Chapter 2: Objects and Primitive Data
Lab Exercises

Lab Exercises

Two Meanings of Plus

In Java, the symbol + can be used to add numbers *or* to concatenate strings. This exercise illustrates both uses.

When using a string literal (a sequence of characters enclosed in double quotation marks) in Java the complete string must fit on one line. The following is NOT legal (it would result in a compile-time error).

```
System.out.println ("It is NOT okay to go to the next line
                     in a LONG string!!!");
```

The solution is to break the long string up into two shorter strings that are joined using the *concatenation* operator (which is the + symbol). This is discussed in Section 2.2 in the text. So the following would be legal

```
System.out.println ("It is OKAY to break a long string into " +
                    "parts and join them with a + symbol.");
```

So, when working with strings the + symbol means to concatenate the strings (join them). BUT, when working with numbers the + means what it has always meant—add!

1. **Observing the Behavior of +** To see the behavior of + in different settings do the following:
   a. Study the program below, which is in file `PlusTest.java`.

```
// ****************************************************************
//   PlusTest.java
//
//   Demonstrate the different behaviors of the + operator
// ****************************************************************

public class PlusTest
{
    // -----------------------------------------------
    // main prints some expressions using the + operator
    // -----------------------------------------------
    public static void main (String[] args)
    {
      System.out.println ("This is a long string that is the " +
                          "concatenation of two shorter strings.");

      System.out.println ("The first computer was invented about" + 55 +
                          "years ago.");

      System.out.println ("8 plus 5 is " + 8 + 5);

      System.out.println ("8 plus 5 is " + (8 + 5));

      System.out.println (8 + 5 + " equals 8 plus 5.");
    }
}
```

   b. Open `PlusTest.java` in your `\Chapter 2` folder.

c. Compile and run the program. For each of the last three output statements (the ones dealing with 8 plus 5) write down what was printed. Next to each of those three output statements, select one of the reasons below why it printed what it did and write the corresponding letter. the following rules are used for +. Write out complete explanations.

   A. If both operands are numbers + is treated as ordinary addition. (NOTE: in the expression a + b the a and b are called the operands.)
   B. If at least one operand is a string the other operand is converted to a string and + is the concatenation operator.
   C. If an expression contains more than one operation expressions inside parentheses are evaluated first. If there are no parentheses the expression is evaluated left to right.

```
System.out.println ("8 plus 5 is " + 8 + 5);

System.out.println ("8 plus 5 is " + (8 + 5));

System.out.println (8 + 5 + " equals 8 plus 5.");
```

d. The statement about when the computer was invented is too scrunched up. How should that be fixed?

A Table of Student Grades

Write a Java program called `StudentGrades.java` that prints a table with a list of at least 5 students together with their grades earned (lab points, bonus points, and the total) in the format below.

```
//////////////////\\\\\\\\\\\\\\\\\\
==          Student Points          ==
\\\\\\\\\\\\\\\\\\//////////////////

Name            Lab     Bonus   Total
----            ---     -----   -----
Joe             43      7       50
William         50      8       58
Mary Sue        39      10      49
```

The requirements for the program are as follows:

1. Print the border on the top as illustrated (using the slash and backslash characters).
2. Use *tab character* (an escape sequence `\t`) to get your columns aligned and you must use the + operator both for addition and string concatenation. (i.e., use addition to calculate the `Total` column using `Lab` + `Bonus` columns.)
3. Make up your own student names and points—the ones shown are just for illustration purposes. You need 5 names.

Area and Circumference of a Circle

Class `Circle.java` describes a circle with a given radius. The radius has the type `double`, which is a primitive data type used for representing real numbers. The `CircleTest.java` class is the driver program that prompts the user to enter a number for the radius, creates a `Circle` object of that radius and displays its area by callig the `Circle's getArea()` method.

Open the file, `Circle.java`, in your `\Chapter 2` folder and modify it as follows:

1. Copy and paste the `getArea()` method. Name the new method `getCircumference()`.
2. Modify the code in the `getCircumference()` method to calculate and return the circumference of the circle object.
3. Compile the `Circle.java` program and fix any syntax errors.

