Java Network Programming

Reader 1

Learning Java Programming By Examples

Dr. Wanlei Zhou

Contents

Lesson 1. The First Java Programs

1.1. Install Java compiler.

This can be done by downloading the Java compiler and installing it following the instructions.

1.2. The first Java program

a. Use any editor, create a file called "HelloWorld.java":

```
/* Hello World, the first Java application */
class HelloWorld {
	public static void main (String args[]) {
		System.out.println("Hello World!");
	}
}
```

b. Compile the program using "javac HelloWorld.java"

c. Execute the program using "Java HelloWorld"

d. Change the output into "This is my first Java program".

1.3. The first Java applet

a. Use any editor, create a file called "HelloWorldApplet.java":

```
/* First Hello World Applet */
import java.awt.Graphics;
public class HelloWorldApplet extends java.applet.Applet {
    public void paint(Graphics g) {
       g.drawString("Hello world!", 5, 25);
    }
}
```

b. Create a file called "testapplet.html":

```
<HTML>
<HEAD>
<TITLE>Hello to Everyone!</TITLE>
</HEAD>
<BODY>
<P>My Java Applet says:
<APPLET CODE="HelloWorldApplet.class" WIDTH=150 HEIGHT=25>
</APPLET>
</BODY>
</HTML>
```

c. Compile "HelloWorldApplet.java"

d. Use Netscape of IE to open the testapplet.html file. View the execution of the applet.

e. Change the output into "This is my first Java applet".

1.4. The first object-oriented program

a. Create Java program "Motorcycle.java":

```
class Motorcycle {
  String make;
  String color;
  boolean engineState;

  void startEngine() {
     if (engineState == true)
         System.out.println("The engine is already on.");
     else {
         engineState = true;
         System.out.println("The engine is now on.");
     }
  }

  void showAtts() {
    System.out.println("This motorcycle is a "
        + color + " " + make);
    if (engineState == true)
        System.out.println("The engine is on.");
    else System.out.println("The engine is off.");
  }

  public static void main (String args[]) {
    Motorcycle m = new Motorcycle();
    m.make = "Yamaha RZ350";
    m.color = "yellow";
    System.out.println("Calling showAtts...");
    m.showAtts();
    System.out.println("--------");
    System.out.println("Starting engine...");
    m.startEngine();
    System.out.println("--------");
    System.out.println("Calling showAtts...");
    m.showAtts();
    System.out.println("--------");
    System.out.println("Starting engine...");
    m.startEngine();
  }
}
```

b. Compile and test run the program.

c. Add an attribute "int yearMade", modify other parts of the program, and test it again.

1.5. Fonts in Java

a. Create "HelloAgainApplet.java":

```
import java.awt.Graphics;
import java.awt.Font;
import java.awt.Color;

public class HelloAgainApplet extends java.applet.Applet {
  Font f = new Font("TimesRoman",Font.BOLD,36);
  public void paint(Graphics g) {
    g.setFont(f);
    g.setColor(Color.red);
    g.drawString("Hello again!", 5, 40);
  }
}
```

b. Create "HelloAgain.html":

```
<HTML>
<HEAD>
<TITLE>Another Applet</TITLE>
</HEAD>
<BODY>
<P>My second Java applet says:
<BR><APPLET CODE="HelloAgainApplet.class" WIDTH=200 HEIGHT=50>
</APPLET>
</BODY>
</HTML>
```

c. Compile and test run the applet.

d. Modify the program to use "Font.ITALIC", and the "Color.yellow".

Lesson 2. Java Basics

2.1. Arithmetic calculation

a. Create program "ArithmeticTest.java":

```
class ArithmeticTest {
public static void main (String args[]) {
    short x = 6;
    int y = 4;
    float a = 12.5f;
    float b = 7f;

    System.out.println("x is " + x + ", y is " + y);
    System.out.println("x + y = " + (x + y));
    System.out.println("x - y = " + (x - y));
    System.out.println("x / y = " + (x / y));
    System.out.println("x % y = " + (x % y));
    System.out.println("a is " + a + ", b is " + b);
    System.out.println("a / b = " + (a / b));
}
}
```

b. Compile and test run the program.

2.2. Dates

a. Create program "CreateDates.java":

```
import java.util.*;
import java.text.*;
class CreateDates {
    public static void main(String args[]) {
        Date d1 = new Date(); // the current date
        System.out.println("Date 1: " + d1);

        // year, month (starting 0), day
        GregorianCalendar d2 = new GregorianCalendar(1999, 7, 1);
        System.out.println("Date 2: " + d2.get(d2.DAY_OF_WEEK) +" "+
d2.get(d2.MONTH) +" "+ d2.get(d2.DATE) +" "+ d2.get(d2.YEAR) );

         // A date string
         DateFormat fmt = DateFormat.getDateInstance(DateFormat.FULL,
Locale.US);
        try {
           Date d3 = fmt.parse("Saturday, July 4, 1998");
```

```
          System.out.println("Date 3: " + d3);
         } catch (ParseException e) {
           System.err.println(e);
         }
    }
}
```

b. Compile and test run the program.

2.3. Strings

a. Create program "TestString.java":

```
class TestString {
    public static void main(String args[]) {
        String str = "Now is the winter of our discontent";
        String str1 = "This is another string";
        String str2;

        System.out.println("The string is: " + str);
        System.out.println("Length of this string: "
                + str.length());
        System.out.println("The character at position 5: "
                + str.charAt(5));
        System.out.println("The substring from 11 to 17: "
                + str.substring(11, 17));
        System.out.println("The index of the character d: "
                + str.indexOf('d'));
        System.out.print("The index of the beginning of the ");
        System.out.println("substring \"winter\": "
                + str.indexOf("winter"));
        System.out.println("The string in upper case: "
                + str.toUpperCase());
        System.out.println("String1: " + str1);
        System.out.println("Same object (str and str1)? " + (str == str1));
        str2 = str;
        System.out.println("String2: " + str2);
        System.out.println("Same object (str and str2)? " + (str == str2));
    }
}
```

b. Compile and test run the program.

2.4. Arrays

a. Create program "ArrayTest.java":

```
class ArrayTest {
   String[] firstNames = { "Dennis", "Grace", "Bjarne", "James" };
   String[] lastNames = new String[firstNames.length];

   void printNames() {
             int i = 0;
             System.out.println(firstNames[i] + " " + lastNames[i]);
             i++;
             System.out.println(firstNames[i] + " " + lastNames[i]);
             i++;
             System.out.println(firstNames[i] + " " + lastNames[i]);
             i++;
             System.out.println(firstNames[i] + " " + lastNames[i]);
       }
       public static void main (String args[]) {
             ArrayTest a = new ArrayTest();
             a.printNames();
             System.out.println("----------");
             a.lastNames[0] = "Ritchie";
             a.lastNames[1] = "Hopper";
             a.lastNames[2] = "Stroustrup";
             a.lastNames[3] = "Gosling";
             a.printNames();
       }
}
```

b. Compile and test run the program.

2.5. Loops

a. Create program "NameLoop.java":

```
class NameLoop {
   String[] firstNames = { "Dennis", "Grace", "Bjarne", "James" };
   String[] lastNames = new String[firstNames.length];

   void printNames() {
              for (int i = 0; i < firstNames.length; i++)
                     System.out.println(firstNames[i] + " " + lastNames[i]);
       }

       public static void main (String args[]) {
              ArrayTest a = new ArrayTest();
              a.printNames();
              System.out.println("----------");
              a.lastNames[0] = "Ritchie";
              a.lastNames[1] = "Hopper";
              a.lastNames[2] = "Stroustrup";
              a.lastNames[3] = "Gosling";
              a.printNames();
       }
}
```

b. Compile and test run the program.

2.6. Parameter passing – by reference

a. Create program “PassReference.java”:

```
class PassByReference {
     int onetoZero(int arg[]) {
         int count = 0;

         for (int i = 0; i < arg.length; i++) {
             if (arg[i] == 1) {
                 count++;
                 arg[i] = 0;
             }
         }
         return count;
     }
     public static void main (String arg[]) {
       int arr[] = { 1, 3, 4, 5, 1, 1, 7 };
        PassByReference test = new PassByReference();
        int numOnes;
        System.out.print("Values of the array: [ ");
        for (int i = 0; i < arr.length; i++) {
           System.out.print(arr[i] + " ");
        }
        System.out.println("]");
        numOnes = test.onetoZero(arr);
        System.out.println("Number of Ones = " + numOnes);
        System.out.print("New values of the array: [ ");
        for (int i = 0; i < arr.length; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println("]");
     }
}
```

b. Compile and test run the program.

2.7. Handling arguments

a. Create program “EchoArgs”:

```
class EchoArgs {
    public static void main(String args[]) {
        for (int i = 0; i < args.length; i++) {
            System.out.println("Argument " + i + ": " + args[i]);
        }
    }
}
```

b. Compile and test run the program, using “javac EchoArgs 1 2 3 Go”

2.8. Converting types

a. Create program “SumAverage.java”:

```
class SumAverage {
    public static void main (String args[]) {
        int sum = 0;

        for (int i = 0; i < args.length; i++) {
                        sum += Integer.parseInt(args[i]);
        }

        System.out.println("Sum is: " + sum);
        System.out.println("Average is: " +
            (float)sum / args.length);
    }
}
```

b. Compile and test run the program, using “javac SumArgs 1 2 3 4”

Lesson 3. Classes and Methods

3.1. Constructor

a. Create program “Person.java”:

```
class Person {
    String name;
    int age;

   Person(String n, int a) {
        name = n;
        age = a;
    }

    void printPerson() {
        System.out.print("Hi, my name is " + name);
        System.out.println(". I am " + age + " years old.");
    }

  public static void main (String args[]) {
     Person p;
     p = new Person("Laura", 20);
     p.printPerson();
     System.out.println("--------");
     p = new Person("Tommy", 3);
     p.printPerson();
     System.out.println("--------");
  }
}
```

b. Compile and test run the program.

c. Add attribute “String sex” to the Person class. Test it again.

3.2. Inheritance

a. Create class “PrintClass.java”:

```
class PrintClass {
    int x = 0;
    int y = 1;

    void printMe() {
        System.out.println("x is " + x + ", y is " + y);
        System.out.println("I am an instance of the class " +
        this.getClass().getName());
    }
}
```

b. Create class “PringSubClass.java” that inherits from PrintClass:

```
class PrintSubClass extends PrintClass {
    int z = 3;

    public static void main(String args[]) {
        PrintSubClass obj = new PrintSubClass();
        obj.printMe();
    }
}
```

c. Compile and test the program. The problem: z is not printed out.

d. To overcome the problem: overwrite the PrintMe() method.

```
class PrintSubClass2 extends PrintClass {
    int z = 3;

    void printMe() {
        System.out.println("x is " + x + ", y is " + y +
               ", z is " + z);
        System.out.println("I am an instance of the class " +
               this.getClass().getName());
    }

    public static void main(String args[]) {
        PrintSubClass2 obj = new PrintSubClass2();
        obj.printMe();
    }
}
```

Lesson 4. Applet Basics

4.1. HTML page and applet

a. Create applet “HelloAgainApplet.java”:

```
import java.awt.Graphics;
import java.awt.Font;
import java.awt.Color;

public class HelloAgainApplet extends java.applet.Applet {

  Font f = new Font("TimesRoman",Font.BOLD,36);

  public void paint(Graphics g) {
    g.setFont(f);
    g.setColor(Color.red);
    g.drawString("Hello again!", 5, 40);
  }
}
```

b. Create HTML file “HelloAgainApplet.html”:

```
<HTML>
<HEAD>
<TITLE>This page has an applet on it</TITLE>
</HEAD>
<BODY>
<H2>Lesson 4: Session 1: Hello Again</H2>
<P>My second Java applet says:
<BR><APPLET CODE="HelloAgainApplet.class" WIDTH=200 HEIGHT=50>
Hello Again!
</APPLET><P>
<A HREF="HelloAgainApplet.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and view the html file.

4.2. Alignment

a. Create html file “HelloAgainAlign.html”:

```
<HTML>
<HEAD>
<TITLE>This page has an applet on it, aligned left</TITLE>
</HEAD>
<BODY>
<H2>Lesson 4: Session 2: Hello Again (Align)</H2>
```

```
<P><APPLET CODE="HelloAgainApplet.class"
WIDTH=200 HEIGHT=50 ALIGN=LEFT>Hello Again!</APPLET>
To the left of this paragraph is an applet. It's a
simple, unassuming applet, in which a small string is
printed in red type, set in 36 point Times bold.
<BR CLEAR=ALL>
<P>In the next part of the page, we demonstrate how
under certain conditions, styrofoam peanuts can be
used as a healthy snack.
<P>
<A HREF="HelloAgainApplet.java">The Source</A>
</BODY>
</HTML>
```

b. Test the html file.

4.3. Passing parameters

a. Create applet “MoreHelloApplet.java”

```
import java.awt.Graphics;
 import java.awt.Font;
 import java.awt.Color;

 public class MoreHelloApplet extends java.applet.Applet {

    Font f = new Font("TimesRoman", Font.BOLD, 36);
    String name;

    public void init() {
        name = getParameter("name");
        if (name == null)
            name = "Laura";
        name = "Hello " + name + "!";
    }
    public void paint(Graphics g) {
        g.setFont(f);
        g.setColor(Color.red);
        g.drawString(name, 5, 40);
    }
}
```

b. Create html file “MoreHelloApplet.html”:

```
<HTML>
 <HEAD>
 <TITLE>Hello!</TITLE>
 </HEAD>
 <BODY>
<H2> Lesson 4: Session 3: More Hello (Bonzo)</H2>
 <P>
 <APPLET CODE="MoreHelloApplet.class" WIDTH=200 HEIGHT=50>
 <PARAM NAME=name VALUE="Bonzo">
 Hello to whoever you are!
</APPLET><P>
<A HREF="MoreHelloApplet.java">The Source</A>
</BODY>
</HTML>
```

c. Test the html file again without the parameter.

Lesson 5. Graphics, Fonts, and Colours

5.1. Lines and rectangles

a. Create applet “LineRec.java”:

```
/* lines + rectangles */

import java.awt.Graphics;
public class LineRec extends java.applet.Applet {
  public void paint(Graphics g) {
    g.drawLine(10,10,150,25);
    g.drawRect(10,30,60,60);
    g.fillRect(120,30,60,60);
    g.drawRoundRect(10,110,60,60,10,10);
    g.fillRoundRect(120,110,60,60,20,20);
  }
}
```

b. Create html file “LineRec.html”:

```
<HTML>
<HEAD>
<TITLE>Lines</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<H2>Lesson 5, Session 1: Lines and Rectangles</H2>
<P>A simple graphics example that draws a line:
<BR><APPLET CODE="LineRec.class" WIDTH=200 HEIGHT=200>
</APPLET>
<P>
<A HREF="LineRec.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and test the html file.

d. Change the parameters of line and rectangles to experiment the effects.

