

Java Tutorial Exercise 1 (Introduction to Java)

Topics: Java Libraries – Swing
Event driven programming

Goals: Upon successful completion of this tutorial you should be able to:

1. Write a simple event driven Java program using the Swing libraries

Related text sections:

Java Tutorial Exercise 1 Instructions

1. Define the following java terms and concepts
 - a. class
 - b. object
 - c. primitive
2. The remainder of this tutorial exercise should be executed on the local linux or DOS platforms. You can store your directories and files in the local student directory (in which case you will need to delete them at the end of the tutorial session) or on a flash drive. Completed exercises should be uploaded to the server for submission using scp or ftp. Create a new directory called Tutorial1;
3. Enter the following Java code in a file called Tutorial1a.java in your Tutorial1 directory.

```
// file: Tutorial1a.java

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

public class Tutorial1a extends JPanel
{
    int hwX, hwY;
    String helloWorld = null;

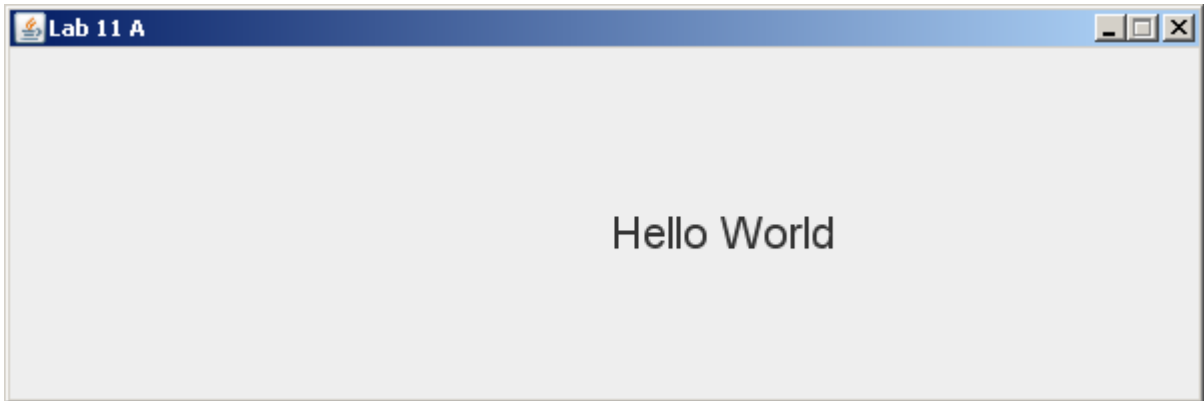
    public Tutorial1a ()
    {
        hwX = 300;
        hwY = 300;
        helloWorld = "Hello World";
        repaint();
    }

    public void paintComponent (Graphics g)
    {
        super.paintComponent (g);
        Graphics2D g2 = (Graphics2D) g;
        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
            RenderingHints.VALUE_ANTIALIAS_ON);
        g2.setFont (new Font ("Arial", Font.PLAIN, 22));
        g2.drawString (helloWorld, hwX, hwY);
    }

    public static void main (String[] args)
    {
        JFrame frame = new JFrame ("Tutorial 1 A");
        Tutorial1a Tutorial1a = new Tutorial1a ();
        frame.getContentPane().add (Tutorial1a);
        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        frame.setSize (600,600);
        frame.setVisible (true);
        frame.setResizable (false);
        frame.setLocation (100, 100);
    }
}
```

4. Compile and execute your program:
javac Tutorial1a.java
java Tutorial1a

You should see:



You have now written your first Java program (for this course)!

