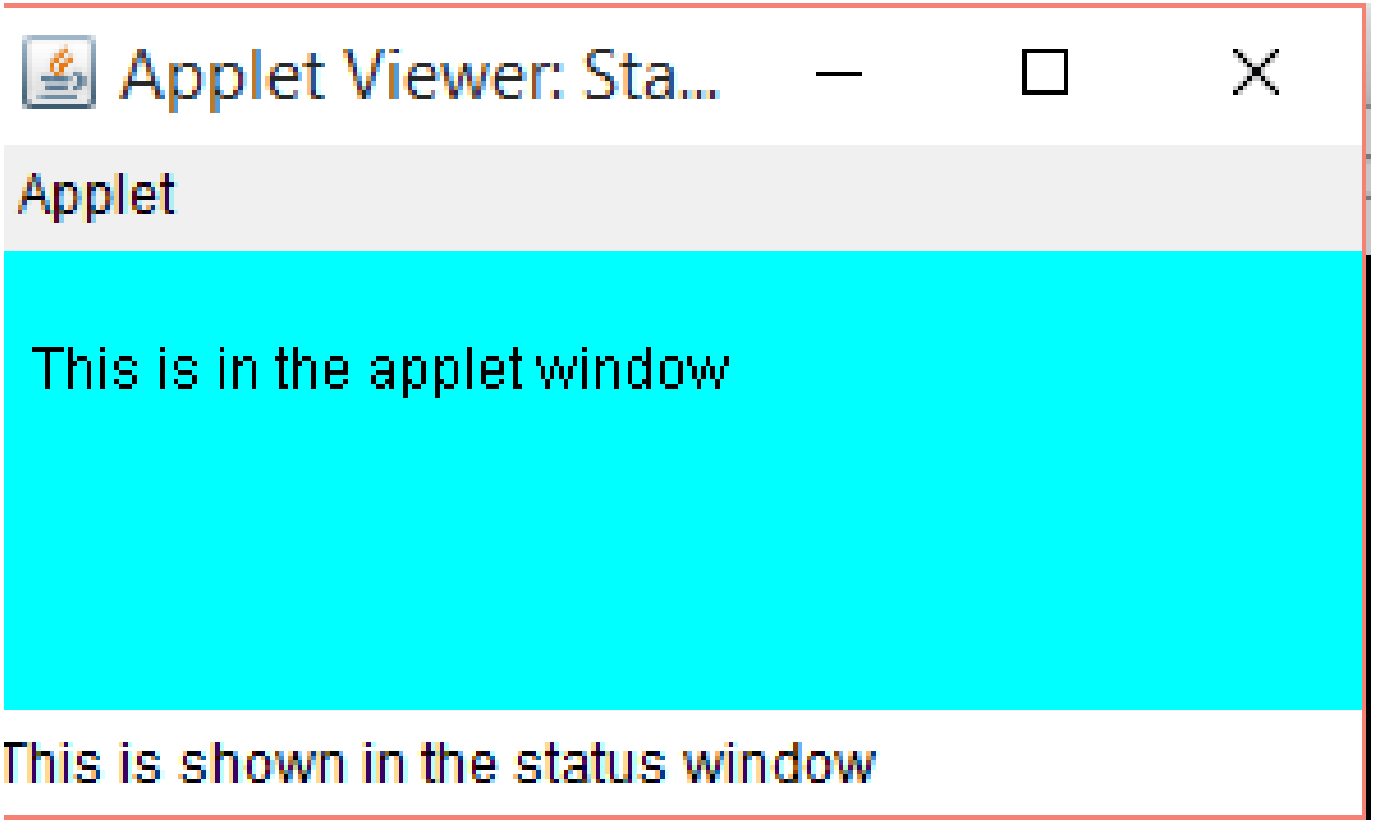
```
import java.awt.*;
import java.applet.*;
/*
<applet code="Sample" width=300 height=100>
</applet>
*/
public class Sample extends Applet{
        String msg;
        //set the  foregorund and background colors
        public void init(){
                setBackground(Color.cyan);
                setForeground(Color.red);
                msg="Inside init()--";

        }
```

```
public void start(){
        msg="Inside start()--";
    }
    public void paint(Graphics g){
        msg="Inside paint()";
        g.drawString(msg,10,30);
        }
}
```