5.2. Polygons, ellipses, and arcs.

a. Create Applet “MyPoly.java”:

```
/* polygons */
import java.awt.Graphics;
public class MyPoly extends java.applet.Applet {
  public void paint(Graphics g) {
    int exes[] = { 39,94,97,142,53,58,26 };
    int whys[] = { 33,74,36,70,108,80,106 };
    int pts = exes.length;
    g.drawPolygon(exes,whys,pts);
  }
}
```

b. Create Applet “MyPoly2.java”:

```
/* polygons 2 */
import java.awt.Graphics;
import java.awt.Polygon;
public class MyPoly2 extends java.applet.Applet {
  public void paint(Graphics g) {
    int exes[] = { 39,94,97,142,53,58,26 };
    int whys[] = { 33,74,36,70,108,80,106 };
    int pts = exes.length;
       Polygon poly = new Polygon(exes,whys,pts);
    g.fillPolygon(poly);
  }
}
```

c. Create Applet “MyArc1.java”:

```
/* arcs 1 */
import java.awt.Graphics;
public class MyArc1 extends java.applet.Applet {
  public void paint(Graphics g) {
    g.drawArc(20,20,60,60,90,180);
    g.fillArc(120,20,60,60,90,180);
  }
}
```

d. Create Applet “MyArc2.java”:

```
/* arcs */
import java.awt.Graphics;
public class MyArc2 extends java.applet.Applet {
    public void paint(Graphics g) {
    g.drawArc(10,20,150,50,25,-130);
    g.fillArc(10,80,150,50,25,-130);
  }
}
```

e. Create Applet “MyOval.java”:

```
import java.awt.Graphics;
public class MyOval extends java.applet.Applet {
  public void paint(Graphics g) {
    g.drawOval(20,20,60,60);
    g.fillOval(120,20,100,60);
  }
}
```

f. Create html file “PolyElliArc.html”:

```
<HTML>
<HEAD>
<TITLE>Polygons, Ellipses, and Arcs</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<H2>Lesson 5, Session 2: Polygons, Ellipses, and Arcs</H2>
<P>A simple graphics example that draws a polygon:
<BR><APPLET CODE="MyPoly.class" WIDTH=200 HEIGHT=150>
</APPLET>
<P>
<A HREF="MyPoly.java">The Source</A>

<P>A simple graphics example that draws a filled polygon:
<BR><APPLET CODE="MyPoly2.class" WIDTH=200 HEIGHT=150>
</APPLET>
<P>
<A HREF="MyPoly2.java">The Source</A>
```

```
<P>A simple graphics example that draws circular arcs:
<BR><APPLET CODE="MyArc1.class" WIDTH=200 HEIGHT=100>
</APPLET>
<P>
<A HREF="MyArc1.java">The Source</A>

<P>A simple graphics example that draws elliptical arcs:
<BR><APPLET CODE="MyArc2.class" WIDTH=200 HEIGHT=150>
</APPLET>
<P>
<A HREF="MyArc2.java">The Source</A>

<P>A simple graphics example that draws ovals:
<BR><APPLET CODE="MyOval.class" WIDTH=260 HEIGHT=120>
</APPLET>
<P>
<A HREF="MyOval.java">The Source</A>

</BODY>
</HTML>
```

g. Compile the applets and test the html file

h. Change some of the parameters in applets to test the effects.

5.3. Fonts.

a. Create applet “ManyFonts.java”

```
import java.awt.Font;
import java.awt.Graphics;
public class ManyFonts extends java.applet.Applet {
   public void paint(Graphics g) {
       Font f = new Font("TimesRoman", Font.PLAIN, 18);
       Font fb = new Font("TimesRoman", Font.BOLD, 18);
       Font fi = new Font("TimesRoman", Font.ITALIC, 18);
       Font fbi = new Font("TimesRoman", Font.BOLD + Font.ITALIC, 18);

       g.setFont(f);
       g.drawString("This is a plain font", 10, 25);
       g.setFont(fb);
       g.drawString("This is a bold font", 10, 50);
       g.setFont(fi);
       g.drawString("This is an italic font", 10, 75);
       g.setFont(fbi);
       g.drawString("This is a bold italic font", 10, 100);
   }
}
```

b. Create html file “ManyFonts.html”:

```
<HTML>
<HEAD>
<TITLE>Many Fonts</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<H2>Lesson 5, Session 3: Fonts</H2>
<P>Print lots of fonts in a Java applet:
<BR><APPLET CODE="ManyFonts.class" WIDTH=250 HEIGHT=150>
</APPLET>
<P>
```

```
<A HREF="ManyFonts.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and test the html file.

d. Change the parameters of the applet and view the effects.

5.4. Colours

a. Create applet “ColorBoxes.java”:

```
import java.awt.Graphics;
import java.awt.Color;
public class ColorBoxes extends java.applet.Applet {
     public void paint(Graphics g) {
         int rval, gval, bval;

         for (int j = 30; j < (getSize().height -25); j += 30)
            for (int i = 5; i < (getSize().width -25); i += 30) {
                rval = (int)Math.floor(Math.random() * 256);
                gval = (int)Math.floor(Math.random() * 256);
                bval = (int)Math.floor(Math.random() * 256);

                g.setColor(new Color(rval,gval,bval));
                g.fillRect(i, j, 25, 25);
                g.setColor(Color.black);
                g.drawRect(i-1, j-1, 25, 25);
            }
    }
}
```

b. Create html file “ColorBoxes.html”:

```
<HTML>
<HEAD>
<TITLE>Colored Boxes</TITLE>
</HEAD>
<BODY>
<H2>Lesson 5, Session 4: Colored Boxes</H2>
<P>
<APPLET CODE="ColorBoxes.class" WIDTH=400 HEIGHT=150>
</APPLET>
<P>
<A HREF="ColorBoxes.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and test the html file.

5.5. Put things together: A lamp.

a. Create applet “Lamp.java”:

```
import java.awt.*;
public class Lamp extends java.applet.Applet {
   public void paint(Graphics g) {
       // the lamp platform
       g.fillRect(0,250,290,290);
       // the base of the lamp
       g.drawLine(125,250,125,160);
       g.drawLine(175,250,175,160);
       // the lamp shade, top and bottom edges
        g.drawArc(85,157,130,50,-65,312);
        g.drawArc(85,87,130,50,62,58);
        // lamp shade, sides
        g.drawLine(85,177,119,89);
        g.drawLine(215,177,181,89);
        // dots on the shade
        g.fillArc(78,120,40,40,63,-174);
        g.fillOval(120,96,40,40);
        g.fillArc(173,100,40,40,110,180);
   }
}
```

b. Create html file “Lamp.html”:

```
<HTML>
<HEAD>
<TITLE>A Lamp (or a mushroom)</TITLE>
</HEAD>
<BODY>
<H2>Lesson 5, Session 5: A Lamp (or a mushroom)</H2>
<P>
<APPLET CODE="Lamp.class" WIDTH=300 HEIGHT=300>
If you were running Java, you would see a lamp here.
</APPLET>
<P>
<A HREF="Lamp.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and test the html file.

Lesson 6. Animation, Images, Threads, and Sound

6.1. A digital clock

a. Create applet “DigitalClock.java”:

```
import java.awt.Graphics;
import java.awt.Font;
import java.util.Calendar;
import java.util.GregorianCalendar;
public class DigitalClock extends java.applet.Applet
    implements Runnable {
    Font theFont = new Font("TimesRoman",Font.BOLD,24);
    GregorianCalendar theDate;
    Thread runner;

    public void start() {
        if (runner == null) {
            runner = new Thread(this);
            runner.start();
        }
    }
    public void stop() {
        if (runner != null) {
            runner.interrupt();
            runner = null;
        }
    }
    public void run() {
        while (true) {
            repaint();
            try { Thread.sleep(1000); }
            catch (InterruptedException e) { }
        }
    }
    public void paint(Graphics g) {
        theDate = new GregorianCalendar();
        g.setFont(theFont);
        g.drawString("" + theDate.getTime(), 10, 50);
    }
}
```

b. Create html file “DigitalClock.html”:

```
<applet code="DigitalClock.class" height=85 width=340>
</applet>
```

c. Compile the applet and test the html file.

d. Understand the thread concept and structure.

6.2. The colour swirl

a. Create applet “ColorSwirl.java”:

```
import java.awt.Graphics;
import java.awt.Color;
import java.awt.Font;
public class ColorSwirl extends java.applet.Applet
    implements Runnable {

    Font f = new Font("TimesRoman", Font.BOLD, 48);
    Color colors[] = new Color[50];
    Thread runner;

    public void start() {
        if (runner == null) {
            runner = new Thread(this);
            runner.start();
        }
    }

    public void stop() {
        if (runner != null) {
            runner.interrupt();
            runner = null;
        }
    }

    public void run() {

        // initialize the color array
        float c = 0;
        for (int i = 0; i < colors.length; i++) {
            colors[i] =
            Color.getHSBColor(c, (float)1.0,(float)1.0);
            c += .02;
        }
        // cycle through the colors
        int i = 0;
        while (true) {
            setForeground(colors[i]);
            repaint();
            i++;
            try { Thread.sleep(200); }
            catch (InterruptedException e) { }
            if (i == colors.length ) i = 0;
        }
    }

    public void paint(Graphics g) {
        g.setFont(f);
        g.drawString("Look to the Cookie!", 15, 50);
    }
}
```

b. Create html file “ColorSwirl.html”:

```
<applet code="ColorSwirl.class" height=70 width=430>
</applet>
```

c. Compile the applet and test the html file.

6.3. Use images.

a. Create applet “LadyBug.java”:

```
import java.awt.Graphics;
import java.awt.Image;
public class LadyBug extends java.applet.Applet {
    Image bugimg;
    public void init() {
        bugimg = getImage(getCodeBase(),
            "images/ladybug.gif");
    }

    public void paint(Graphics g) {
        int iwidth = bugimg.getWidth(this);
        int iheight = bugimg.getHeight(this);
        int xpos = 10;
        // 25 %
        g.drawImage(bugimg, xpos, 10,
            iwidth / 4, iheight / 4, this);
        // 50 %
        xpos += (iwidth / 4) + 10;
        g.drawImage(bugimg, xpos , 10,
            iwidth / 2, iheight / 2, this);
        // 100%
        xpos += (iwidth / 2) + 10;
        g.drawImage(bugimg, xpos, 10, this);
        // 150% x, 25% y
        g.drawImage(bugimg, 10, iheight + 30,
            (int)(iwidth * 1.5), iheight / 4, this);
    }
}
```

b. Create html file “LadyBug.html”:

```
<applet code="LadyBug.class" height=250 width=230>
</applet>
```

c. Compile the applet and test the html file.

6.4. Image adjustment

a. Create applet “ImageSampler.java”:

```
import java.applet.*;
import java.awt.*;

/** An applet that demonstrates image scaling, cropping, and flipping */
public class ImageSampler extends Applet {
  Image i;

  /** Load the image */
  public void init() {  i = getImage(this.getDocumentBase(), "tiger.gif"); }
  /** Display the image in a variety of ways */
  public void paint(Graphics g) {
    g.drawString("Original image:", 20, 20);     // Display original image
    g.drawImage(i, 110, 10, this);               // Old version of drawImage()

    g.drawString("Scaled Images:", 20, 120);     // Display scaled images
    g.drawImage(i, 20, 130, 40, 150, 0, 0, 100, 100, this);  // New version
    g.drawImage(i, 60, 130, 100, 170, 0, 0, 100, 100, this);
    g.drawImage(i, 120, 130, 200, 210, 0, 0, 100, 100, this);
    g.drawImage(i, 220, 80, 370, 230, 0, 0, 100, 100, this);

    g.drawString("Cropped Images:", 20, 250);    // Display cropped images
    g.drawImage(i, 20, 260, 70, 310, 0, 0, 50, 50, this);
    g.drawImage(i, 80, 260, 130, 310, 25, 25, 75, 75, this);
    g.drawImage(i, 140, 260, 190, 310, 50, 50, 100, 100, this);

    g.drawString("Flipped Images:", 20, 330);    // Display flipped images
    g.drawImage(i, 20, 340, 120, 440, 100, 0, 0, 100, this);
    g.drawImage(i, 130, 340, 230, 440, 0, 100, 100, 0, this);
    g.drawImage(i, 240, 340, 340, 440, 100, 100, 0, 0, this);

    g.drawString("Scaled, Cropped, and Flipped:", 20, 460);  // Do all three
    g.drawImage(i, 20, 470, 170, 550, 90, 70, 10, 20, this);
  }
}
```

b. Create html file “ImageSampler.html”:

```
<APPLET CODE="ImageSampler.class" WIDTH=400 HEIGHT=600></APPLET>
```

c. Compile the applet and test the html file.

6.5. Using Swing: A Simple Exmaple

a. Create the “SwingApplication.java” program:

```
import javax.swing.*;          //This is the final package name.
//import com.sun.java.swing.*; //Used by JDK 1.2 Beta 4 and all
                               //Swing releases before Swing 1.1 Beta 3.
import java.awt.*;
import java.awt.event.*;
public class SwingApplication {
    private static String labelPrefix = "Number of button clicks: ";
    private int numClicks = 0;
    public Component createComponents() {
        final JLabel label = new JLabel(labelPrefix + "0    ");
        JButton button = new JButton("I'm a Swing button!");
        button.setMnemonic(KeyEvent.VK_I);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                numClicks++;
                label.setText(labelPrefix + numClicks);
            }
        });
        label.setLabelFor(button);
        /*
         * An easy way to put space between a top-level container
         * and its contents is to put the contents in a JPanel
         * that has an "empty" border.
         */
        JPanel pane = new JPanel();
        pane.setBorder(BorderFactory.createEmptyBorder(
                                        30, //top
                                        30, //left
                                        10, //bottom
                                        30) //right
                                        );
        pane.setLayout(new GridLayout(0, 1));
        pane.add(button);
        pane.add(label);
        return pane;
    }
    public static void main(String[] args) {
        try {
            UIManager.setLookAndFeel(
                UIManager.getCrossPlatformLookAndFeelClassName());
        } catch (Exception e) { }
        //Create the top-level container and add contents to it.
        JFrame frame = new JFrame("SwingApplication");
        SwingApplication app = new SwingApplication();
        Component contents = app.createComponents();
        frame.getContentPane().add(contents, BorderLayout.CENTER);
        //Finish setting up the frame, and show it.
        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
        frame.pack();
        frame.setVisible(true);
    }
}
```

b. Compile the program and test run it.

6.6. Simple animation: A bouncing circle

a. Create applet “BouncingCircle.java”:

```
import java.applet.*;
import java.awt.*;
/** An applet that displays a simple animation */
public class BouncingCircle extends Applet implements Animation {
  int x = 150, y = 50, r=50;    // position and radius of the circle
  int dx = 11, dy = 7;          // trajectory of circle

  /** A timer for animation: call our animate() method ever 100
   *  milliseconds.  Creates a new thread. */
  AnimationTimer timer = new AnimationTimer(this, 100);
  /** Draw the circle at its current position */
  public void paint(Graphics g) {
    g.setColor(Color.red);
    g.fillOval(x-r, y-r, r*2, r*2);
  }

  /** Move and bounce the circle and request a redraw.
   *  The timer calls this method periodically. */
  public void animate() {
    // Bounce if we've hit an edge.
    if ((x - r + dx < 0) || (x + r + dx > bounds().width)) dx = -dx;
    if ((y - r + dy < 0) || (y + r + dy > bounds().height)) dy = -dy;
    // Move the circle.
    x += dx;  y += dy;
    // Ask the browser to call our paint() method to draw the circle
    // at its new position.
    repaint();
  }

  /** Start the timer when the browser starts the applet */
  public void start() { timer.start_animation(); }

  /** Pause the timer when browser pauses the applet */
  public void stop() { timer.pause_animation(); }
}

/** This interface for objects that can be animated by an AnimationTimer */
interface Animation { public void animate(); }

/** The thread class that periodically calls the animate() method */
class AnimationTimer extends Thread {
  Animation animation;  // The animation object we're serving as timer for
  int delay;            // How many milliseconds between "animation frames"

  public AnimationTimer(Animation animation, int delay) {
    this.animation = animation;
    this.delay = delay;
  }

  public void start_animation() {
    if (isAlive()) super.resume();
    else start();
  }
  public void pause_animation() { suspend(); }

  /** Loop forever, calling animate(), and then pausing the specified time. */
  public void run() {
    for(;;) {
      animation.animate();
      try { Thread.sleep(delay); } catch (InterruptedException e) { ; }
    }
  }
}
```

b.Create html file “BouncingCircle.html”:

```
<APPLET CODE="BouncingCircle.class" WIDTH=300 HEIGHT=300></APPLET>
```

c. Compile the applet and test the html file.

d. Change the image to a square and test the program again.

6.7. Sound

a. Create applet “AudioLoop.java”:

```
import java.awt.Graphics;
import java.applet.AudioClip;

public class AudioLoop extends java.applet.Applet
    implements Runnable {
    AudioClip bgsound;
    AudioClip beep;
    Thread runner;

    public void start() {
         if (runner == null) {
             runner = new Thread(this);
             runner.start();
         }
    }

    public void stop() {
        if (runner != null) {
            if (bgsound != null) bgsound.stop();
            runner.interrupt();
            runner = null;
        }
    }

    public void init() {
        bgsound = getAudioClip(getCodeBase(),"audio/loop.au");
        beep = getAudioClip(getCodeBase(), "audio/beep.au");
    }

    public void run() {
        if (bgsound != null) bgsound.loop();
        while (runner != null) {
            try { Thread.sleep(5000); }
            catch (InterruptedException e) { }
            if (beep != null) beep.play();
        }
    }

    public void paint(Graphics g) {
        g.drawString("Playing Sounds....", 10, 10);
    }
}
```

b. Create html file “AudioLoop.html”:

```
<applet code="AudioLoop.class" height=200 width=400>
</applet>
```

c. Compile the applet and test the html file.

d. Change the audio clips to play another set of sound.

Lesson 7. Event and Interactivity

7.1. Draw spots

a. Create applet “Spots.java”:

```
/* draw blue spots at each mouse click */
import java.awt.Graphics;
import java.awt.Color;
import java.awt.Event;

public class Spots extends java.applet.Applet {
    final int MAXSPOTS = 10;
    int xspots[] = new int[MAXSPOTS];
    int yspots[] = new int[MAXSPOTS];
    int currspots = 0;

    public void init() {
        setBackground(Color.white);
    }

    public boolean mouseDown(Event evt, int x, int y) {
        if (currspots < MAXSPOTS) {
            addspot(x,y);
            return true;
        }
        else {
           System.out.println("Too many spots.");
           return false;
        }
    }

    void addspot(int x,int y) {
         xspots[currspots] = x;
         yspots[currspots] = y;
         currspots++;
         repaint();
    }

    public void paint(Graphics g) {
        g.setColor(Color.blue);
        for (int i = 0; i < currspots; i++) {
              g.fillOval(xspots[i] - 10, yspots[i] - 10, 20, 20);
       }
    }
}
```

b. Create html file “Spots.html”

```
<HTML>
<HEAD>
<TITLE>Draw Spots</TITLE>
</HEAD>
<H2>Lesson 7, Session 1: Draw Spots</H2>
<BODY>
<P>
<APPLET CODE="Spots.class" WIDTH=300 HEIGHT=300>
</APPLET>
<P>
<A HREF="Spots.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet the test the html file.

d. Change the applet to draw small red squares..

7.2. Draw lines

a. Create applet “Lines.java”:

```
/* draw lines at each click and drag */
import java.awt.Graphics;
import java.awt.Color;
import java.awt.Event;
import java.awt.Point;

public class Lines extends java.applet.Applet {
    final int MAXLINES = 10;
    Point starts[] = new Point[MAXLINES]; // starting points
    Point ends[] = new Point[MAXLINES];    // endingpoints
    Point anchor;    // start of current line
    Point currentpoint; // current end of line
    int currline = 0; // number of lines

    public void init() {
        setBackground(Color.white);
    }

    public boolean mouseDown(Event evt, int x, int y) {
       if (currline < MAXLINES) {
           anchor = new Point(x,y);
           return true;
         }
       else  {
             System.out.println("Too many lines.");
          return false;
      }
    }

    public boolean mouseUp(Event evt, int x, int y) {
        if (currline < MAXLINES) {
            addline(x,y);
            return true;
        }
        else return false;
    }

    public boolean mouseDrag(Event evt, int x, int y) {
       if (currline < MAXLINES) {
           currentpoint = new Point(x,y);
           repaint();
           return true;
        }
        else return false;
    }

    void addline(int x,int y) {
        starts[currline] = anchor;
        ends[currline] = new Point(x,y);
        currline++;
        currentpoint = null;
             anchor = null;
        repaint();
    }

    public void paint(Graphics g) {
        // Draw existing lines
        for (int i = 0; i < currline; i++) {
            g.drawLine(starts[i].x, starts[i].y,
                 ends[i].x, ends[i].y);
        }
        // draw current line
        g.setColor(Color.blue);
        if (currentpoint != null)
            g.drawLine(anchor.x,anchor.y,
            currentpoint.x,currentpoint.y);
    }
}
```

b. Create html file “Line.html”:

```
<HTML>
<HEAD>
<TITLE>Draw Some Lines</TITLE>
</HEAD>
<H2>Lesson 7, Session 2: Draw Some Lines</H2>
<BODY>
<P>
<APPLET CODE="Lines.class" WIDTH=300 HEIGHT=300>
</APPLET>
<P>
<A HREF="Lines.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and test the html file.

7.3. Scribble lines

a. Create applet “Scribble.java”:

```
import java.applet.*;
import java.awt.*;

/**
 * This applet lets the user scribble with the mouse.  It demonstrates
 * the Java 1.0 event model.
 **/
public class Scribble extends Applet {
  private int last_x = 0, last_y = 0;  // Fields to store a point in.

  // Called when the user clicks.
  public boolean mouseDown(Event e, int x, int y) {
    last_x = x; last_y = y;            // Remember the location of the click.
    return true;
  }

  // Called when the mouse moves with the button down
  public boolean mouseDrag(Event e, int x, int y)  {
    Graphics g = getGraphics();        // Get a Graphics to draw with.
    g.drawLine(last_x, last_y, x, y);  // Draw a line from last point to this.
    last_x = x; last_y = y;            // And update the saved location.
    return true;
  }
}
```

b. Create html file “Scribble.html”:

```
<HTML>
<HEAD>
<TITLE>The Scribble Applet</TITLE>
```

```
</HEAD>
<BODY bgColor="white">
Hold the mouse button down and scribble in the applet.
This applet does not know how to refresh itself.
<P>
<APPLET code="Scribble.class" width=500 height=300>
</APPLET>
</BODY>
</HTML>
```

c. Compile the applet and test the html file.

7.4. Keyboard control

a. Create applet “Keys.java”:

```
/* press a key then use arrows to move it around */

import java.awt.Graphics;
import java.awt.Color;
import java.awt.Event;
import java.awt.Font;

public class Keys extends java.applet.Applet {
  char currkey;
  int currx;
  int curry;

  public void init() {
    currx = (this.size().width / 2) -8;  //default
    curry = (this.size().height / 2) -16;
    setBackground(Color.white);
    setFont(new Font("Helvetica",Font.BOLD,36));
  }

  public boolean keyDown(Event evt, int key) {
    switch (key) {
    case Event.DOWN:
      curry += 5;
      break;
    case Event.UP:
      curry -= 5;
      break;
    case Event.LEFT:
      currx -= 5;
      break;
    case Event.RIGHT:
      currx += 5;
      break;
    default:
      currkey = (char)key;
    }
    repaint();
    return true;
  }

  public void paint(Graphics g) {
    if (currkey != 0) {
      g.drawString(String.valueOf(currkey), currx,curry);
    }
  }
}
```

b. Create html file “Keys.html”:

```
<HTML>
<HEAD>
<TITLE>Type a Character</TITLE>
</HEAD>
<BODY>
<H2>Lesson 7, Session 3: Type a Character and move it around</H2>
<P>
<APPLET CODE="Keys.class" WIDTH=300 HEIGHT=300>
</APPLET>
<P>
<A HREF="Keys.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and test the html file.

7.5. Draw lines (Handling Java 1.1 events)

a. Create applet “LinesNew.java”:

```
/* draw lines at each click and drag (1.1) */
import java.awt.Graphics;
import java.awt.Color;
import java.awt.Point;
import java.awt.event.*;

public class LinesNew extends java.applet.Applet
   implements MouseListener,MouseMotionListener {
    final int MAXLINES = 10;
    Point starts[] = new Point[MAXLINES]; // starting points
    Point ends[] = new Point[MAXLINES];    // endingpoints
    Point anchor;    // start of current line
    Point currentpoint; // current end of line
    int currline = 0; // number of lines

    public void init() {
        setBackground(Color.white);
             // register event listeners
             addMouseListener(this);
             addMouseMotionListener(this);
    }

       // needed to satisfy listener interfaces
    public void mouseMoved(MouseEvent e) {}
       public void mouseClicked(MouseEvent e) {}
    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}

       // same as mouseDown
    public void mousePressed(MouseEvent e) {
       if (currline < MAXLINES)
           anchor = new Point(e.getX(),e.getY());
       else
             System.out.println("Too many lines.");
    }

       // same as mouseUp
    public void mouseReleased(MouseEvent e) {
        if (currline < MAXLINES)
```

```
            addline(e.getX(),e.getY());
    }

       // same as mouseDrag
       public void mouseDragged(MouseEvent e) {
        if (currline < MAXLINES) {
            currentpoint = new Point(e.getX(),e.getY());
            repaint();
        }
    }

    void addline(int x,int y) {
        starts[currline] = anchor;
        ends[currline] = new Point(x,y);
        currline++;
        currentpoint = null;
             anchor = null;
        repaint();
    }

    public void paint(Graphics g) {
        // Draw existing lines
        for (int i = 0; i < currline; i++) {
            g.drawLine(starts[i].x, starts[i].y,
                 ends[i].x, ends[i].y);
        }
        // draw current line
        g.setColor(Color.blue);
        if (currentpoint != null)
            g.drawLine(anchor.x,anchor.y,
            currentpoint.x,currentpoint.y);
    }
}
```

b. Create html file:

```
<HTML>
<HEAD>
<TITLE>Draw Some Lines</TITLE>
</HEAD>
<H2>Lesson 7, Session 4: Draw Some Lines (new events)</H2>
<BODY>
<P>
<APPLET CODE="LinesNew.class" WIDTH=300 HEIGHT=300>
</APPLET>
<P>
<A HREF="LinesNew.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet in Java 1.1 and test the html file.

Lesson 8. Using the Abstract Windowing Toolkit (AWT)

8.1. Buttons

a. Create applet “ButtonTest.java”:

```
/* create a few buttons */
import java.awt.*;
public class ButtonTest extends java.applet.Applet {
  public void init() {
       add(new Button("Rewind"));
    add(new Button("Play"));
    add(new Button("Fast Forward"));
    add(new Button("Stop"));
  }
}
```

b. Create html file “ButtonTest.html”:

```
<HTML>
<HEAD>
<TITLE>Buttons</TITLE>
</HEAD>
<BODY>
<H2>Lesson 8, Session 1: Buttons</H2>
<P>
<APPLET CODE="ButtonTest.class" WIDTH=250 HEIGHT=50>
</APPLET>
<P>
<A HREF="ButtonTest.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and test the html file.

d. Change the applet to use buttons of “home, up, down, left, right”.