Open the file, `CircleTest.java`, in your `\Chapter 2` folder and modify it as follows:

1. Look at the statement which creates a `double` variable named `area`. This statement creates the variable and assigns it the value returned when calling (invoking) the `circle1` objects `getArea()` method. Note the name of this `Circle` object is `circle1`.
2. Add a statement to create a `double` variable named `circumference`. Call (invoke) the `circle1` object's `getCircumference()` method.
3. Add a statement to print out `circle1's` circumference.
4. Copy and paste the correct statements to prompt the user for another radius, create a second `Circle` object named `circle2` and print `circle2's` area and circumference.
5. Modify the `println` statements to indicate which `Circle` object is being printed, either `circle1` or `circle2`.

Painting a Room

File `Paint.java` contains the partial program below, which when complete will calculate the amount of paint needed to paint the walls of a room of the given length and width. It assumes that the paint covers 350 square feet per gallon.

```
//*************************************************************
//File: Paint.java
//
//Purpose: Determine how much paint is needed to paint the walls
//of a room given its length, width, and height
//*************************************************************
import java.util.Scanner;

public class Paint
{
    public static void main(String[] args)
    {
        final int COVERAGE = 350;  //paint covers 350 sq ft/gal
        //declare integers length, width, and height;
        //declare double totalSqFt;
        //declare double paintNeeded;
        //declare and initialize Scanner object

        //Prompt for and read in the length of the room

        //Prompt for and read in the width of the room

        //Prompt for and read in the height of the room

        //Compute the total square feet to be painted--think
        //about the dimensions of each wall

        //Compute the amount of paint needed

        //Print the length, width, and height of the room and the
        //number of gallons of paint needed.
    }
}
```

Open the `Paint.java` file in the `\Chapter 2` directory and do the following:

1. Fill in the missing statements (the comments tell you where to fill in) so that the program does what it is supposed to. Assume that your are only painting the walls, not the ceiling or floor. Compile and run the program and correct any errors.

   Test your program with a length 12 feet, width 16 feet and height 10 feet. *(1.6 gallons needed is answer)*

2. Suppose the room has doors and windows that don't need painting. Ask the user to enter the number of doors and number of windows in the room, and adjust the total square feet to be painted accordingly. Assume that each door is 20 square feet and each window is 15 square feet.

   Test your program with the dimensions above plus 2 windows and 1 door. *(1.457 gallons needed is answer)*

Computing Distance

The file `Distance.java` contains an incomplete program to compute the distance between two points. Recall that the distance between the two points (x1, y1) and (x2, y2) is computed by taking the square root of the quantity $(x1 - x2)^2 + (y1 - y2)^2$. The program already has code to get the two points as input. You need to add an assignment statement to compute the distance and then a print statement to print it out (appropriately labeled). Test your program using the following data: The distance between the points (3, 17) and (8, 10) is 8.6023... (lots more digits printed); the distance between (–33, 49) and (–9, –15) is 68.352.…

```
// ************************************************************
//   Distance.java
//
//   Computes the distance between two points
// ************************************************************

import java.util.Scanner;

public class Distance
{
    public static void main (String[] args)
    {
      double x1, y1, x2, y2; // coordinates of two points
      double distance;       // distance between the points

      Scanner scan = new Scanner(System.in);

      // Read in the two points
      System.out.print ("Enter the coordinates of the first point " +
                "(put a space between them): ");
      x1 = scan.nextDouble();
      y1 = scan.nextDouble();

      System.out.print ("Enter the coordinates of the second point: ");
      x2 = scan.nextDouble();
      y2 = scan.nextDouble();

      // Compute the distance

      // Print out the answer

    }
}
```

Rolling Dice

Write a complete Java program, `Dice.java`, that simulates the rolling of a pair of dice. For each die in the pair, the program should generate a random number between 1 and 6 (inclusive). It should print out the result of the roll for each die and the total roll (the sum of the two dice), all appropriately labeled. You must use the `random( )` method of the `Math` class. Look at your Chapter 2 notes for an example on how to use `Math.random( ).`

Introduction to HTML

HTML is the HyperText Markup Language. It is used to describe how text, images, and multimedia are displayed by Web browsers. In this lab you will learn a little about HTML so that you can create a web page containing headings, interesting fonts, lists, and links as well as applets.