5. Copy Tutorial1a.java to Tutorial1b.java. Modify the program as indicated:

```
// file: Tutorial1b.java

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

public class Tutorial1b extends JPanel implements ActionListener,
    MouseMotionListener
{
    JButton theButton;
    int hwX, hwY;
    String helloWorld = null;
    int red, green, blue;

    public Tutorial1b ()
    {
        hwX = 300;
        hwY = 300;
        helloWorld = "Hello World";
        red = green = blue = 30;
        theButton = new JButton ("Change Color");
        add (theButton);
        theButton.addActionListener (this);
        addMouseMotionListener(this);
        repaint();
    }

    public void mouseDragged(MouseEvent e)
    {
        hwX = e.getX();
        hwY = e.getY();
        repaint();
    }
}
```

```

public void mouseMoved(MouseEvent e) {}

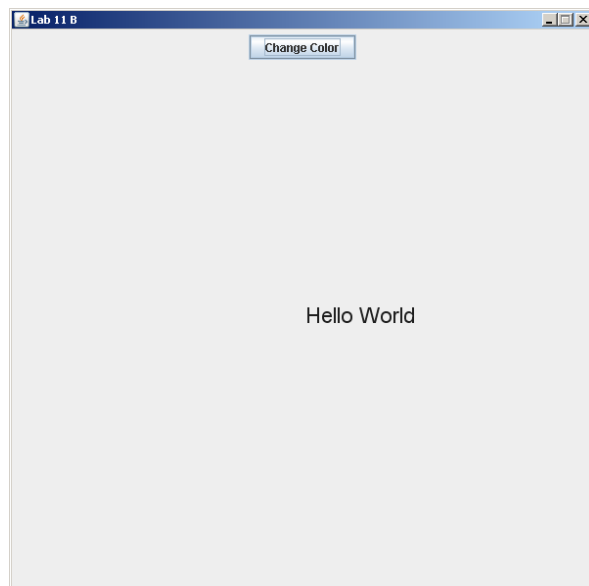
public void paintComponent (Graphics g)
{
    super.paintComponent (g);
    Graphics2D g2 = (Graphics2D) g;
    g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
        RenderingHints.VALUE_ANTIALIAS_ON);
    g2.setPaint (new Color (red, green, blue));
    g2.setFont (new Font ("Arial", Font.PLAIN, 22));
    g2.drawString (helloWorld, hwX, hwY);
}

public void actionPerformed (ActionEvent e)
{
    if (e.getSource() == theButton)
    {
        red = (red * 29) % 254 + 2;
        green = (green * 31) % 254 + 2;
        blue = (blue * 53) % 254 + 2;
        repaint ();
    }
}

public static void main (String[] args)
{
    JFrame frame = new JFrame ("Tutorial 1 B");
    Tutorial1b Tutorial1b = new Tutorial1b ();
    frame.getContentPane().add (Tutorial1b);
    frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
    frame.setSize (600,600);
    frame.setVisible (true);
    frame.setResizable (false);
    frame.setLocation (200, 200);
}
}

```

6. Compile and execute the program; you should see:
7. Use the button to change the color of the text and use the mouse to drag the text around the screen.



8. Copy Tutorial1b.java to Tutorial1c.java. Make the changes necessary to compile and execute the class Tutorial1c.

9. Add a Mouse Listener to Tutorial1c:

```
public class Tutorial1b extends JPanel implements ActionListener,  
MouseMotionListener, MouseListener
```

You will need to include the following MouseListener method stubs:

```
public void mousePressed(MouseEvent e) {}  
public void mouseReleased(MouseEvent e) {}  
public void mouseEntered(MouseEvent e) {}  
public void mouseExited(MouseEvent e) {}  
public void mouseClicked(MouseEvent e) {}
```

10. Modify the mouseExited and mouseEntered methods to prevent the user from dragging the text outside the bounds of the window frame.