8.2. Check boxes (nonexclusive)

a. Create applet “CheckboxTest.java”:

```
/* check boxes */
import java.awt.*;
public class CheckboxTest extends java.applet.Applet {
  public void init() {
    setLayout(new FlowLayout(FlowLayout.LEFT));
    add(new Checkbox("Shoes"));
    add(new Checkbox("Socks"));
    add(new Checkbox("Pants"));
    add(new Checkbox("Underwear", true));
    add(new Checkbox("Shirt"));
  }
}
```

b. Create html file “CheckboxTest.html”:

```
<HTML>
<HEAD>
<TITLE>Checkboxes (nonexclusive)</TITLE>
</HEAD>
<BODY>
<H2>Lesson 8, Session 2: Checkboxes (nonexclusive)</H2>
<P>
<APPLET CODE="CheckboxTest.class" WIDTH=100 HEIGHT=150>
</APPLET>
<P>
<A HREF="CheckboxTest.java">The Source</A>
```

```
</BODY>
</HTML>
```

c. Compile the applet and test the html file.

d. Change the checkboxes to “papers, books, pencils, clips, writing pads, rulers” and test the program again.

8.3. Radio Buttons (exclusive)

a. Create applet “CheckboxGroupTest.java”:

```
/* check boxes (radio buttons) */

import java.awt.*;
public class CheckboxGroupTest extends java.applet.Applet {
  public void init() {
    setLayout(new FlowLayout(FlowLayout.LEFT));
    CheckboxGroup cbg = new CheckboxGroup();

    add(new Checkbox("Red", false, cbg));
    add(new Checkbox("Blue", false, cbg));
    add(new Checkbox("Yellow", false, cbg));
    add(new Checkbox("Green", true, cbg));
    add(new Checkbox("Orange", false, cbg));
    add(new Checkbox("Purple", false, cbg));
  }
}
```

b. Create html file “CheckboxGroupTest.html”:

```
<HTML>
<HEAD>
<TITLE>Radio Buttons (exclusive)</TITLE>
</HEAD>
<BODY>
<H2>Lesson 8, Session 3: Radio Buttons</H2>
<P>
<APPLET CODE="CheckboxGroupTest.class" WIDTH=75 HEIGHT=175>
</APPLET>
<P>
<A HREF="CheckboxGroupTest.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and test the html file.

d. Change the radio buttons to “0-2 years old, 3-5 years old, 6-15 years old, 16-18 years old, 19-55 years old, 55+ years old”, and test the program again.

8.4. Choice Menus

a. Create applet “ChoiceTest.java”:

```
/* choice menus */
import java.awt.*;
public class ChoiceTest extends java.applet.Applet {
  public void init() {
    Choice c = new Choice();
    c.add("Apples");
    c.add("Oranges");
    c.add("Strawberries");
    c.add("Blueberries");
    c.add("Bananas");
    add(c);
  }
}
```

b. Create html file “ChoiceTest.html”:

```
<HTML>
<HEAD>
<TITLE>Choice Menus</TITLE>
</HEAD>
<BODY>
<H2>Lesson 8, Session 4: Choice Menus</H2>
<P>
<APPLET CODE="ChoiceTest.class" WIDTH=100 HEIGHT=150>
</APPLET>
<P>
<A HREF="ChoiceTest.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and test the html file.

d. Change the menu to “Chinses, French, Australian, Indian, Thai, Malay”, and test the program again.

8.5. Text Field

a. Create applet “TextFieldTest.java”:

```
/* text fields */
import java.awt.*;
public class TextFieldTest extends java.applet.Applet {
  public void init() {
       setLayout(new GridLayout(3,2,5,15));
    add(new Label("Enter your name:"));
    add(new TextField("your name here",45));
    add(new Label("Enter your phone number:"));
    add(new TextField(12));
    add(new Label("Enter your password:"));
    TextField t = new TextField(20);
    t.setEchoChar('*');
    add(t);
  }
}
```

b. Create html file “TextFieldTest.html”:

```
<HTML>
<HEAD>
<TITLE>Text Fields</TITLE>
</HEAD>
<BODY>
<H2>Lesson 8, Session 5: Text Fields</H2>
<P>
<APPLET CODE="TextFieldTest.class" WIDTH=325 HEIGHT=125>
</APPLET>
<P>
<A HREF="TextFieldTest.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and test the html file.

d. Test the program again by changing the text fields.

8.6. The FlowLayout class

a. Create applet “FlowLayoutTest.java”:

```
/* flowlayout test */

import java.awt.*;

public class FlowLayoutTest extends java.applet.Applet {

  public void init() {
       setLayout(new FlowLayout());
       add(new Button("One"));
    add(new Button("Two"));
    add(new Button("Three"));
    add(new Button("Four"));
    add(new Button("Five"));
    add(new Button("Six"));
  }
}
```

b. Create html file “`FlowLayoutTest`.html”:

```
<HTML>
<HEAD>
<TITLE>Flow Layout</TITLE>
</HEAD>
<BODY>
<H2>Lesson 8, Session 6: Flow Layout</H2>
<P>
<APPLET CODE="FlowLayoutTest.class" WIDTH=500 HEIGHT=200>
</APPLET>
<P>
<A HREF="FlowLayoutTest.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and test the html file.

d. Change the FlowLayout class function call of the `FlowLayoutTest.java program into the following and then test again:`

```
  (1) setLayout(new FlowLayout(FlowLayout.LEFT));
  (2) setLayout(new FlowLayout(FlowLayout.RIGHT));
  (3) setLayout(new FlowLayout(FlowLayout.LEFT, 30, 10));
```

8.7. Grid Layout

a. Create applet “`GridLayoutTest`.java”:

```
/* grid layouts */

import java.awt.*;
public class GridLayoutTest extends java.applet.Applet {
  public void init() {
       setLayout(new GridLayout(3,2));
```

```
      add(new Button("One"));
    add(new Button("Two"));
    add(new Button("Three"));
    add(new Button("Four"));
    add(new Button("Five"));
    add(new Button("Six"));
  }
}
```

b. Create html file “`GridLayoutTest`.html”:

```
<HTML>
<HEAD>
<TITLE>Grid Layout</TITLE>
</HEAD>
<BODY>
<H2>Lesson 8, Session 7: Grid Layout</H2>
<P>
<APPLET CODE="GridLayoutTest.class" WIDTH=500 HEIGHT=200>
</APPLET>
<P>
<A HREF="GridLayoutTest.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and test the html file.

d. Change the `GridLayout` class function call of the `GridLayoutTest.java program into the following and then test again:`

```
  setLayout(new GridLayout(3, 3,10,30));
```

8.8. The `Border` Layouts

a. Create applet “`BorderLayoutTest`.java”:

```
/* border layouts */

import java.awt.*;
public class BorderLayoutTest extends java.applet.Applet {
  public void init() {
      setLayout(new BorderLayout());
      add("North", new Button("One"));
    add("East", new Button("Two"));
    add("South", new Button("Three"));
    add("West", new Button("Four"));
    add("Center", new Button("Five"));
  }
}
```

b. Create html file “`BorderLayoutTest`.html”:

```
<HTML>
<HEAD>
<TITLE>Border Layout</TITLE>
</HEAD>
<BODY>
<H2>Lesson 8, Session 8: Border Layout</H2>
<P>
<APPLET CODE="BorderLayoutTest.class" WIDTH=500 HEIGHT=200>
</APPLET>
<P>
```

```
<A HREF="BorderLayoutTest.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and test the html file.

d. Change the `BorderLayout` class function call of the `BorderLayoutTest.java program into the following and then test again:`

```
  setLayout(new BorderLayout (10,10));
```

8.9. A Color Switcher

a. Create applet “`ButtonActionsTest`.java”:

```
/* button actions */

import java.awt.*;

public class ButtonActionsTest extends java.applet.Applet {
  Button redButton,blueButton,greenButton,whiteButton,blackButton;

  public void init() {
    setBackground(Color.white);
       setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));
       HandleButton he;
       redButton = new Button("Red");
       he = new HandleButton(this, Color.red);
       redButton.addActionListener(he);
    add(redButton);
       blueButton = new Button("Blue");
       he = new HandleButton(this, Color.blue);
       blueButton.addActionListener(he);
    add(blueButton);
       greenButton = new Button("Green");
       he = new HandleButton(this, Color.green);
       greenButton.addActionListener(he);
    add(greenButton);
       whiteButton = new Button("White");
       he = new HandleButton(this, Color.white);
       whiteButton.addActionListener(he);
    add(whiteButton);
       blackButton = new Button("Black");
       he = new HandleButton(this, Color.black);
       blackButton.addActionListener(he);
    add(blackButton);
  }
}
```

b. Create the HandleButton class:

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

public class HandleButton implements ActionListener {
   Color theColor;
   ButtonActionsTest theApp;
   HandleButton(ButtonActionsTest a, Color c) {
      theApp = a;
         theColor = c;
   }
   public void actionPerformed(ActionEvent e) {
      theApp.setBackground(theColor);
         theApp.repaint();
   }
}
```

c. Create html file “ButtonActionsTest.html”:

```
<HTML>
<HEAD>
<TITLE>Button Actions</TITLE>
</HEAD>
<BODY>
<H2>Lesson 8, Session 9: Button Actions</H2>
<P>
<APPLET CODE="ButtonActionsTest.class" WIDTH=250 HEIGHT=150>
</APPLET>
<P>
<A HREF="ButtonActionsTest.java">The Source</A>
<A HREF="HandleButton.java">The source for the event code</A>
</BODY>
</HTML>
```

d. Compile the applet and test the html file.

Lesson 9. Advanced Usage of the AWT

9.1. Text area

a. Create a Java applet “TextAreaTest.java”:

```
/* text areas */

import java.awt.*;
public class TextAreaTest extends java.applet.Applet {
  public void init() {
    String str = "Once upon a midnight dreary, while I pondered, weak and
weary,\n" +
     "Over many a quaint and curious volume of forgotten lore,\n" +
     "While I nodded, nearly napping, suddenly there came a tapping,\n" +
     "As of some one gently rapping, rapping at my chamber door.\n" +
     "\"'Tis some visitor,\" I muttered, \"tapping at my chamber door-\n" +
     "Only this, and nothing more.\"\n\n" +
     "Ah, distinctly I remember it was in the bleak December,\n" +
     "And each separate dying ember wrought its ghost upon the floor.\n" +
     "Eagerly I wished the morrow;- vainly I had sought to borrow\n" +
     "From my books surcease of sorrow- sorrow for the lost Lenore-\n" +
     "For the rare and radiant maiden whom the angels name Lenore-\n" +
     "Nameless here for evermore.";

     add(new TextArea(str));
  }
}
```

b. Create an HTML file “TextAreaTest.haml”:

```
<HTML>
<HEAD>
<TITLE>Text Area</TITLE>
</HEAD>
```

```
<BODY>
<H2>Lesson 9, Session 1: Text Area</H2>
<P>
<APPLET CODE="TextAreaTest.class" WIDTH=450 HEIGHT=200>
</APPLET>
<P>
<A HREF="TextAreaTest.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and test the html file

9.2. Scrolling a list of text.

a. Create the Java applet “`ListsTest.java`”:

```
/* scrolling lists */

import java.awt.*;
public class ListsTest extends java.applet.Applet {

  public void init() {
    List lst = new List(5, true);

        // this version is for 1.02; replace with add() for 1.1
    lst.add("Hamlet");
    lst.add("Claudius");
    lst.add("Gertrude");
    lst.add("Polonius");
    lst.add("Horatio");
    lst.add("Laertes");
    lst.add("Ophelia");

    add(lst);
  }
}
```

b. Create the HTML file “ListsTest.java”:

```
<HTML>
<HEAD>
<TITLE>Scrolling Lists</TITLE>
</HEAD>
<BODY>
<H2>Lesson 9,Session 2: Scrolling Lists</H2>
<P>
<APPLET CODE="ListsTest.class" WIDTH=200 HEIGHT=110>
</APPLET>
<P>
<A HREF="ListsTest.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and test the html file

9.3. Sliding bar

a. Create the Java applet “`SliderTest.java`”:

```
/* sliders (scrollbars) */
```

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

public class SliderTest extends java.applet.Applet
  implements AdjustmentListener {
  Label l;

  public void init() {
    setLayout(new GridLayout(1,2));
    l = new Label("1", Label.CENTER);
    add(l);
       Scrollbar sb = new
          Scrollbar(Scrollbar.HORIZONTAL,0,0,1,100);
       sb.addAdjustmentListener(this);
    add(sb);
  }

  public Insets getInsets() {
    return new Insets(15,15,15,15);
  }

  public void adjustmentValueChanged(AdjustmentEvent e) {
      int v = ((Scrollbar)e.getSource()).getValue();
      l.setText(String.valueOf(v));
         repaint();
  }
}
```

b. Create the HTML file “`SliderTest.haml`”:

```
<HTML>
<HEAD>
<TITLE>Sliders (scrollbars)</TITLE>
</HEAD>
<BODY>
<H2>Lesson 9, Session 3: Sliders (scrollbars)</H2>
<P>
<APPLET CODE="SliderTest.class" WIDTH=150 HEIGHT=50>
</APPLET>
<P>
<A HREF="SliderTest.java">The Source</A>
</BODY>
</HTML>
```

c. Compile the applet and test the html file

9.4. Pop-up window

a. Create the Java applet “`PopupWindow.java`”:

```
import java.awt.*;

public class PopupWindow extends java.applet.Applet {
   Frame window;
   Button open, close;

   public void init() {
      PopupActions handlebutton = new PopupActions(this);
      open = new Button("Open Window");
      open.addActionListener(handlebutton);
      add(open);
      close = new Button("Close Window");
      close.addActionListener(handlebutton);
      add(close);

      window = new BaseFrame1("A Popup Window");
      window.resize(150,150);
      window.show();
     }
}
```

b. Create class file "`PopupActions.java`":

```
/* actions for popup window */

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

public class PopupActions implements ActionListener {
PopupWindow theApp;

PopupActions(PopupWindow win) {
   theApp = win;
}

public void actionPerformed(ActionEvent e) {
   if (e.getSource() instanceof Button) {

      if (e.getSource() == theApp.open) {
         if (!theApp.window.isShowing())
            theApp.window.show();
      }
      else if (e.getSource() == theApp.close) {
         if (theApp.window.isShowing())
            theApp.window.hide();
      }
   }
}
}
```

c. Create class file "`BaseFrame1.java`":

