Primitive Numeric Types, Arithmetic

[For more information, see your class notes and your textbook, pages 184–185 and 336–342.]

Console Output

To print information to the console, use one of the two following Java statements:

- `System.out.println(...)` prints a line of text followed by a newline character.
- `System.out.print(...)` prints a line of text and leaves the cursor at the end of the line (i.e., does not output a newline character).

The text is either a string (enclosed in double quotes) or a single value of any type (it will be automatically converted into a string before printing) or a mix of strings and values separated by “+”.

(The preceding sentence is inaccurate in a number of ways, but is okay as a first approximation!)

Examples:

```
System.out.println("Enter your first name: ");
System.out.println(x);
System.out.print("How much is " + x + " + " + y + "? ");
System.out.print("Good morning, " + name + ", how are you? ");
```

In the third example, the output might be something like this (depending on the values of the variables `x` and `y`):

```
How much is 10 + 7?
```

Console Input

[For more information, see textbook, pages 565–569] To read in values from the terminal window, you must:

- `import java.util.Scanner;`
- Declare and define a variable of class `Scanner` similar to the following:
Scanner terminal = new Scanner(System.in);

- Use the methods of the class Scanner to obtain input from the user. Methods are `nextInt()`, `nextDouble()`, `nextLine()`, and a few others (see online Java documentation for more).

NOTE: there are some tricky things involved in using the `nextLine()` method, because it does not automatically begin reading on a new line — see me if you are trying to use it and having no luck. Each of these methods returns a value of the appropriate type (in the case of `nextLine` the type is `String`).

Examples:

```java
import java.util.Scanner;
...
Scanner term = new Scanner(System.in);
...
int n = term.nextInt(); // reads an integer
double x = term.nextDouble(); // reads a double
String line = term.nextLine(); //WARNING! may not do what you expect!
```

Download Files

On the class Web page, download the files `Lab4.java`, `MyRectangle.java`, and `Circle.java`. IMPORTANT! If you downloaded these files in lab on Tuesday, 27 February, you will need to get new copies!

Use `javac Lab4.java` and `java Lab4` to make sure the files compile and execute.

Modify Files

The program `Lab4` consists of two phases.

- In phase 1, the user enters the coordinates for the centers of two circles and the radii of the two circles. The application draws these circles in a specially modified rectangle that has, in addition to the two circles, a conversation bubble. When the user clicks and drags a circle, it moves around the screen. When the user clicks anywhere outside the two circles, the conversation bubble says either “Touching” or “Not touching”, indicating whether or not the two circles overlap.

- In phase 2, the user enters data as before, but now when the user clicks outside the circles the conversation bubble shows a numeric value indicating how far apart the edges of the two circles are. If the circles overlap, this distance will be zero (even if one circle lies inside the other). If the circles do not overlap, then it should be the shortest distance from the edge of one circle to the edge of the other circle. To do this, you will need to display a “double” value inside your conversation bubble. This is easy. For instance, if the variable `separation` contains a `double` value, we can display it in a bubble like this:
In class on Wednesday, 28 February, we saw how to create a Circle class that permits dragging circles around on the screen.

In class on Friday we will discuss if statements (you will need these in order to display the correct information in the conversation bubble).

You will need to add the appropriate “mouseDragged” or “mouseClicked” methods to the appropriate classes, as well as any other methods, variables, etc., needed to achieve the goal.

Optional

If you found the above tasks easy, then please feel free to try something more ambitious. Be sure to demonstrate this for me in lab next week, and be sure to explain in your program documentation exactly what your modification does. Here are a few suggestions (I haven’t tried these!):

- When the user clicks in the rectangle, have the circle that is nearest the top of the screen change color (or nearest the bottom, or nearest the left, etc.)

- When the user clicks in the rectangle, have the circles move towards each other until their edges touch

- When the user clicks in the rectangle, the conversation bubble displays whether or not one circle is entirely contained in the other. (You need to first use “if” statements to figure out which circle is the smaller one, then use information about the centers and radii to figure out if the small one is contained in the large one).

- When the user clicks in the rectangle, find the area of the overlapping portion of the circles and display it in the conversation bubble (this is really a math problem, not a programming problem! If you know the radii and centers, you should be able to find the area — I can help you with the math). Unfortunately the “wheels” package doesn’t make it easy to show the overlapping area, since one circle obscures the other one.

- Make up an interesting problem involving these two circles and try to solve it using mouse clicks, arithmetic, and if statements!

Hand in any code that you have modified, fully commented. Remove any obsolete comments. Don’t hand in files that you have not made changes to (for instance, you will probably not need to make changes to the Lab4 file since it simply sets up the frame.