HTML uses *tags* to describe the layout of a document; the browser then uses these tags to figure out how to display the document. Tags are enclosed in angle brackets. For example, <title> is a tag that indicates that this section contains the title of the document. Many tags, including <title>, have corresponding end tags that indicate where the section ends. The end tags look just like the start tags except that they start with the character /, e.g., </title>. So the following text indicates that the title of the document is `Introduction to HTML`:

```
<title>Introduction to HTML</title>
```

There are a few tags that almost every document will have: <html>, <head>, <title>, and <body>. Here is an example of a simple HTML document:

```
<HTML>
  <HEAD>
    <TITLE>Introduction to HTML</TITLE>
  </HEAD>

  <BODY>
  In this lab you will learn about HTML, which is lots of fun
  to use.  In particular, you will learn how to use fonts,
  paragraphs, lists, links and applets in a web page.  Now you
  can make your own web page for your friends to visit!
  </BODY>
</HTML>
```

Create a folder called `\HTML` in your `\APCS` folder. Open up Notepad and cut and paste the HTML document above. Save as `test.html` and close the file.

To see what this looks like, open the file in the web browser by double clicking on the filename. Change the size of the browser window (click and drag any corner) and see how the text is reformatted as the window changes. Note that the title appears on the window, not as part of the document.

The HEAD of a document (everything between <HEAD> and </HEAD>) contains the introduction to the document. The title goes in the head, but for now we won't use the head for anything else. The BODY of a document (everything between <BODY> and </BODY>) contains everything that will be displayed as part of the document. Both the HEAD and the BODY are enclosed by the HTML tags, which begin and end the document.

This document contains only plain text, but an HTML document can have much more structure: headings, paragraphs, lists, bold and italic text, images, links, tables, and so on. Here is a document containing a heading, two paragraphs, and some fancy fonts:

```
<HTML>
  <HEAD>
    <TITLE>Introduction to HTML</TITLE>
  </HEAD>

  <BODY BGCOLOR="lightgreen">
  <H1 align="center">Introduction to HTML</H1>

  <P>In this lab you will learn about <I>HTML</I>, which
  is lots of fun
  to use.  In particular, you will learn how to use fonts,
  paragraphs, lists, links, and colors in a web page.  Now you
  can make your <B>own</B> web page for your friends to visit!</P>

  <P>Later in this lab you will do some fancier stuff with
  applets and graphics and include an applet on your web page.
  Can't you just feel the job offers start rolling in?</P>
  <U>Yippee!</U>
  </BODY>
</HTML>
```

Open Notepad again and copy the above code to `test2.html`. Close the file. Run the HTML document to see what this looks like in the browser.

In this document the <H1> tag creates a level 1 heading. This is the biggest heading; it might be used at the beginning of the document or the start of a new chapter. Level 2 through level 6 headings are also available with the <H2> through <H6> tags.

The <P> tag creates a new paragraph. Most browsers leave a blank line between paragraphs. The <B> tag creates bold text, the <I> tag creates italic text, and the <U> tag creates underlined text. Note that each of these tags is closed with the corresponding end tag. The BGCOLOR attribute on the BODY tag sets the background color.

Note that line breaks and blank lines in the HTML document do not matter—the browser will format paragraphs to fit the window. If it weren't for the <P> tag, the blank line between the paragraphs in this document would not show up in the displayed document.

Exercise #1

Open a new file in Notepad called `MyPage.html` in your `\HTML` folder. Write a simple web page about things that interest you. Your page should contain at least the following:

- A title (using the <TITLE> tag)
- Two different levels of headings
- Two paragraphs
- Some bold, italic, or underlined text

Your name should appear somewhere in the document.

When you are done, view your document from the browser.

More HTML

Lists We often want to add a list to a document. HTML provides two kinds of lists, *ordered* (e.g., 1, 2, 3) and *unordered* (e.g., bulleted). A list is introduced with the <OL> or <UL> tag, depending on whether it is ordered or unordered. Each list item is introduced with a <LI> tag and ended with the </LI> tag. The entire list is then ended with </OL> or </UL>, as appropriate. For example, the code below creates the list shown; replacing the <OL> and </OL> tags with <UL> and </UL> would produce the same list with bullets instead of numbers.

```
Things I like:
<OL>
<LI>chocolate
<LI>rabbits
<LI>chocolate rabbits
</OL>
```

Things I like:

1. chocolate
2. rabbits
3. chocolate rabbits

Exercise #2: Add a list, either ordered or unordered, of at least three elements to your document.