Java Tutorial Exercise 2 (Java Lists and Sorting)

Topics: Java Lists

Goals: Upon successful completion of this tutorial you should be able to:

1. Create an array based list in a java program
2. Randomly populate a list of integers
3. Sort the list of integers using your simple and advanced sorts

Related text sections:

Java Tutorial Exercise 2 Instructions

Create a new subdirectory called Tutorial2 and enter the following java program as RList.java

```
// file: RList.java

import java.lang.*;
import java.util.Random;

public class RList
{
    int size;
    int list [];
    static int ASCENDING = 1;
    static int DESCENDING = 2;

    public RList ( )
    {
    }

    public RList (int S)
    {
        size = S;
        list = new int [size];
        for (int i = 0; i < size; i++)
            list[i] = i * 3 % (2 * i + 1);
    }

    public RList (RList other)
    {
        size = other.size;
        list = new int [size];
        for (int i = 0; i < size; i++)
            list[i] = other.list[i];
    }

    public String toString ( )
    {
        String string = "";
        for (int i = 0; i < size; i++)
            string += list[i] + " ";
        return string;
    }

    public void Sort (int AD)
    {
    }

    public static void main (String[] args)
    {
        RList rlist1 = new RList (10);
        RList rlist2 = new RList (rlist1);
        System.out.println ("\nRandom List 1");
        System.out.println (rlist1);
        rlist1.Sort (RList.ASCENDING);
        System.out.println (rlist1);
    }
}
```

```

        System.out.println ("\nRandom List 2");
        System.out.println (rlist2);
        rlist2.Sort (RList.DESCEDING);
        System.out.println (rlist2);
    }
}

```

1. Compile and execute the program:

```

javac RList.java
java RList

```

On the system console you should see:

```

Random List 1
0 1 1 4 7 10 0 1 2 3
0 1 1 4 7 10 0 1 2 3

Random List 2
0 1 1 4 7 10 0 1 2 3
0 1 1 4 7 10 0 1 2 3

```

Note that the lists are not sorted.....

2. Modify the Sort method of the RList class to sort the list into ascending order using your simple sort if the input parameter is ASCENDING and into descending order using your advanced sort if the input parameter is DESCENDING.
3. This program uses an internal main function to test the RList class. The RList class can also be used as part of another program. Enter, compile and run the following class called Tutorial2a.

```

// file: Tutorial2a.java

import java.lang.*;

public class Tutorial2a
{
    RList rlist1;
    RList rlist2;

    public Tutorial2a ()
    {
        rlist1 = new RList (10);
        rlist2 = new RList (rlist1);
        System.out.println ("\nRandom List 1");
        System.out.println (rlist1.toString());
        rlist1.Sort (RList.ASCENDING);
        System.out.println (rlist1.toString());
        System.out.println ("\nRandom List 2");
        System.out.println (rlist2.toString());
        rlist2.Sort (RList.DESCEDING);
        System.out.println (rlist2.toString());
    }

    public static void main (String[] args)
    {
        Tutorial2a tutorial2a = new Tutorial2a ();
    }
}

```

4. Modify the RList class to include a random number generator. Information about the java Random class can be found at:

<http://java.sun.com/j2se/1.4.2/docs/api/java/util/Random.html>

Modify the constructor with integer input parameter `S` to generate a list populated with `S` randomly selected integers in the range $-100 \leq \text{list}[i] \leq 100$.

Modify the default constructor to randomly select a value for `size` ($5 \leq \text{size} \leq 50$) and to generate a list populated with `size` randomly selected integers in the range $-100 \leq \text{list}[i] \leq 100$.

Java Tutorial Exercise 3 (Java Illustration)

Topics: Java 2D graphics

Goals: Upon successful completion of this tutorial you should be able to:

1. Use Swing and Java graphics to illustrate the actions of your simple sorting algorithm
2. Use Swing and Java graphics to illustrate the actions of your advanced sorting algorithm

Related text sections:

Java Tutorial Exercise 3 Instructions

Create a new subdirectory called Tutorial3. Copy RList.java from Tutorial2 to Tutorial3. Add the following Paint method to RList.

```
public void Paint (Graphics2D g2, int ulX, int ulY, int lrX, int lrY,
                  Color minColor, Color maxColor)
{
    if (size < 1)
        return;
    int min = 0;
    int max = 0;
    for (int i = 1; i < size; i++)
    {
        if (list[i] < min) min = list[i];
        if (list[i] > max) max = list[i];
    }
    int deltaX = (lrX - ulX) / (2 * size + 1);
    int range = max - min + 1;
    int deltaY = (lrY - ulY) / range;
    int deltaR = (maxColor.getRed() - minColor.getRed()) / range;
    int deltaG = (maxColor.getGreen() - minColor.getGreen()) / range;
    int deltaB = (maxColor.getBlue() - minColor.getBlue()) / range;
    for (int i = 0; i < size; i++)
    {
        int left = ulX + (2*i+1) * deltaX;
        int right = left + deltaX;
        int top = lrY - list[i] * deltaY;
        int bottom = lrY;
        Color c = new Color (list[i] * deltaG, list[i] * deltaR,
                             list[i] * deltaB);
        g2.setPaint (c);
        g2.fillRect (left, top, right-left, bottom-top);
    }
}
```

This function is designed to draw vertical bars representing the values in the list using the rectangle and colors passed to it. The parameters to this function are, in order, the x and y coordinates of the upper left corner of the rectangle in which the vertical bars will be drawn; the x and y coordinates of the lower right corner of the rectangle; the minimum and maximum colors. The width of the rectangular area is divided so that 'size' bars can be drawn along a horizontal axis. The height of the rectangular area is divided into units representing the range of the magnitudes of the values in the list. The color range is divided into units representing the range of the magnitudes of the values in the list.

Currently this function does not handle negative integer values in the list.

1. Enter the following java class:

```
// file:Tutorial3a.java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Tutorial3a extends JPanel implements ActionListener
{
```

```

RList rlist;
JButton theButton;
public Tutorial3a ()
{
    rlist = new RList ();
    System.out.println (rlist);
    theButton = new JButton ("Sort List");
    add (theButton);
    theButton.addActionListener (this);
    repaint();
}

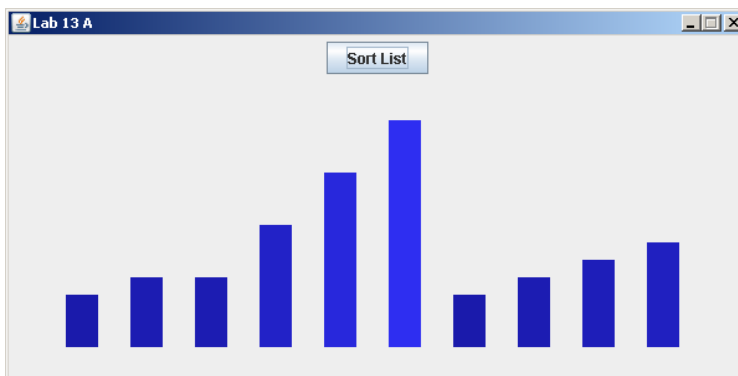
public void paintComponent (Graphics g)
{
    super.paintComponent (g);
    Graphics2D g2 = (Graphics2D) g;
    g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
        RenderingHints.VALUE_ANTIALIAS_ON);
    rlist.Paint (g2, 20, 50, 580, 250, new Color (20, 20, 150),
        new Color (50, 50, 250));
}

public void actionPerformed (ActionEvent e)
{
    if (e.getSource() == theButton)
    {
        rlist.Sort (RList.ASCENDING);
        repaint();
    }
}

public static void main (String[] args)
{
    JFrame frame = new JFrame ("Tutorial 3 A");
    Tutorial3a Tutorial3a = new Tutorial3a ();
    frame.getContentPane().add (Tutorial3a);
    frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
    frame.setSize (600,300);
    frame.setVisible (true);
    frame.setResizable (false);
    frame.setLocation (200, 200);
}
}

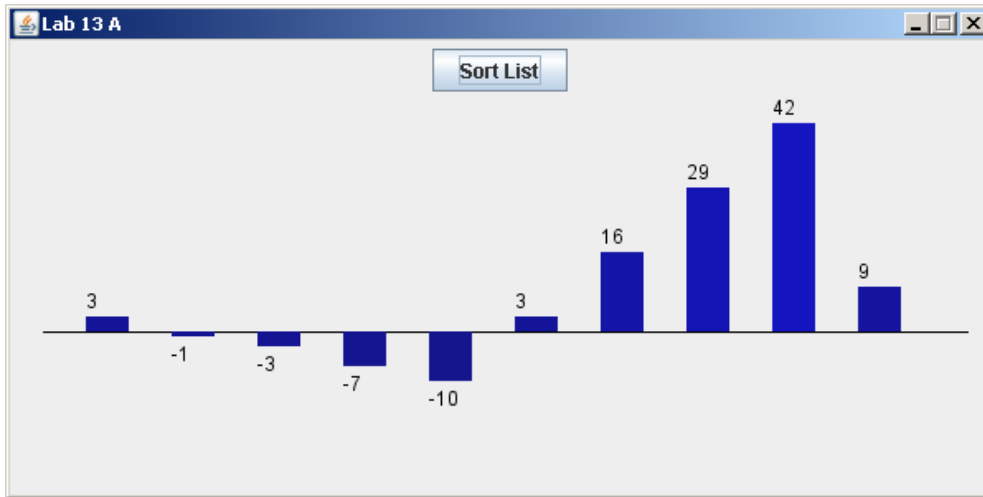
```

