Programming Techniques COSC1284/2010

Tutorial 7

Agenda

- Tutorial/Lab
    - Read chapter 9 (9.1 - 9.4) and 10 (10.1 - 10.4) from the textbook.
    - Discuss the concepts with your tutor and fellow classmates
        - Complete chapter 9 - Exercises 1 - 2
        - Complete chapter 10 - Exercises 1
    - Attempt on your own
        - Complete chapter 9 - Exercises 6
        - Complete chapter 10 - Exercises 2 - 3
- Note: Please refer to tutorial 4 for online instructions.

Simple variables are known as primitives

- As we have working through the semester, we have been generally working with primitive types, that is int, double, boolean, with three exceptions, String, Scanner and more recently Array.
- Primitives hold a value that is direct, meaning if you look at the memory address of a primitive, that location will contain that value.
- This is not true for non-primitives.

Complex variables are known as objects

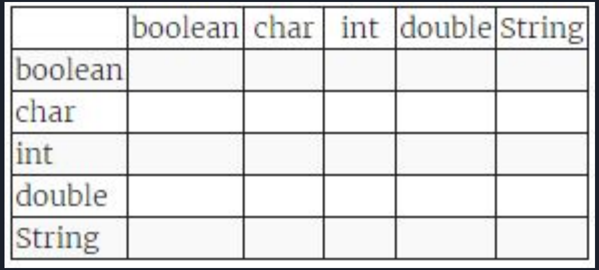
- As we have been working, where we can represent an idea through multiple variables (including different types) as well as including behaviours within that variable.
- For instance, if we wanted to create a variable that describes a person.
- This is basically impossible with a single primitive.
- Instead we need to include multiple primitives and enforce a grouping around these variables.
- For example, we could create three primitives called personId, personAge and personGender or ….
- Create a variable called person (the object) that holds and groups the variables id, age, gender within itself.
- Object variable have an indirect reference to the object data.
- And just to complicate things a bit more, objects can also contain methods.

Exercise 9.1

The point of this exercise is to explore Java types and fill in some of the details that aren't covered in the chapter.

1. Create a new program named Test.java and write a main method that contains expressions that combine various types using the + operator. For example, what happens when you "add" a String and a char? Does it perform character addition or string concatenation? What is the type of the result? (How can you determine the type of the result?)
2. Make a bigger copy of the following table and fill it in. At the intersection of each pair of types, you should indicate whether it is legal to use the + operator with these types, what operation is performed (addition or concatenation), and what the type of the result is.

Exercise 9.1 (cont).



	boolean	char	int	double	String
boolean					
char					
int					
double					
String					

Exercise 9.1 (cont).

3. Think about some of the choices the designers of Java made, based on this table. How many of the entries seem unavoidable, as if there was no other choice? How many seem like arbitrary choices from several equally reasonable possibilities? Which entries seem most problematic?
4. Here's a puzzler: normally, the statement x++ is exactly equivalent to x = x + 1. But if x is a char, it's not exactly the same! In that case, x++ is legal, but x = x + 1 causes an error. Try it out and see what the error message is, then see if you can figure out what is going on.
5. What happens when you add "" (the empty string) to the other types, for example, "" + 5?

Exercise 9.2

The goal of this exercise is to practice encapsulation and generalization using some of the examples in previous chapters.

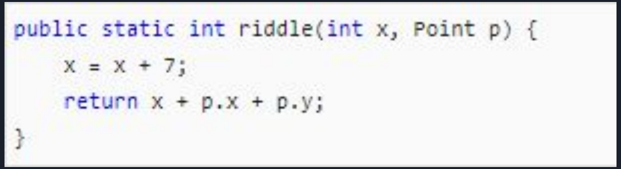
1. Starting with the code in Section 7.5, write a method called powArray that takes a double array, a, and returns a new array that contains the elements of a squared. Generalize it to take a second argument and raise the elements of a to the given power.
2. Starting with the code in Section 7.8, write a method called histogram that takes an int array of scores from 0 to (but not including) 100, and returns a histogram of 100 counters. Generalize it to take the number of counters as an argument.

Exercise 10.1

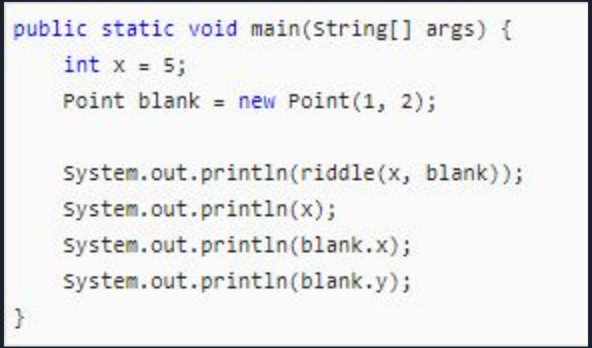
The point of this exercise is to make sure you understand the mechanism for passing objects as parameters.

1. For the following program, draw a stack diagram showing the local variables and parameters of main and riddle just before riddle returns. Use arrows to show which objects each variable references.
2. What is the output of the program?
3. Is the blank object mutable or immutable? How can you tell?

Exercise 10.1



```java
public static int riddle(int x, Point p) {
    x = x + 7;
    return x + p.x + p.y;
}
```



```java
public static void main(String[] args) {
    int x = 5;
    Point blank = new Point(1, 2);

    System.out.println(riddle(x, blank));
    System.out.println(x);
    System.out.println(blank.x);
    System.out.println(blank.y);
}
```