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

class BaseFrame1 extends Frame {
  String message = "This is a Window";
  Label l;

  BaseFrame1(String title) {
     super(title);
     setLayout(new BorderLayout());
     l = new Label(message, Label.CENTER);
     l.setFont(new Font("Helvetica", Font.PLAIN, 12));
     add("Center", l);
   }

   public Insets getInsets() {
     return new Insets(20,0,25,0);
   }
}
```

d. Create HTML file "PopupWindow.html":

```
<HTML>
<HEAD>
<TITLE>A simple popup window</TITLE>
</HEAD>
<BODY>
<H2>Lesson 9, Session 4: A simple popup window</H2>
<P>
<APPLET CODE="PopupWindow.class" WIDTH=200 HEIGHT=100>
</APPLET>
<P>
<A HREF="PopupWindow.java">The Source for Popup Window</A><BR>
<A HREF="PopupActions.java">The Source for Popup actions</A><BR>
<A HREF="BaseFrame1.java">The Source for Base Frame</A>
</BODY>
</HTML>
```

e. Compile the applet and class files and test the html file

9.5. Pop-up window with a dialogue

a. Create Java applet “`PopupWindowDialog.java`”:

```
import java.awt.*;

public class PopupWindowDialog extends java.applet.Applet {
   Frame window;
   Button open, close;

   public void init() {
      PopupActions2 handlebutton = new PopupActions2(this);
      open = new Button("Open Window");
      open.addActionListener(handlebutton);
      add(open);

      close = new Button("Close Window");
      close.addActionListener(handlebutton);
      add(close);

      window = new BaseFrame2("A Popup Window");
      window.resize(150,150);
      window.show();
     }
}
```

b. Create class file “`PopupActions2.java`”:

```
/* actions for popup window */

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

public class PopupActions2 implements ActionListener {
   PopupWindowDialog theApp;
   PopupActions2(PopupWindowDialog win) {
      theApp = win;
   }

   public void actionPerformed(ActionEvent e) {
      if (e.getSource() instanceof Button) {
         if (e.getSource() == theApp.open) {
            if (!theApp.window.isShowing())
            theApp.window.show();
         }
         else if (e.getSource() == theApp.close) {
            if (theApp.window.isShowing())
               theApp.window.hide();
         }
      }
   }
}
```

c. Create class file "`BaseFrame2.java`":

```
import java.awt.*;

class BaseFrame2 extends Frame {
   String message = "This is a Window";
   TextDialog dl;
   Label l;

   BaseFrame2(String title) {
     super(title);
     setLayout(new BorderLayout());
     l = new Label(message, Label.CENTER);
     l.setFont(new Font("Helvetica", Font.PLAIN, 12));
     add("Center", l);
     // make a dialog for this window
     dl = new TextDialog(this, "Enter Text", true);
     dl.resize(150,100);

     Button b = new Button("Set Text");
     BaseFrameActions handlebutton = new BaseFrameActions(this);;
     b.addActionListener(handlebutton);
     add("South", b);
   }

   public Insets getInsets() {
     return new Insets(20,0,20,0);
   }
}
```

d. Create class file "`BaseFrameActions.java`":

```
/* actions for frame window */

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

public class BaseFrameActions implements ActionListener {
   BaseFrame2 theApp;
   BaseFrameActions(BaseFrame2 win) {
      theApp = win;
   }

   public void actionPerformed(ActionEvent e) {
      if (e.getSource() instanceof Button)
         theApp.dl.show();
   }
}
```

e. Create class file "`TextDialog.java`":

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

class TextDialog extends Dialog implements ActionListener {
  TextField tf;
  BaseFrame2 theFrame;

  TextDialog(Frame parent, String title, boolean modal) {
    super(parent, title, modal);

    theFrame = (BaseFrame2)parent;
    setLayout(new BorderLayout(10,10));
    setBackground(Color.white);
    tf = new TextField(theFrame.message,20);
    add("Center", tf);

    Button b = new Button("OK");
    b.addActionListener(this);
    add("South", b);
  }

  public Insets insets() {
    return new Insets(30,10,10,10);
  }

  public void actionPerformed(ActionEvent e) {
    if (e.getSource() instanceof Button) {
      String label = ((Button)e.getSource()).getLabel();
      if (label == "OK") {
         hide();
         theFrame.l.setText(tf.getText());
      }
    }
  }
}
```

f. Create HTML file “`PopupWindowDialog.html`”:

```
<HTML>
<HEAD>
<TITLE>A simple popup window with a dialog</TITLE>
</HEAD>
<BODY>
<H2>Lesson 9, Session 5: A simple popup window with a dialog</H2>
<P>
<APPLET CODE="PopupWindowDialog.class" WIDTH=200 HEIGHT=100>
</APPLET>
<P>
<A HREF="PopupWindowDialog.java">The Source for Popup Window</A><BR>
<A HREF="PopupActions2.java">The Source for Popup actions</A><BR>
<A HREF="BaseFrame2.java">The Source for Base Frame</A><BR>
<A HREF="BaseFrameActions.java">The Source for Base Frame actions</A><BR>
<A HREF="TextDialog.java">The Source for Text Dialog</A>
</BODY>
</HTML>
```

g. Compile the applet and class files and test the html file

9.6. Popup window with dialog and menu

a. Create Java applet “`PopupWindowMenu.java`”:

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

```
public class PopupWindowMenu extends java.applet.Applet {
   Frame window;
   Button open, close;

   public void init() {
      PopupActions3 handlebutton = new PopupActions3(this);

      open = new Button("Open Window");
      open.addActionListener(handlebutton);
      add(open);

      close = new Button("Close Window");
      close.addActionListener(handlebutton);
      add(close);

      window = new BaseFrame3("A Popup Window");
      window.resize(150,150);
      window.show();
     }
}
```

b. Create Java class “`PopupActions3.java`”:

```
/* actions for popup window */

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

public class PopupActions3 implements ActionListener {
   PopupWindowMenu theApp;
   PopupActions3(PopupWindowMenu win) {
      theApp = win;
   }

   public void actionPerformed(ActionEvent e) {
      if (e.getSource() instanceof Button) {
         if (e.getSource() == theApp.open) {
            if (!theApp.window.isShowing())
               theApp.window.show();
         }
         else if (e.getSource() == theApp.close) {
            if (theApp.window.isShowing())
               theApp.window.hide();
         }
      }
   }
}
```

c. Create Java class “`BaseFrame3.java`”:

```
import java.awt.*;

class BaseFrame3 extends Frame {
   String message = "This is a Window";
   TextDialog2 dl;
   Label l;

   BaseFrame3(String title) {
     super(title);
     setLayout(new BorderLayout());
     l = new Label(message, Label.CENTER);
     l.setFont(new Font("Helvetica", Font.PLAIN, 12));
     add("Center", l);

     // make a dialog for this window
     dl = new TextDialog2(this, "Enter Text", true);
     dl.resize(150,100);

     Button b = new Button("Set Text");
     BaseFrameActions2 handleact = new BaseFrameActions2(this);;
     b.addActionListener(handleact);
     add("South", b);

     MenuBar mb = new MenuBar();
     Menu m = new Menu("Colors");
     MenuItem redmi = new MenuItem("Red");
     redmi.addActionListener(handleact);
     m.add(redmi);
     MenuItem bluemi = new MenuItem("Blue");
     bluemi.addActionListener(handleact);
     m.add(bluemi);
     MenuItem greenmi = new MenuItem("Green");
     greenmi.addActionListener(handleact);
     m.add(greenmi);
     m.add(new MenuItem("-"));
     CheckboxMenuItem boldmi = new CheckboxMenuItem("Bold Text");
     boldmi.addActionListener(handleact);
     m.add(boldmi);
     mb.add(m);
     setMenuBar(mb);
  }

   public Insets getInsets() {
     return new Insets(20,0,25,0);
   }
}
```

d. Create Java class file “`BaseFrameActions2.java`”:

```
/* actions for frame window */

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

public class BaseFrameActions2 implements ActionListener {
   BaseFrame3 theApp;
   BaseFrameActions2(BaseFrame3 win) {
      theApp = win;
   }

   public void actionPerformed(ActionEvent e) {
      if (e.getSource() instanceof Button)
         theApp.dl.show();
      else if (e.getSource() instanceof MenuItem) {
         String label = ((MenuItem)e.getSource()).getLabel();
         if (label.equals("Red"))
            theApp.l.setBackground(Color.red);
         else if (label.equals("Blue"))
            theApp.l.setBackground(Color.blue);
         else if (label.equals("Green"))
            theApp.l.setBackground(Color.green);
         else if (label.equals("Bold Text")) {
            if (theApp.l.getFont().isPlain()) {
                theApp.l.setFont(new Font("Helvetica", Font.BOLD, 12));
            }
            else theApp.l.setFont(new Font("Helvetica", Font.PLAIN, 12));
         }
         theApp.repaint();
      }
  }
}
```

e. Create class file “`TextDialog2.java`”:

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

class TextDialog2 extends Dialog implements ActionListener {
  TextField tf;
  BaseFrame3 theFrame;

  TextDialog2(Frame parent, String title, boolean modal) {
    super(parent, title, modal);

    theFrame = (BaseFrame3)parent;
    setLayout(new BorderLayout(10,10));
    setBackground(Color.white);
    tf = new TextField(theFrame.message,20);
    add("Center", tf);

    Button b = new Button("OK");
    b.addActionListener(this);
    add("South", b);
  }

  public Insets insets() {
    return new Insets(30,10,10,10);
  }

  public void actionPerformed(ActionEvent e) {
    if (e.getSource() instanceof Button) {
      String label = ((Button)e.getSource()).getLabel();
      if (label == "OK") {
         hide();
         theFrame.l.setText(tf.getText());
      }
    }
  }
}
```

f. Create HTML file “`PopupWindowMenu.html`”:

```
<HTML>
<HEAD>
<TITLE>A simple popup window with menus</TITLE>
</HEAD>
<BODY>
<H2>Lesson 9, Session 6: A simple popup window with menus</H2>
<P>
<APPLET CODE="PopupWindowMenu.class" WIDTH=200 HEIGHT=100>
</APPLET>
<P>
<A HREF="PopupWindowMenu.java">The Source for Popup Window</A><BR>
<A HREF="PopupActions3.java">The Source for popup actions</A><BR>
<A HREF="BaseFrame3.java">The Source for Base Frame</A><BR>
```

```
<A HREF="BaseFrameActions2.java">The Source for Base Frame actions</A><BR>
<A HREF="TextDialog2.java">The Source for TextDialog</A>
</BODY>
</HTML>
```

g. Compile the applet and class files and test the html file

9.7. Creating links inside Applets

a. Create the “ButtonLink.java” applet:

```
import java.awt.*;
import java.net.URL;
import java.net.MalformedURLException;

public class ButtonLink extends java.applet.Applet {
   Bookmark bmlist[] = new Bookmark[3];
   public void init() {
       bmlist[0] = new Bookmark("Wanlei’s Home Page",
            "http://www.cm.deakin.edu.au/~wanlei/");
        bmlist[1] = new Bookmark("Wsoft",
            "http://www.wsoft.com.au");
        bmlist[2]= new Bookmark("Java Home Page",
            "http://java.sun.com");

        setLayout(new GridLayout(bmlist.length,1, 10, 10));
        for (int i = 0; i < bmlist.length; i++) {
            add(new Button(bmlist[i].name));
        }
    }
    public boolean action(Event evt, Object arg) {
        if (evt.target instanceof Button) {
            LinkTo((String)arg);
            return true;
        }
        else return false;
    }
    void LinkTo(String name) {
        URL theURL = null;
        for (int i = 0; i < bmlist.length; i++) {
            if (name.equals(bmlist[i].name))
                theURL = bmlist[i].url;
        }
        if (theURL != null)
            getAppletContext().showDocument(theURL);
    }
}
```

b. Create “Bookmark.java” class:

```
import java.net.URL;
import java.net.MalformedURLException;

class Bookmark {
    String name;
    URL url;

    Bookmark(String name, String theURL) {
        this.name = name;
        try { this.url = new URL(theURL); }
        catch ( MalformedURLException e) {
            System.out.println("Bad URL: " + theURL);
        }
    }
}
```

c. Create “ButtonLink.html” file:

```
<HTML>
<HEAD>
<TITLE>Button links in Java</TITLE>
</HEAD>
<BODY>
<H2>Lesson 9, Session 7: Button links in Java</H2>
<P>
<APPLET CODE="ButtonLink.class" WIDTH=200 HEIGHT=100>
</APPLET>
<P>
<A HREF="ButtonLink.java">The Source for Button Link</A><BR>
<A HREF="Bookmark.java">The Source for Bookmark</A>
</BODY>
</HTML>
```

d. Compile the Java files and test the html file.

e. Change the URLs and test again.

Lesson 10. Networking in Java

10.1. Simple Client/Server communication using sockets: Local Host

a. Create the server Java program “S.java”:

```
// An echo server
// Usage: S [port]
// The default port is 6789
// The Server program should be run first

import java.net.*;
import java.io.*;
public class S {
  public final static int DEFAULT_PORT = 6789;
  public static void main (String args[]) throws IOException {
    Socket client;
    if (args.length != 1)
      client = accept (DEFAULT_PORT);
    else
      client = accept (Integer.parseInt (args[0]));
    try {
      PrintWriter writer;
      BufferedReader reader;
       reader = new BufferedReader(new
InputStreamReader(client.getInputStream()));
       writer = new PrintWriter(new
OutputStreamWriter(client.getOutputStream()));
       // read a line
       String line = reader.readLine();
       System.out.println("Client says: " + line);
       // write a line
      writer.println ("You have connected to the Very Simple Server.");
       writer.flush();
       reader.close();
       writer.close();
    } finally { // closing down the connection
      System.out.println ("Closing");
      client.close ();
    }
  }
  static Socket accept (int port) throws IOException {
    System.out.println ("Starting on port " + port);
    ServerSocket server = new ServerSocket (port);
    System.out.println ("Waiting");
    Socket client = server.accept ();
    System.out.println ("Accepted from " + client.getInetAddress ());
    server.close ();
    return client;
  }
}
```

b. Create the client Java program “C.java”:

```
// A client of an echo server
// Usage: java C <hostname> [<port>]
// The Server program should be run first

import java.io.*;
import java.net.*;
public class C {
  public static final int DEFAULT_PORT = 6789;
  public static void usage() {
    System.out.println("Usage: java C [<port>]");
    System.exit(0);
  }
  public static void main(String[] args) {
    int port = DEFAULT_PORT;
    Socket s = null;
    // parse the port specification
    if ((args.length != 0) && (args.length != 1)) usage();
    if (args.length == 0) port = DEFAULT_PORT;
    else {
      try {
        port = Integer.parseInt(args[0]);
      }
      catch(NumberFormatException e) {
        usage();
      }
    }
    try {
      BufferedReader reader;
      PrintWriter writer;
      // create a socket to communicate to the specified host and port
      s = new Socket("localhost", port);
      // create streams for reading and writing
       reader = new BufferedReader(new InputStreamReader(s.getInputStream()));
       writer = new PrintWriter(new OutputStreamWriter(s.getOutputStream()));
      // tell the user that we've connected
      System.out.println("Connected to " + s.getInetAddress() +
        ":" + s.getPort());
      String line;
       // write a line to the server
      writer.println("Hello, Server");
       writer.flush();
      // read the response (a line) from the server
      line = reader.readLine();
      // write the line to console
      System.out.println("Server says: " + line);
       reader.close();
       writer.close();
    }
    catch (IOException e) {
      System.err.println(e);
    }
    // always be sure to close the socket
    finally {
      try {
        if (s != null) s.close();
      }
      catch (IOException e2) { }
    }
  }
}
```

c. Compile both programs and test run them. Start the server first, then the client. They should be on the local machines.

10.2. Exchange of multiple messages

a. Create the server Java program “S1.java”:

```
// An echo server
// Usage: S1 [port]
// The default port is 6789
// The Server program should be run first

import java.net.*;
import java.io.*;

public class S1 {
  public final static int DEFAULT_PORT = 6789;

  public static void main (String args[]) throws IOException {
    Socket client;
    if (args.length != 1)
      client = accept (DEFAULT_PORT);
    else
      client = accept (Integer.parseInt (args[0]));

    try {
      PrintWriter writer;
      BufferedReader reader;
       reader = new BufferedReader(new
InputStreamReader(client.getInputStream()));
       writer = new PrintWriter(new
OutputStreamWriter(client.getOutputStream()));
      writer.println ("You are now connected to the Simple Echo Server.");
       writer.flush();
      for (;;) {
         // read a line
         String line = reader.readLine();
        // and send back ACK
         writer.println("OK");
         writer.flush();
         System.out.println("Client says: " + line);
         if (line.equals("Server Exit")) {
           break;
        }
      }
       reader.close();
      writer.close();
    } finally {
      System.out.println ("Closing");
      client.close ();
    }
  }

  static Socket accept (int port) throws IOException {
    System.out.println ("Starting on port " + port);
    ServerSocket server = new ServerSocket (port);

    System.out.println ("Waiting");
    Socket client = server.accept ();
    System.out.println ("Accepted from " + client.getInetAddress ());

    server.close ();
    return client;
  }
}
```

b. Create the client Java program “C1.java”:

```
// A client of an echo server (localhost)
// Usage: java C1 [<port>]
// The Server program should be run first
// For Java 2

import java.io.*;
import java.net.*;

public class C1 {
  public static final int DEFAULT_PORT = 6789;
  public static void usage() {
    System.out.println("Usage: java C1 [<port>]");
    System.exit(0);
  }

  public static void main(String[] args) {
    int port = DEFAULT_PORT;
    Socket s = null;
    int end = 0;

    // parse the port specification
    if ((args.length != 0) && (args.length != 1)) usage();
    if (args.length == 0) port = DEFAULT_PORT;
    else {
      try {
        port = Integer.parseInt(args[0]);
      }
      catch(NumberFormatException e) {
         usage();
      }
    }

    try {
      PrintWriter writer;
      BufferedReader reader;
      BufferedReader kbd;
      // create a socket to communicate to the specified host and port
      //InetAddress myhost = getLocalHost();
      s = new Socket("localhost", port);
      // create streams for reading and writing
       reader = new BufferedReader(new InputStreamReader(s.getInputStream()));
       OutputStream sout = s.getOutputStream();
       writer = new PrintWriter(new OutputStreamWriter(s.getOutputStream()));
      // create a stream for reading from keyboard
      kbd = new BufferedReader(new InputStreamReader(System.in));
      // tell the user that we've connected
      System.out.println("Connected to " + s.getInetAddress() +
        ":" + s.getPort());

      String line;
      // read the first response (a line) from the server
      line = reader.readLine();
      // write the line to console
      System.out.println(line);
      while (true) {
        // print a prompt
        System.out.print("> ");
        System.out.flush();
        // read a line from console, check for EOF
        line = kbd.readLine();
        if (line.equals("Server Exit")) end = 1;
        // send it to the server
        writer.println(line);
        writer.flush();

        // read a line from the server
        line = reader.readLine();
        // check if connection is closed, i.e., EOF
        if (line == null) {
          System.out.println("Connection closed by server.");
          break;
        }
        if (end == 1) {
           break;
         }
        // write the line to console
        System.out.println("Server says: " + line);
      }
       reader.close();
       writer.close();
    }
    catch (IOException e) {
      System.err.println(e);
    }
    // always be sure to close the socket
    finally {
      try {
        if (s != null) s.close();
      }
      catch (IOException e2) { }
    }
  }
}
```

c. Compile both programs and test run them. Start the server first, then the client. They should be on the same local machines.

10.3. Executing programs on Internet hosts

a. Create the Java program “InetExample.java” for displaying address information of an Internet host:

```
import java.net.*;
import java.io.*;
```

```
public class InetExample {
  public static void main (String args[]) {
    printLocalAddress ();
    Reader kbd = new FileReader (FileDescriptor.in);
    BufferedReader bufferedKbd = new BufferedReader (kbd);
    try {
      String name;
      do {
        System.out.print ("Enter a hostname or IP address: ");
        System.out.flush ();
        name = bufferedKbd.readLine ();
        if (name != null)
          printRemoteAddress (name);
      } while (name != null);
      System.out.println ("exit");
    } catch (IOException ex) {
      System.out.println ("Input error:");
      ex.printStackTrace ();
    }
  }
  static void printLocalAddress () {
    try {
      InetAddress myself = InetAddress.getLocalHost ();
      System.out.println ("My name : " + myself.getHostName ());
      System.out.println ("My IP : " + myself.getHostAddress ());
      System.out.println ("My class : " + ipClass (myself.getAddress ()));
    } catch (UnknownHostException ex) {
      System.out.println ("Failed to find myself:");
      ex.printStackTrace ();
    }
  }
  static char ipClass (byte[] ip) {
    int highByte = 0xff & ip[0];
    return (highByte < 128) ? 'A' : (highByte < 192) ? 'B' :
      (highByte < 224) ? 'C' : (highByte < 240) ? 'D' : 'E';
  }
  static void printRemoteAddress (String name) {
    try {
      System.out.println ("Looking up " + name + "...");
      InetAddress machine = InetAddress.getByName (name);
      System.out.println ("Host name : " + machine.getHostName ());
      System.out.println ("Host IP : " + machine.getHostAddress ());
      System.out.println ("Host class : " +
                          ipClass (machine.getAddress ()));
    } catch (UnknownHostException ex) {
      System.out.println ("Failed to lookup " + name);
    }
  }
}
```

b. Create the client Java program “C2.java”:

```
// A client of an echo server (localhost)
// Usage: java C2 <serverhost>
// The Server program should be run first
// For Java 2

import java.io.*;
import java.net.*;

public class C2 {
  public static final int DEFAULT_PORT = 6789;
  public static void usage() {
    System.out.println("Usage: java C2 <serverhost>");
    System.exit(0);
  }

  public static void main(String[] args) {
    int port = DEFAULT_PORT;
    String address = "";
    Socket s = null;
    int end = 0;

    // parse the port specification
    if ((args.length != 0) && (args.length != 1)) usage();
    if (args.length == 0) {
      port = DEFAULT_PORT;
       address = "localhost";
    } else {
        address = args[0];
    }

    try {
      PrintWriter writer;
      BufferedReader reader;
      BufferedReader kbd;
      // create a socket to communicate to the specified host and port
      s = new Socket(address, port);
      // create streams for reading and writing
       reader = new BufferedReader(new InputStreamReader(s.getInputStream()));
       OutputStream sout = s.getOutputStream();
       writer = new PrintWriter(new OutputStreamWriter(s.getOutputStream()));
      // create a stream for reading from keyboard
      kbd = new BufferedReader(new InputStreamReader(System.in));
      // tell the user that we've connected
      System.out.println("Connected to " + s.getInetAddress() +
        ":" + s.getPort());

      String line;
      // read the first response (a line) from the server
      line = reader.readLine();
      // write the line to console
      System.out.println(line);
      while (true) {
        // print a prompt
        System.out.print("> ");
        System.out.flush();
        // read a line from console, check for EOF
        line = kbd.readLine();
        if (line.equals("Server Exit")) end = 1;
        // send it to the server
        writer.println(line);
        writer.flush();

        // read a line from the server
        line = reader.readLine();
        // check if connection is closed, i.e., EOF
        if (line == null) {
          System.out.println("Connection closed by server.");
          break;
        }
        if (end == 1) {
           break;
         }
        // write the line to console
        System.out.println("Server says: " + line);
      }
       reader.close();
       writer.close();
    }
    catch (IOException e) {
      System.err.println(e);
    }
    // always be sure to close the socket
    finally {
      try {
        if (s != null) s.close();
      }
      catch (IOException e2) { }
    }
  }
}
```

c. Compile both programs and test run them. The “InetExample.java” program displays the information of any Internet host. The “C2.java” program can run with the “S1.java” program of the previous section. You may start the S1.java program on any Internet host, then the C2.java program on any Internet host as well. The two programs can communicate with each other.

10.4. Supporting multiple clients

a. Create the server Java program “S3.java”:

```
// An echo server
// Usage: S3 [port]
// The default port is 6789
// The Server program should be run first

import java.net.*;
import java.io.*;

public class S3 extends Thread {
  public final static int DEFAULT_PORT = 6789;
  private Socket client = null;

  public S3(Socket inSock) {
    super("echoServer");
    client = inSock;
  }

  public void run() {
    Socket cSock = client;
    PrintWriter writer;
    BufferedReader reader;
    try {
       String line;
       System.out.println ("Accepted from " + cSock.getInetAddress());
       reader = new BufferedReader(new
InputStreamReader(cSock.getInputStream()));
       writer = new PrintWriter(new
OutputStreamWriter(cSock.getOutputStream()));
      writer.println ("You are now connected to the Simple Echo Server.");
       writer.flush();
      for (;;) {
         // read a line
         line = reader.readLine();
        // and send back ACK
         writer.println("OK");
         writer.flush();
```

```
          System.out.println("Client says: " + line);
          if (line.equals("Server Exit") || line.equals("Client Exit")) break;
        }
        System.out.println ("Closing the client " + cSock.getInetAddress());
        reader.close();
         writer.close();
        cSock.close ();
         if (line.equals("Server Exit")) {
          System.out.println ("Closing the server");
          // server.close ();
           System.exit(0);
         }
      } catch (IOException e1) {
        System.err.println("Exception: " + e1.getMessage());
         System.exit(1);
      }
    }

    public static void main (String args[]) {
      ServerSocket server = null;
      try {
        server = new ServerSocket(DEFAULT_PORT);
        System.out.println ("Starting on port " + DEFAULT_PORT);
      } catch (IOException e) {
        System.err.println("Exception: could't make server socket.");
         System.exit(1);
      }
      while (true) {
        Socket incomingSocket = null;
        // wait fot a connection request
        System.out.println("Waiting...");
        try {
          incomingSocket = server.accept();
          // call a thread to deal with a connection
          S3 es = new S3(incomingSocket);
          es.start();
        } catch (IOException e) {
          System.err.println("Exception: could't make server socket.");
           System.exit(1);
        }
      }
    }
}
```

b. Compile the program and test run it together with the C2.java program of the previous section. Start the server S3.java first, then the client C2.java. They can run in different machines. You can start a number of C2.java programs on many host, communicating with the server simultaneously.

Lesson 11. Java Database Connectivity (JDBC)

11.1. Prepare the Access database for JDBC

a. Create a blank Access database, called “dbtest.mdb”

b. Start → Setings → Control Panel → ODBC Data Sources (32bit) → System DSN → Add → Microsoft Database Driver (*.mdb) → Type in data source name and use “Select” to find the database “dbtest.mdb”; Use “Options” to set the username and password.

11.2. Create the CUSTOMER table

a. Create Java program “CreateCustomer.java”:

```
import java.sql.*;

public class CreateCustomer {
      public static void main(String args[]) {
            String url = "jdbc:odbc:dbtest";
            Connection con;
            String createString;
            createString = "create table CUSTOMER " +
                  "(C_NAME varchar(30), " +
                  "C_ID int, " +
                  "C_ADDR varchar(50), " +
                  "C_PHONE varchar(12) )";
            Statement stmt;
            try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
            } catch(Exception e) {
                  System.err.print("ClassNotFoundException: ");
                  System.err.println(e.getMessage());
            }
            try {
                  con = DriverManager.getConnection(url, "admin", "admin");
                  stmt = con.createStatement();

                  stmt.executeUpdate(createString);
                  stmt.close();
                  con.close();
            } catch(SQLException ex) {
                  System.err.println("SQLException: " + ex.getMessage());
            }
      }
}
```

b. Compile the program and run it. Check the database to see if the table is created.

11.3. Create the PRODUCT table

a. Create Java program “CreateProduct.java”:

```
import java.sql.*;

public class CreateProduct {
      public static void main(String args[]) {
            String url = "jdbc:odbc:dbtest";
            Connection con;
            String createString;
            createString = "create table PRODUCT " +
                  "(P_NAME varchar(20), " +
                  "P_DESC varchar(40), " +
                  "P_CODE varchar(8), " +
                  "P_UNIT_PRICE float, " +
                  "P_STOCK int )";
            Statement stmt;
            try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
            } catch(Exception e) {
                  System.err.print("ClassNotFoundException: ");
                  System.err.println(e.getMessage());
            }
```

```
		try {
			con = DriverManager.getConnection(url, "admin", "admin");
			stmt = con.createStatement();

			stmt.executeUpdate(createString);
			stmt.close();
			con.close();
		} catch(SQLException ex) {
			System.err.println("SQLException: " + ex.getMessage());
		}
	}
}
```

b. Compile the program and run it. Check the database to see if the table is created.

11.4. Create the TRANSACTION table

a. Create Java program “CreateTransaction.java”:

```
import java.sql.*;

public class CreateTransaction {
	public static void main(String args[]) {
		String url = "jdbc:odbc:dbtest";
		Connection con;
		String createString;
		createString = "create table TRANSACTION " +
			"(T_ID int, " +
			"C_ID int, " +
			"P_CODE varchar(8), " +
			"T_NUM int, " +
			"T_TOTAL_PRICE float, " +
			"T_DATE date )";
		Statement stmt;
		try {
		Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
		} catch(Exception e) {
			System.err.print("ClassNotFoundException: ");
			System.err.println(e.getMessage());
		}
		try {
			con = DriverManager.getConnection(url, "admin", "admin");
			stmt = con.createStatement();

			stmt.executeUpdate(createString);
			stmt.close();
			con.close();
		} catch(SQLException ex) {
			System.err.println("SQLException: " + ex.getMessage());
		}
	}
}
```

b. Compile the program and run it. Check the database to see if the table is created.

11.5. Populate the three tables

a. Create Java program “InsertCustomer.java”:

```
import java.sql.*;
public class InsertCustomer {
	public static void main(String args[]) {
		String url = "jdbc:odbc:dbtest";
		Connection con;
		Statement stmt;
		String query = "select * from CUSTOMER";
		try {
			Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

		} catch(java.lang.ClassNotFoundException e) {
			System.err.print("ClassNotFoundException: ");
			System.err.println(e.getMessage());
		}
		try {
			con = DriverManager.getConnection(url, "admin", "admin");
			stmt = con.createStatement();

			stmt.executeUpdate("insert into CUSTOMER " +
			  "values('John Smith', 100, '123 King St.', '03-9123
4567')");
			stmt.executeUpdate("insert into CUSTOMER " +
			  "values('Alex Lee', 101, '234 Queen St.', '03-9234
5678')");
			stmt.executeUpdate("insert into CUSTOMER " +
			  "values('Anne Wong', 102, '345 Yarra Ave.', '03-9345
6789')");
			stmt.executeUpdate("insert into CUSTOMER " +
			  "values('Tanya Foo', 103, '456 Irving Rd.', '03-9456
7890')");

			ResultSet rs = stmt.executeQuery(query);
			System.out.println("C_NAME C_ID C_ADDR C_PHONE");
			while (rs.next()) {
				String s = rs.getString("C_NAME");
				int i = rs.getInt("C_ID");
				String s1 = rs.getString("C_ADDR");
				String s2 = rs.getString("C_PHONE");
				System.out.println(s + "  " + i +
					"  " + s1 + "  " + s2);
			}
			stmt.close();
			con.close();
		} catch(SQLException ex) {
			System.err.println("SQLException: " + ex.getMessage());
		}
	}
}
```

b. Compile the program and run it. Check the database to see if the table is populated.

c. Create Java program “InsertProduct.java”:

```
import java.sql.*;
public class InsertProduct {
	public static void main(String args[]) {
		String url = "jdbc:odbc:dbtest";
		Connection con;
		Statement stmt;
		String query = "select * from PRODUCT";
		try {
			Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

		} catch(java.lang.ClassNotFoundException e) {
			System.err.print("ClassNotFoundException: ");
			System.err.println(e.getMessage());
		}
		try {
			con = DriverManager.getConnection(url, "admin", "admin");
			stmt = con.createStatement();

			stmt.executeUpdate("insert into PRODUCT " +
			  "values('TV', 'Philip, 68cm, flat screen', 'T0010',
1200.00, 10)");
			stmt.executeUpdate("insert into PRODUCT " +
			  "values('VCR', 'Sony, Mid-Drive', 'V100', 500.00, 15)");
			stmt.executeUpdate("insert into PRODUCT " +
			  "values('TV', 'Tohisba, 34cm, remote control', 'T0012',
300.00, 20)");
			stmt.executeUpdate("insert into PRODUCT " +
			  "values('PC', 'Dell, 256M RAM, 10GHD, 17\" monitor',
'P0012', 2400.00, 12)");

			ResultSet rs = stmt.executeQuery(query);
			System.out.println("P_NAME  P_DESC  P_CODE  P_UNIT_PRICE
P_STOCK");
			while (rs.next()) {
				String s = rs.getString("P_NAME");
				String s1 = rs.getString("P_DESC");
				String s2 = rs.getString("P_CODE");
				float f = rs.getFloat("P_UNIT_PRICE");
				int i = rs.getInt("P_STOCK");
				System.out.println(s + "  " + s1 + "  " + s2 +
					"  " + f + "  " + i);
			}
			stmt.close();
			con.close();
		} catch(SQLException ex) {
			System.err.println("SQLException: " + ex.getMessage());
		}
	}
}
```

d. Compile the program and run it. Check the database to see if the table is populated.

e. Create Java program “InsertTransaction.java”:

```
import java.sql.*;
public class InsertTransaction {
	public static void main(String args[]) {
		String url = "jdbc:odbc:dbtest";
		Connection con;
		Statement stmt;
		String query = "select * from TRANSACTION";
		try {
			Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

		} catch(java.lang.ClassNotFoundException e) {
			System.err.print("ClassNotFoundException: ");
			System.err.println(e.getMessage());
		}
		try {
			con = DriverManager.getConnection(url, "admin", "admin");
			stmt = con.createStatement();

			stmt.executeUpdate("insert into TRANSACTION " +
			  "values(500, 100, 'T0010', 1, 1200.00, #1/8/2000#)");
```

```
                    stmt.executeUpdate("insert into TRANSACTION " +
                      "values(501, 101, 'V100', 2, 1000.00, #2/20/2000#)");

                    ResultSet rs = stmt.executeQuery(query);
                    System.out.println("T_ID  C_ID  P_CODE  T_NUM
T_TOTAL_PRICE  T_DATE");
                    while (rs.next()) {
                          int i = rs.getInt("T_ID");
                          int i1 = rs.getInt("C_ID");
                          String s = rs.getString("P_CODE");
                          int i2 = rs.getInt("T_NUM");
                          float f = rs.getFloat("T_TOTAL_PRICE");
                          Date d = rs.getDate("T_DATE");
                          System.out.println(i + "  " + i1 + "  " + s +
                                "  " + i2 + "  " + f + "  " + d);
                    }
                    stmt.close();
                    con.close();
             } catch(SQLException ex) {
                    System.err.println("SQLException: " + ex.getMessage());
             }
      }
}
```

d. Compile the program and run it. Check the database to see if the table is populated.

11.6. Print all column of the three tables

.a. Create Java program “PrintColumns.java”:

```
import java.sql.*;
class PrintColumns  {
      public static void main(String args[]) {
             String url = "jdbc:odbc:dbtest";
             Connection con;
             String query = "select * from CUSTOMER";
             Statement stmt;
             try {
                    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
             } catch(java.lang.ClassNotFoundException e) {
                    System.err.print("ClassNotFoundException: ");
                    System.err.println(e.getMessage());
             }
             try {
                    con = DriverManager.getConnection(url, "admin", "admin");
                    stmt = con.createStatement();

                    ResultSet rs = stmt.executeQuery(query);
                    ResultSetMetaData rsmd = rs.getMetaData();
                    PrintColumnTypes.printColTypes(rsmd);
                    System.out.println("");
                    int numberOfColumns = rsmd.getColumnCount();
                    for (int i = 1; i <= numberOfColumns; i++) {
                          if (i > 1) System.out.print(",  ");
                          String columnName = rsmd.getColumnName(i);
                          System.out.print(columnName);
                    }
                    System.out.println("");
                    while (rs.next()) {
                          for (int i = 1; i <= numberOfColumns; i++) {
                                if (i > 1) System.out.print(",  ");
                                String columnValue = rs.getString(i);
                                System.out.print(columnValue);
                        }
                        System.out.println("");
                }
                stmt.close();
                con.close();
        } catch(SQLException ex) {
                System.err.print("SQLException: ");
                System.err.println(ex.getMessage());
        }
    }
}
```

b. Compile the program and run it. Check the database to see if the table is printed properly.

c. Create Java programs to print the columns of the PRODUCT and the TRANSACTION tables.

11.7. Select statement (one table)

a. Create Java program “SelectStatement.java”:

```
import java.sql.*;
public class SelectStatement  {
        public static void main(String args[]) {
                String url = "jdbc:odbc:dbtest";
                Connection con;
                String query = "select P_DESC, P_STOCK " +
                   "from PRODUCT " +
                   "where P_NAME like 'TV'";
                Statement stmt;
                try {
                        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                } catch(java.lang.ClassNotFoundException e) {
                        System.err.print("ClassNotFoundException: ");
                        System.err.println(e.getMessage());
                }
                try {
                        con = DriverManager.getConnection(url, "admin", "admin");
                        stmt = con.createStatement();

                        ResultSet rs = stmt.executeQuery(query);
                        ResultSetMetaData rsmd = rs.getMetaData();
                        int numberOfColumns = rsmd.getColumnCount();
                        int rowCount = 1;
                        while (rs.next()) {
                                System.out.println("Row " + rowCount + ":  ");
                                for (int i = 1; i <= numberOfColumns; i++) {
                                        System.out.print("   Column " + i + ":  ");
                                        System.out.println(rs.getString(i));
                                }
                                System.out.println("");
                                rowCount++;
                        }
                        stmt.close();
                        con.close();

                } catch(SQLException ex) {
                        System.err.print("SQLException: ");
                        System.err.println(ex.getMessage());
                }
        }
}
```

b. Compile the program and run it. Check the database to see if the result is selected properly.

Lesson 12. Putting All Together: A Project

This project has a database that stores data, a server that manages the access of the database, and a client that interfaces with users. The client uses Java applet to access the server and the server uses JDBC to access the database.

12.1. Prepare the Access database and the HTML file

a. Use the Access database, “dbtest.mdb”, created in the last chapter. It includes three tables: CUSTOMER, PRODUCT, and TRANSACT
ION.

b. Prepare the following HTML file:

```
<HTML>
<title>Database Operations</title>

<applet code="ClientApplet.class"
       width=600 height=350>
</applet>
</HTML>
```

12.2. Prepare the Java applet programs

a. Create the main applet program, “ClientApplet.java”. This program implements the user interface.

```
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;
import java.io.*;

public class ClientApplet extends Applet {
  private static final String host= "192.168.0.1";
  TextArea ta;
  ClientComm cc;
  ClientCommExit cce;
  ClientCommSQL ccs;
  TextField sqlcommand;

  public void init () {
    Panel p1 = new Panel(new BorderLayout(10, 10));
    Button p1b1 = new Button ("Resulsts Returned");
    p1.add (p1b1, BorderLayout.NORTH);
    ta = new TextArea ();
    ta.setEditable(false);
    p1.add (ta, BorderLayout.CENTER);
    sqlcommand = new TextField ("", 50);
    p1.add (sqlcommand, BorderLayout.SOUTH);
    add (p1);

    Panel p2 = new Panel (new FlowLayout());
    Button p2bCus = new Button ("All Customers");
    p2.add (p2bCus);
    p2bCus.addActionListener (new ActionListener() {
      public void actionPerformed (ActionEvent e) {
        ByteArrayOutputStream bao = new ByteArrayOutputStream();
        cc = new ClientComm(host, 0, 1, bao);
        ta.setText (bao.toString()+"Returned by the <All Customer> request");
      }
    } );

    Button p2bPro = new Button ("All Products");
    p2.add (p2bPro);
    p2bPro.addActionListener (new ActionListener() {
      public void actionPerformed (ActionEvent e) {
        ByteArrayOutputStream bao = new ByteArrayOutputStream();
        cc = new ClientComm(host, 0, 2, bao);
        ta.setText (bao.toString()+"Returned by the <All Product> request");
      }
    } );

    Button p2bTra = new Button ("All Transactions");
    p2.add (p2bTra);
    p2bTra.addActionListener (new ActionListener() {
      public void actionPerformed (ActionEvent e) {
        ByteArrayOutputStream bao = new ByteArrayOutputStream();
        cc = new ClientComm(host, 0, 3, bao);
        ta.setText (bao.toString()+"Returned by the <All Transaction>
request");
      }
    } );

    Button p2bSQL = new Button ("SQL Command");
    p2.add (p2bSQL);
    p2bSQL.addActionListener (new ActionListener() {
      public void actionPerformed (ActionEvent e) {
        ByteArrayOutputStream bao = new ByteArrayOutputStream();
        ccs = new ClientCommSQL(host, 0, 4, sqlcommand.getText(), bao);
        ta.setText (bao.toString()+"Returned by the <SQL Command> request");
      }
    } );

    Button p2bExit = new Button ("ShutDown Server");
    p2.add (p2bExit);
    p2bExit.addActionListener (new ActionListener() {
      public void actionPerformed (ActionEvent e) {
        ByteArrayOutputStream bao = new ByteArrayOutputStream();
        cce = new ClientCommExit(host, 0, bao);
        ta.setText (bao.toString()+"Returned by the <ShutDown Server>
request");
        //System.exit(0);
      }
    } );

    add (p2);
  }
}
```

b. Create the Java program that implements the “ClientComm” class used in the applet: “ClientComm.java”. This program deals with the major communication work between the applet and the server.

