Introduction

Welcome to the Intro to Java: Functional Programming, Lesson 3 problem set! These problem sets are an opportunity for you to practice the concepts you learned in class before moving on to the next lesson. Learning a computer programming language is similar to learning a human language. Nobody can pick it up overnight, there's a lot of vocabulary and syntax to remember. Language learners often speak of the moment when they realized they stopped translating in their head and actually started thinking in their second language. This will happen with Java too! Eventually, you will be able to consider a task that needs coding and immediately imagine what Java code would complete it. To get there, though, requires practice.

That's where the problem sets come in. They aren't mandatory, and they aren't graded. They're just extra learning materials to help you along.

Completing the Problem Sets

There isn't a right or wrong way to work on these. Some problems require you to examine code or do some arithmetic. You can take notes on paper, print this document and use the space provided, or try to do it all in your head—whatever works for you. For the exercises that require programming, we highly recommend that you pick your favorite text editor, open a blank text file, and try writing out the code.

Question 1

In the function signature below, what is the return type?

```
public float squareRoot(int x)
```

A. `public`

B. `float`

C. `squareRoot`

D. `int`

Question 1 Solution

B. `float`

In the signature

```
public float squareRoot(int x)
```

`public` is the access modifier, meaning the function can be used from anywhere. `float` is the return type, indicating that it will return a float value. `squareRoot` is the function name, and `int` is the type of the parameter. Since the square root of an integer might not be an integer, the return type is different from the parameter type.

Question 2

Write the signature of a function called `isPrime()`. The access modifier should be public, the return type should be boolean, and it should take a single integer parameter.

Question 2 Solution

```
public boolean isPrime(int n)
```

In a function signature, the access modifier comes first, then the return type, then the name. The type/name of the parameters is in parentheses after the name of the function.

Question 3

Which of the following function signatures has an error?

A. `public getAccountBalanace(long accountNumber)`

B. `public void displayInTextBox(String string)`

C. `public int roundToNearestInt(double x)`

D. `public double getTemperature()`

Question 3 Solution

A. `public getAccountBalance(long accountNumber)`

This function signature has no return type. The return type *always* comes before the name of the function. If nothing is returned, the keyword `void` is used where the return type belongs. Since this function is called `getAccountBalance`, it should probably return a numerical value such as a `double`, `float`, or `int`.

Question 4

Write a Java function called `absoluteValue()`. The access modifier should be public, it should have a return type of double, and it should take one double parameter as input. If the double parameter is less than 0, it should return that number negated. Otherwise, it should return the parameter unchanged.

Question 4 Solution

Example solution code:

```
public double absoluteValue(double x) {
    if (x < 0) {
        return -x;
    } else {
        return x;
    }
}
```

Question 5

Write a Java function named `calculateTip()`. The access modifier should be public, it should have a return type of double, and it should take as input a double parameter representing the cost of a meal at a restaurant. And finally, it should return a double equal to 15% of the cost parameter.

Question 5 Solution

Example solution code:

```
public double calculateTip(double cost) {
    double tip = cost * 0.15;
    return tip;
}
```

Question 6

Write a Java function called `nametagText()`. The access modifier should be public, the return type should be String, and it should take a String parameter called name. In the body of the function, return the String "Hello, my name is " with the name parameter added to the end. (Hint: use String concatenation.)

Question 6 Solution

Example solution code:

```
public String nametagText(String name) {
    String nametagText = "Hello, my name is " + name;
    return nametagText;
}
```

Question 7

Define two functions. The first should be called `fahrenheitToCelsius()`. It should be a public function with return type double that takes a double argument that represents a temperature in Fahrenheit degrees. It should return the equivalent temperature in Celsius degrees. (To convert from Fahrenheit to Celsius, use the formula $C = (F - 32) \times 5/9$.)

Next, define a function called `printTemperature()`. It should be public, it should have a return type of void, and it should take a double parameter that represents a temperature in Fahrenheit degrees. This function should print "F: " followed by the Fahrenheit parameter, then "C: " followed by the equivalent value in Celsius degrees. Use the first function you defined to calculate the appropriate Celsius value inside the second function.

Bonus challenge: write javadoc comments for both functions.

Example solution code:

```
/**
 * Converts from Fahrenheit to Celsius degrees.
 * @param  fahrenheit  Temperature in degrees Fahrenheit.
 * @return  Equivalent temperature in degrees Celsius.
 */
public double fahrenheitToCelsius(double fahrenheit) {
        return (fahrenheit - 32) * 5 / 9;
}

/**
 * Prints a temperature in both Fahrenheit and Celsius degrees.
 * @param  fahrenheit  Temperature in degrees Fahrenheit.
 */
public void printTemperature(double fahrenheit) {
        System.out.println("F: " + fahrenheit);
        System.out.println("C: " + fahrenheitToCelsius(fahrenheit));
}
```

Question 8

Define a function called `monopolyRoll()`. This function provides a random result based on the dice-rolling rules for the board game Monopoly. In Monopoly, players roll two six-sided dice to determine their move. If they roll the same value on both dice, this is called "rolling doubles," and it means they go again. In our simplified version, the dice-roll function should behave like this:

1. Generate two random integers in the 1 to 6 range.
2. If the numbers are not the same, return the sum.
3. If the numbers are the same, generate two more random integers in the 1 to 6 range, and return the sum of all 4 numbers.

Hint: to make your code neater, you can define a second function that generates a random integer in the 1 to 6 range (or in the 1 to x range based on a parameter) so that you do not need to keep repeating that code.

Question 8 Solution

Example solution code:

```
/**
 * Returns a random integer simulating a dice roll.
 * @param  sides  Number of sides on the virtual die being rolled.
 * @return  random number in the range of 1 to sides.
 */
public int diceRoll(int sides) {
    //This expression generates a random double in the interval
    //[0, sides).  That is, a double greater than or equal to
    //0 and less than sides.
    double randomNumber = Math.random() * sides;
    //Our random number is now in the interval [1, sides + 1)
    randomNumber = randomNumber + 1;
    //Casting the random number to an integer will round it down to an
    //integer in the 1 to sides range.
    return (int) randomNumber;
}

public int monopolyRoll() {
    int roll1 = diceRoll(6);
    int roll2 = diceRoll(6);
    int total = roll1 + roll2;
    if (roll1 == roll2) {
        int roll3 = diceRoll(6);
        int roll4 = diceRoll(6);
        total = total + roll3 + roll4;
    }
    return total;
}
```