Links Links connect one document to another. Links are created in HTML with the <A> (anchor) tag. When creating a link you have to specify two things:

- The URL of the document to go to when the link is clicked. This is given as the HREF attribute of the A element.
- How the link should be displayed (that is, what text or image to click on to go to the linked document). This appears between the <A> and </A> tags.

For example, the code below creates the link shown, which goes to a page about the history of computing:

```
Learn more about <A HREF="http://ei.cs.vt.edu/~history">the history of
computing.</A>
```

Learn more about the history of computing.

Exercise #3: Add at least one link that ties in to the material on your page.

Drawing Shapes

The following is a simple applet that draws a blue rectangle on a yellow background.

```
// ********************************************************************
//   Shapes.java
//
//   The program will draw two filled rectangles and a
//   filled oval.
// ********************************************************************

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

public class Shapes extends JApplet
{
    public void init()
    {
      // Set the background color
      getContentPane().setBackground (Color.yellow);
    }

    public void paint (Graphics page)
    {
      // Declare size constants
      final int MAX_SIZE = 300;
      final int PAGE_WIDTH = 600;
      final int PAGE_HEIGHT = 400;

      // Declare variables
      int x, y;    // x and y coordinates of upper left-corner of each shape
      int width, height; // width and height of each shape

      super.paint(page);

      // Set the color for the next shape to be drawn
      page.setColor (Color.blue);

      // Assign the corner point and width and height
      x = 200;
      y = 150;
      width = 100;
      height = 70;

      // Draw the rectangle
      page.fillRect(x, y, width, height);
    }
}
```

Study the code, noting the following:

- The program imports javax.swing.JApplet because this is an applet, and it imports `java.awt.*` because it uses graphics.
- There is no `main method`—instead there is a `paint` method. The paint method is automatically invoked when an applet is displayed, just as the main method is automatically invoked when an application is executed.
- Most of the methods that draw shapes (see the list in Figure 2.12) require parameters that specify the upper left-hand corner of the shape (using the coordinate system described in Section 2.9) and the width and height of the shape. You can see this in the calls to `fillRect`, which draws a rectangle filled with the current foreground color.

- This applet will be drawn assuming the window for drawing (the Graphics object - named *page* here) is 600 pixels wide and 400 pixels high. These numbers are defined in constants at the beginning of the program. (They currently have no use but you will use them later). The width and height of the applet are actually specified in the HTML file that instructs the Web browser to run the applet (remember applets are executed by Web browsers and Web browsers get their instructions from HTML documents—note that the code executed by the browser is the *bytecode* for the program, the `Shapes.class` file). The code in the HTML document is as follows:

```
<html>
<applet code="Shapes.class" width=600 height=400>
</applet>
</html>
```

Open `Shapes.java` in BlueJ in the `\Chapter 2` folder. Now do the following:

1. Compile and run `Shapes.java`. The program will run through appletviewer. You should see a blue rectangle on a yellow background.

3. Now change the x and y variables both to 0. Recompile and run the program.

   What happened to the rectangle?

5. Now change the width to 200 and the height to 300. Save, recompile and run to see how this affects the rectangle.

6. Change x to 400, y to 40, width to 50 and height to 200. Test the program to see the effect.

7. Modify the program so that it draws four rectangles in all, as follows:
   - One rectangle should be entirely contained in another rectangle.
   - One rectangle should overlap one of the first two but not be entirely inside of it.
   - The fourth rectangle should not overlap any of the others.

8. One last touch to the program ... Change the colors for at least three of the shapes so the background and each of the three shapes are different colors (a list of colors is in Figure 2.10 of the text). Also change two of the `fillRect` methods to `fillOval` so the final program draws two rectangles and two ovals. Be sure that the overlap rules are still met.