```
import java.io.*;
import java.net.*;

public class ClientComm {
  public static final int DEFAULT_PORT = 6789;
  private String host = "";
  private int port = 0;
  private OutputStream os = null;
  boolean DEBUG = true;

  public ClientComm (String h, int p, int choice, OutputStream o) {
    host = h;
    port = ((p == 0) ? DEFAULT_PORT : p);
    os = o;
    Socket s = null;
    PrintWriter out = new PrintWriter (os, true);
    if (DEBUG) {
      System.out.println("Applet about to create a socket on "
          + host + " at port " + port);
    }

    try {
      // create a socket to communicate to the specified host and port
      s = new Socket(host, port);
      // create streams for reading and writing
      BufferedReader sin = new BufferedReader(new
InputStreamReader(s.getInputStream()));
      PrintStream sout = new PrintStream(s.getOutputStream(), true);
      if (DEBUG) {
        System.out.println("Applet has created sin and sout ");
      }

      // tell the user that we've connected
      out.println("Connected to " + s.getInetAddress() +
        ":" + s.getPort());

      if (DEBUG) {
        System.out.println("Applet has connected to "+ s.getInetAddress() +
            ":" + s.getPort());
      }

      String line;
      // read the first response (a line) from the server
      line = sin.readLine();
      if (DEBUG) {
        System.out.println("Applet has read a line: " + line);
      }

      // write the line to the user
      out.println(line);
      out.flush();
      // send the command choice to the server
      if (choice <=3) {
        sout.println(choice);
      } else {
        sout.println("Wrong command");
      }
      if (DEBUG) {
        System.out.println("Applet has sent sout the choice: "+ choice);
      }

      // read a line from the server
      line = sin.readLine();
      if (DEBUG) {
        System.out.println("Applet has read a line: " + line);
      }

      out.println(line);
      // check if connection is closed, i.e., EOF
      if (line == null) {
        out.println("Connection closed by server.");
      }
       while (true) {
         line=sin.readLine();
         if (DEBUG) {
           System.out.println("Applet has read a line: " + line);
         }

          out.println(line);
          if (line.equals("EndOfRecord")) break;
       }
    }
    catch (IOException e) {
      System.err.println(e);
    }
    // always be sure to close the socket
    finally {
      try {
        if (s != null) s.close();
      }
      catch (IOException e2) { }
    }
  }
}
```

c. Create the Java program that implements the “ClientCommExit” class used in the applet: “ClientCommExit.java”. This program deals with the special applet command of “Server Exit”.

```
import java.io.*;
import java.net.*;

public class ClientCommExit {
  public static final int DEFAULT_PORT = 6789;
  private String host = "";
  private int port = 0;
  private OutputStream os = null;
  boolean DEBUG = true;

  public ClientCommExit (String h, int p, OutputStream o) {
    host = h;
    port = ((p == 0) ? DEFAULT_PORT : p);
    os = o;
    Socket s = null;
    PrintWriter out = new PrintWriter (os, true);
    if (DEBUG) {
      System.out.println("Applet about to create a socket on "
                        + host + " at port " + port);
    }

    try {
      // create a socket to communicate to the specified host and port
      s = new Socket(host, port);
      // create streams for reading and writing
      BufferedReader sin = new BufferedReader(new
InputStreamReader(s.getInputStream()));
      PrintStream sout = new PrintStream(s.getOutputStream(), true);
      if (DEBUG) {
        System.out.println("Applet has created sin and sout ");
      }

      // tell the user that we've connected
      out.println("Connected to " + s.getInetAddress() +
        ":" + s.getPort());
      if (DEBUG) {
        System.out.println("Applet has connected to "+ s.getInetAddress() +
           ":" + s.getPort());
      }

      String line;
      // read the first response (a line) from the server
      line = sin.readLine();
      if (DEBUG) {
        System.out.println("Applet has read a line: " + line);
      }

      // write the line to the user
      out.println(line);
      out.flush();
      // send the command choice to the server
      sout.println("Server Exit");
      if (DEBUG) {
        System.out.println("Applet has sent sout the command: Server Exit");
      }
    }
    catch (IOException e) {
      System.err.println(e);
    }
    // always be sure to close the socket
    finally {
      try {
        if (s != null) s.close();
      }
      catch (IOException e2) { }
    }
  }
}
```

d. Create the Java program that implements the “ClientCommSQL” class used in the applet: “ClientCommSQL.java”. This program deals with the special applet commands for SQL statements.

```
import java.io.*;
import java.net.*;

public class ClientCommSQL {
  public static final int DEFAULT_PORT = 6789;
  private String host = "";
  private int port = 0;
  private OutputStream os = null;
  boolean DEBUG = true;

  public ClientCommSQL (String h, int p, int choice, String cmd, OutputStream
o) {
    host = h;
    port = ((p == 0) ? DEFAULT_PORT : p);
    os = o;
    Socket s = null;
    PrintWriter out = new PrintWriter (os, true);
    if (DEBUG) {
      System.out.println("Applet about to create a socket on "
          + host + " at port " + port);
    }

    try {
      // create a socket to communicate to the specified host and port
      s = new Socket(host, port);
      // create streams for reading and writing
      BufferedReader sin = new BufferedReader(new
InputStreamReader(s.getInputStream()));
      PrintStream sout = new PrintStream(s.getOutputStream(), true);
      if (DEBUG) {
        System.out.println("Applet has created sin and sout ");
      }

      // tell the user that we've connected
      out.println("Connected to " + s.getInetAddress() +
        ":" + s.getPort());

      if (DEBUG) {
        System.out.println("Applet has connected to "+ s.getInetAddress() +
            ":" + s.getPort());
      }

      String line;
      // read the first response (a line) from the server
      line = sin.readLine();
      if (DEBUG) {
        System.out.println("Applet has read a line: " + line);
      }

      // write the line to the user
      out.println(line);
      out.flush();
      // send the command choice to the server
      if (choice ==4) {
        sout.println(choice);
        sout.println(cmd);
      } else {
        sout.println("Wrong command");
      }
      if (DEBUG) {
        System.out.println("Applet has sent sout the choice/SQL: "+ choice+
"/"+cmd);
      }

      // read a line from the server
      line = sin.readLine();
      if (DEBUG) {
        System.out.println("Applet has read a line: " + line);
      }

      out.println(line);
      // check if connection is closed, i.e., EOF
      if (line == null) {
        out.println("Connection closed by server.");
      }
       while (true) {
        line=sin.readLine();
        if (DEBUG) {
          System.out.println("Applet has read a line: " + line);
        }

         out.println(line);
         if (line.equals("EndOfRecord")) break;
       }
    }
    catch (IOException e) {
      System.err.println(e);
    }
    // always be sure to close the socket
    finally {
      try {
        if (s != null) s.close();
      }
      catch (IOException e2) { }
    }
  }
}
```

12.3. Prepare the main server program

Create the main server program, “SDB.java”. This program accepts applet connections and user commands and then dispatches the commands to individual processing programs accordingly..

```
import java.net.*;
import java.io.*;

public class SDB {

  public static void main (String args[]) throws IOException {
    Socket client;
    int port = 0;
    int end = 0;
    BufferedReader in;
    PrintStream out;

    if (args.length != 1)
      port = 6789;
    else
      port = Integer.parseInt(args[0]);

    try {
      while (end == 0) {
        client = accept (port);
        in = new BufferedReader(new
InputStreamReader(client.getInputStream()));
        out = new PrintStream(client.getOutputStream());
        out.println ("You are now connected to the Simple Database Server.");
         // read a line
         String line = in.readLine();
        // and send back ACK
         // out.println("OK");
         System.out.println("Received: " + line);
        if (line.equals("1")) DispCus.DispCus(out);
        else if (line.equals("2")) DispPro.DispPro(out);
        else if (line.equals("3")) DispTra.DispTra(out);
        else if (line.equals("4")) {
           out.println("OK");
           line = in.readLine();
          System.out.println ("Received: " + line);
          ExeSQL.ExeSQL(out, line);
        }
         if (line.equals("Server Exit")) {
          end = 1;
        }
        client.close();
      }
    } finally {
      System.out.println ("Closing");
    }
  }
  static Socket accept (int port) throws IOException {
    System.out.println ("Starting on port " + port);
    ServerSocket server = new ServerSocket (port);

    System.out.println ("Waiting");
    Socket client = server.accept ();
    System.out.println ("Accepted from " + client.getInetAddress ());

    server.close ();
    return client;
  }
}
```

12.4. Prepare the database access programs

a. Create the Java program, “DispCus.java” to display the customer table.

```
import java.net.*;
import java.io.*;
import java.sql.*;
public class DispCus  {
      public static void DispCus(PrintStream out) {
            String url = "jdbc:odbc:dbtest";
            Connection con;
            String query = "select * from Customer ";
            Statement stmt;
            try {
                  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            } catch(java.lang.ClassNotFoundException e) {
                  System.err.print("ClassNotFoundException: ");
                  System.err.println(e.getMessage());
            }
            try {
                  con = DriverManager.getConnection(url, "admin", "admin");
                  stmt = con.createStatement();

                  ResultSet rs = stmt.executeQuery(query);
                  ResultSetMetaData rsmd = rs.getMetaData();
                  int numberOfColumns = rsmd.getColumnCount();
                  int rowCount = 1;
                  while (rs.next()) {
                        out.println("Row " + rowCount + ":  ");
                        for (int i = 1; i <= numberOfColumns; i++) {
                              out.print("   Column " + i + ":  ");
                              out.println(rs.getString(i));
                        }
                        out.println("");
                        rowCount++;
                  }
                  out.println("EndOfRecord");
                  stmt.close();
                  con.close();

            } catch(SQLException ex) {
                  System.err.print("SQLException: ");
                  System.err.println(ex.getMessage());
            }
      }
}
```

b. Create the Java program, “DispPro.java” to display the product table.

```
import java.net.*;
import java.io.*;
import java.sql.*;
public class DispPro  {
       public static void DispPro(PrintStream out) {
              String url = "jdbc:odbc:dbtest";
              Connection con;
              String query = "select * from Product ";
              Statement stmt;
              try {
                     Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
              } catch(java.lang.ClassNotFoundException e) {
                     System.err.print("ClassNotFoundException: ");
                     System.err.println(e.getMessage());
              }
              try {
                     con = DriverManager.getConnection(url, "admin", "admin");
                     stmt = con.createStatement();

                     ResultSet rs = stmt.executeQuery(query);
                     ResultSetMetaData rsmd = rs.getMetaData();
                     int numberOfColumns = rsmd.getColumnCount();
                     int rowCount = 1;
                     while (rs.next()) {
                            out.println("Row " + rowCount + ":  ");
                            for (int i = 1; i <= numberOfColumns; i++) {
                                   out.print("   Column " + i + ":  ");
                                   out.println(rs.getString(i));
                            }
                            out.println("");
                            rowCount++;
                     }
                     out.println("EndOfRecord");
                     stmt.close();
                     con.close();

              } catch(SQLException ex) {
                     System.err.print("SQLException: ");
                     System.err.println(ex.getMessage());
              }
       }
}
```

c. Create the Java program, “DispTra.java” to display the transaction table.

```
import java.net.*;
import java.io.*;
import java.sql.*;
public class DispTra  {
       public static void DispTra(PrintStream out) {
              String url = "jdbc:odbc:dbtest";
              Connection con;
              String query = "select * from Transaction ";
              Statement stmt;
              try {
                     Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
              } catch(java.lang.ClassNotFoundException e) {
                     System.err.print("ClassNotFoundException: ");
                     System.err.println(e.getMessage());
              }
              try {
```

```
                        con = DriverManager.getConnection(url, "admin", "admin");
                        stmt = con.createStatement();

                        ResultSet rs = stmt.executeQuery(query);
                        ResultSetMetaData rsmd = rs.getMetaData();
                        int numberOfColumns = rsmd.getColumnCount();
                        int rowCount = 1;
                        while (rs.next()) {
                                out.println("Row " + rowCount + ":  ");
                                for (int i = 1; i <= numberOfColumns; i++) {
                                        out.print("   Column " + i + ":  ");
                                        out.println(rs.getString(i));
                                }
                                out.println("");
                                rowCount++;
                        }
                        out.println("EndOfRecord");
                        stmt.close();
                        con.close();

                } catch(SQLException ex) {
                        System.err.print("SQLException: ");
                        System.err.println(ex.getMessage());
                }
        }
}
```

d. Create the Java program, “ExeSQL.java” to execute an SQL statement.

```
import java.net.*;
import java.io.*;
import java.sql.*;
public class ExeSQL  {
        public static void ExeSQL(PrintStream out, String sqlstr) {
                String url = "jdbc:odbc:dbtest";
                Connection con;
                Statement stmt;
                try {
                        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                } catch(java.lang.ClassNotFoundException e) {
                        System.err.print("ClassNotFoundException: ");
                        System.err.println(e.getMessage());
                }
                try {
                        con = DriverManager.getConnection(url, "admin", "admin");
                        stmt = con.createStatement();

                        ResultSet rs = stmt.executeQuery(sqlstr);
                        ResultSetMetaData rsmd = rs.getMetaData();
                        int numberOfColumns = rsmd.getColumnCount();
                        int rowCount = 1;
                        while (rs.next()) {
                                out.println("Row " + rowCount + ":  ");
                                for (int i = 1; i <= numberOfColumns; i++) {
                                        out.print("   Column " + i + ":  ");
                                        out.println(rs.getString(i));
                                }
                                out.println("");
                                rowCount++;
                        }
                        out.println("EndOfRecord");
                        stmt.close();
```

```
                    con.close();

                } catch(SQLException ex) {
                    System.err.print("SQLException: ");
                    System.err.println(ex.getMessage());
                    out.println("No result.");
                    out.println("EndOfRecord");

                }
        }
}
```

12.5. Compile and test the programs

a. Compile all the Java programs.
b. Execute the server program SDB.class first.
c. Execute the applet via the applet.html using a Web browser.
d. Note the server's host IP address is hard-coded into the ClientApplet.java program. It can be changed to any host address that the server is running. Of course, the applet program has to be re-compiled. This address can be easily entered as a parameter of the program.