2. Modify your constructor so generates numbers between 0 and 100. Compile program. You should see something

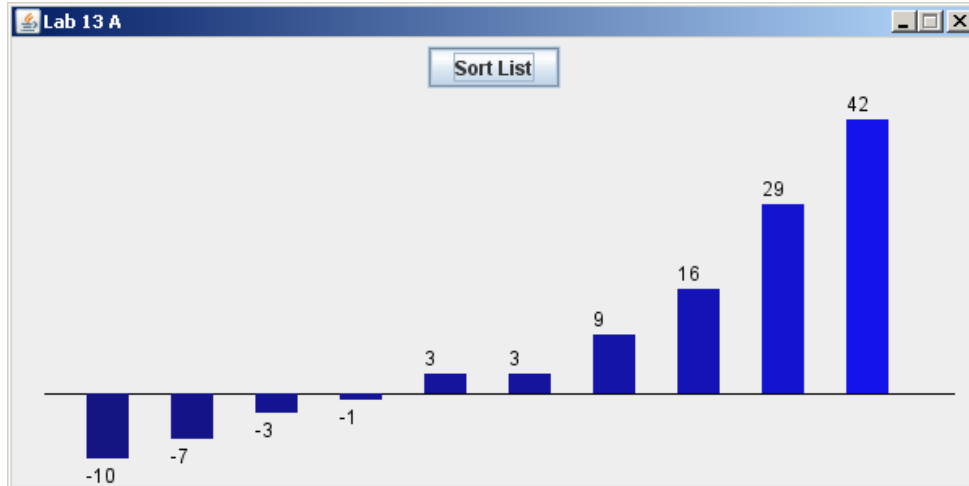


second list that it only positive between 0 and 100 and test the program you should see like:

3. Modify the RList Paint function as follows:
 - a. Include a horizontal axis
 - b. Label each vertical bar with its axis
 - c. Allow negative values – these values should appear as vertical bars below the horizontal axis.
- Return your second constructor to its original state (generating numbers between -100 and 100) and test your program. Your display *might* look like:



Test the simple sort function in your RList class. After the Sort List button is pressed, the list should be displayed in fully sorted order. Your display *might* look like:



4. Modify the program so that each press of the “Sort List” button causes 1 pass of your simple sort to be executed. Multiple presses of the button will illustrate the modifications made to the list that eventually result in a fully sorted list. You will most likely need to add a “SortPass” method to your RList class.

Your completed RList.java and Tutorial3a.java files should be placed in ~tiawatts/cs460drop. The files should be called *your-last-name.java* and *your-last-nameT3.java* respectively.