Colors in Java

The basic scheme for representing a picture in a computer is to break the picture down into small elements called *pixels* and then represent the color of each pixel by a numeric code (this idea is discussed in section 1.6 of the text). In most computer languages, including Java, the color is specified by three numbers—one representing the amount of red in the color, another the amount of green, and the third the amount of blue. These numbers are referred to as the *RGB value* of the color. In Java, each of the three primary colors is represented by an 8-bit code. Hence, the possible base 10 values for each have a range of 0-255. Zero means none of that color while 255 means the maximum amount of the color. Pure red is represented by 255 for red, 0 for green, and 0 for blue, while magenta is a mix of red and blue (255 for red, 0 for green, and 255 for blue). In Java you can create your own colors. So far in the graphics programs we have written we have used the pre-defined colors, such as `Color.red`, from the `Color` class. However, we may also create our own `Color` object and use it in a graphics program. One way to create a `Color` object is to declare a variable of type `Color` and instantiate it using the constructor that requires three integer parameters—the first representing the amount of red, the second the amount of green, and the third the amount of blue in the color. For example, the following declares the `Color` object `myColor` and instantiates it to a color with code 255 for red, 0 for green, and 255 for blue.

```
Color myColor = new Color(255, 0, 255);
```

The statement `page.setColor(myColor)` then will set the foreground color for the page to be the color defined by the `myColor` object. The file `Colors.java` contains an applet that defines `myColor` to be a `Color` object with color code (200, 100, 255) - a shade of purple. From BlueJ, open the file `Colors.html` in the `\Chapter 2` folder, compile and run it using the appletviewer. Now make the following modifications:

1. Change the constructor so the color code is (0,0,0) --- absence of color. What color should this be? Run the program to check.

2. Try a few other combinations of color codes to see what you get. The Java API contains the codes for the pre-defined colors of the Color class.

3. Notice in the Color class in the Java API there is a constructor that takes a single integer as an argument. The first 8 bits of this integer are ignored while the last 24 bits define the color—8 bits for red, 8 for green, and the last 8 bits for blue. Hence, the bit pattern

   ```
   00000000000000001111111100000000
   ```

   should represent pure green. Its base 10 value is 65280. Change the declaration of the *myColor* object to

   ```
   Color myColor = new Color (65280);
   ```

   Compile and run the program. Do you see green?

```
// ************************************************************
//   Colors.java
//
//   Draw rectangles to illustrate colors and their codes in Java
// ************************************************************

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

public class Colors extends JApplet
{
    public void paint (Graphics page)
    {
      // Declare size constants
      final int PAGE_WIDTH = 600;
      final int PAGE_HEIGHT = 400;

      // Declare variables
      int x, y;    // x and y coordinates of upper left-corner of each shape
      int width, height; // width and height of each shape

      Color myColor = new Color (200, 100, 255);

      // Set the background color and paint the screen with a white rectangle
      setBackground (Color.white);
      page.setColor(Color.white);
      page.fillRect(0, 0, PAGE_WIDTH, PAGE_HEIGHT);

      // Set the color for the rectangle
      page.setColor (myColor);

      // Assign the corner point and width and height then draw
      x = 200;
      y = 125;
      width = 200;
      height = 150;

      page.fillRect(x, y, width, height);

    }
}
```

Colors.html

```
<html>
<applet code="Colors.class" width=600 height=400>
</applet>
</html>
```

Drawing a Face

Write an applet that draws a smiling face. Give the face eyes with pupils, ears, a nose, and a mouth. Use at least three different colors, and fill in some of the features. Name this file `Face.java`. View your applet using the applet viewer.

HINT: Look at the `Snowman.java` file for help on creating your face.

Create the following html file and put in the `\Chapter 2` folder.

Face.html

```
<html>
<applet code="Face.class" width=600 height=400>
</applet>
</html>
```

When your `Face` program is working correctly, add html code to your webpage you created earlier (`test2.html` in the `\Chapter 2` folder)to display the face. Copy the `Face.class` file to your `\HTML` folder. Open up your html file and put the following code right above the last line:

```
<applet code="Face.class" width=600 height=400>
</applet>
```

Flipping a Coin

`CoinTest` is part of a program that shows a picture of a coin in the middle of a window and "flips" the coin every two seconds. `Coin.java` is the class representing the coin.

1. In `Coin.java`, add an instance variable of type `int` named `side`. This will be set to 1 when heads is showing and -1 when tails is showing.
2. Compile and run the program.
3. Look at the code in `CoinTest.java`. Try and find the line to change the timing of the coin flip. Make if flip faster and then slower.
4. In the Chapter 2 folder, you will find two .gif files containing pictures of the heads and tails. Look on the internet and download your own pictures to flip through.
5. If you finish early, try downloading a third .gif file and flip through 3 pictures. You may decide to use the values 1, 2 and 3 to flip through the images, rather than 1 and -1.