Using the Status Window

```
import java.awt.*;
import java.applet.*;
/*
<applet code="StatusWindow" width=300 height=100>
</applet>
*/
public class StatusWindow extends Applet{
        public void init(){
                setBackground(Color.cyan);
        }
        //Display msg in applet window
        public void paint(Graphics g){
                g.drawString("This is in the applet
window",10,30);
                showStatus("This is shown in the status
window");
                }
```


Applet Viewer: Sta...
Applet
This is in the applet window
This is shown in the status window

Graphics

Drawing Lines

- Lines are drawn by means of the drawLine( ) method, shown here:
  - void drawLine(int startX, int startY, int endX, int endY)
- drawLine( ) displays a line in the current drawing color that begins at startX, startY and ends at endX, endY.

Drawing Rectangles

- The drawRect( ) and fillRect( ) methods display an outlined and filled rectangle, respectively. They are shown here:
  - void drawRect(int x, int y, int width, int height)
  - void fillRect(int x, int y, int width, int height)
  - The upper-left corner of the rectangle is at x, y. The dimensions of the rectangle are specified by width and height.

- To draw a rounded rectangle, use drawRoundRect( ) or fillRoundRect( ), both shown here:
  - void drawRoundRect(int x,int y,int width,int height,int xDiam,int yDiam)
  - void fillRoundRect(int x,int y,int width,int height,int xDiam,int yDiam)

Drawing Ellipses and Circles

- To draw an ellipse, use drawOval( ). To fill an ellipse, use fillOval( ). These methods are shown here:
  - void drawOval(int x, int y, int width, int height)
  - void fillOval(int x, int y, int width, int height)

Drawing Arcs

- Arcs can be drawn with drawArc( ) and fillArc( ), shown here:
  - void drawArc(int x,int y,int width,int height,int startAngle,int sweepAngle)
  - void fillArc(int x,int y,int width,int height,int startAngle,int sweepAngle)

Graphics Demo

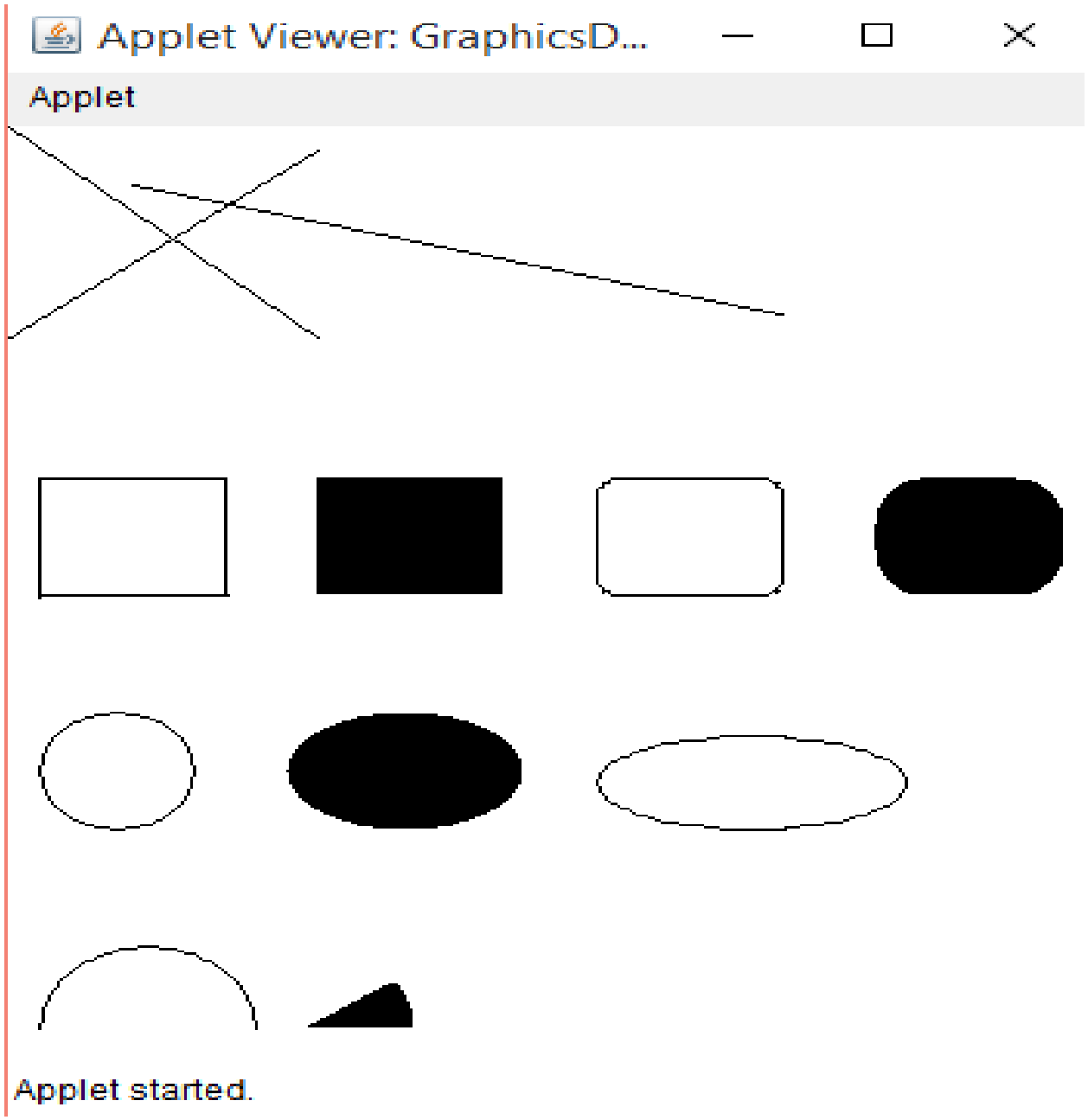
```
import java.applet.*;
import java.awt.*;
/*
<applet code="GraphicsDemo.class" width=350
height=400>
</applet>
*/
public class  GraphicsDemo extends Applet{
  public void paint(Graphics g){
          //draw lines
      g.drawLine(0,0,100,90);
      g.drawLine(0,90,100,10);
      g.drawLine(40,25,250,80);
```

```
//Draw Rectangles
      g.drawRect(10,150,60,50);
      g.fillRect(100,150,60,50);
      g.drawRoundRect(190,150,60,50,15,15);
      g.fillRoundRect(280,150,60,50,30,40);

      //draw Ellipses and Circles
      g.drawOval(10,250,50,50);
      g.fillOval(90,250,75,50);
      g.drawOval(190,260,100,40);

      //Draw Arcs
      g.drawArc(10,350,70,70,0,180);
      g.fillArc(60,350,70,70,0,35);

   }
}
```


Applet Viewer: GraphicsD...
Applet
Applet started.

GUI in Applet

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class Calc extends Applet implements ActionListener {
    Panel p1, p2;
    Label l1, l2, l3;
    TextField t1, t2, t3;
    Button b1, b2, b3, b4;

    public void init() {
        p1 = new Panel(new GridLayout(3, 2));
        p2 = new Panel(new FlowLayout());
        l1 = new Label("Num 1");
        l2 = new Label("Num 2");
        l3 = new Label("Result");
```

```
  t1 = new TextField(10);
  t2 = new TextField(10);
  t3 = new TextField(10);
  t3.setEditable(false);

p1.add(l1);
  p1.add(t1);
  p1.add(l2);
  p1.add(t2);
  p1.add(l3);
  p1.add(t3);

  b1 = new Button("add");
  b2 = new Button("mul");
  b3 = new Button("sub");
  b4 = new Button("div");
```

```
    p2.add(b1);
     p2.add(b2);
     p2.add(b3);
     p2.add(b4);

    setLayout(new GridLayout(2, 1));
     add(p1);
     add(p2);
     b1.addActionListener(this);
     b2.addActionListener(this);
     b3.addActionListener(this);
     b4.addActionListener(this);
}
```

```
public void actionPerformed(ActionEvent ae) {
        int a = Integer.parseInt(t1.getText());
        int b = Integer.parseInt(t2.getText());
        int c;

       if (ae.getActionCommand() == "add") {
            c = a + b;
            t3.setText("" + c);
        }
        if (ae.getActionCommand() == "mul") {
            c = a * b;
            t3.setText("" + c);
        }
```

```
        if (ae.getActionCommand() == "sub") {
            c = a - b;
            t3.setText("" + c);
         }
         if (ae.getActionCommand() == "div") {
            double z = (double)a/ b;
            t3.setText("" + z);
         }

    }

}
```