C1A, C1F Tutorial Course

Homework Lab Set A-1

C1 – Introduction to Computer Science – Learning Java

Due Date: Friday, July 20, 2018 @ 11:59pm

Instructor: Evan Chen

Full Name (PRINT LEGIBLY): _____

Signature Pledge

I, _____, pledge that I will not plagiarize my work. Upon submitting my homework set, I promise that all of my work is my own and that it is not copied from a classmate of mine and it is not work done by any of my friends or colleagues or other teachers and professors. If I do copy some code or any of the answers online, I promise that all of this work is referenced wherever needed. I know that in failure to comply to this pledge, I will receive a 20-mark reduction.

Work on the course material and problems below. You must complete all of the questions displayed in Part I. There will be 5 questions from Part II to work on and 4 questions from Part III.

When you are finished, please hand in your work to me before end of day July 20th, 2018. Your work will include the following:

- *This assignment booklet with all of the Part I questions answered. (You must hand this to me in person or online, by email or Slack **ONLY**.)*
- *Submit all of the code (in Github) online* **ONLY***, DO NOT PRINT IT OUT AND STAPLE IT TO THIS BOOKLET.*

Late submissions result in a 15% reduction of your total mark on this evaluation.

Instructions

1. Please answer all of the Short Answer knowledge questions in Part I. You may record your answers in pen, in pencil, or typed up. If you need assistance, feel free to ask me through text, email, or in person (right before the class starts or after class).

2. In Part II of this booklet, you are required to work on all 5 questions presented there. Please note that some of the sub-questions may require you to directly answer on this booklet, but almost all of the questions will require you to code. You may find the starter code on the tutorial course webpage that corresponds to the questions you have chosen to do. Please note that all code MUST be submitted online, through Github.

3. In Part III of this booklet, you are required to work on all 4 of the questions. However, there will be **no starter code provided for you on the tutorial course webpage**. You are required to write all of your code from scratch. Please note that all code MUST be submitted online, through Github.

Grading Scheme:

Part I	/15 marks	**Total**	**/165 marks**
Part II	/50 marks		
Part III	/100 marks		

Part I – Short Answer + Knowledge Questions

Answer the following questions. There are twenty questions. Each question is worth one mark. Be sure to read each question carefully before answering. Please answer using pen or pencil. You may also choose to type up your answers (if you wish to submit it online). Please keep your answers short.

1) What is the **main** function? What is its purpose? Write down what the header of the **main** function is.

2) What is the difference between **System.out.println**() and **System.out.print**()?

3) How do you return a value from the function? Can you return nothing?

4) Suppose I had the following code written.

```
System.out.print("                    Hello World");
System.out.println("I'm on another line?                    ");
System.out.println("This is another line.");
System.out.print("How about this?   ");
```

What will it print out?

5) What is the purpose of a variable?

6) Consider the following code inside a function.

```
int i = 10;
int j = 20;
int swap;
swap = i;
i = j;
j = swap;
```

What is the value of variable i if I return it?

7) Consider the following code inside a function.

```
int i = 10;
int j = 20;
int swap;
swap = i;
i = j;
j = swap;
```

What is the value of i + j * swap if I return it?

8) What are some of the different types you can use in Java? (Please list at least three different types we can use in Java.)

9) Consider the following code:

```
public int isInfFunction(int x) {
        if(x == 0) {
                return x;
        }
        return 1 + isInfFunction(x);
}
```

If I were to have ran the following code in the main function:

```
int x = 0;
int value = isInfFunction(x);
System.out.println("The value is " + value);
```

What would the System.out.println() snippet print out? Will this call the function infinitely?

10) Consider the following code:

```
public int isInfFunction(int x) {
        if(x == 0) {
                return x;
        }
        return 1 + isInfFunction(x);
}
```

If I were to have ran the following code in the main function:

```
int x = 10;
int value = isInfFunction(x);
System.out.println("The value is " + value);
```

What would the System.out.println() snippet print out? Will this call the function infinitely?

11) Write an if statement based on this scenario (assume the integer variable "coinSide" has been defined already):

    Imagine that the program has asked the user to flip a coin. Suppose the program has been compiled and ran without any errors. The function will return one of the String values, based on what value coinSide is.

    - If the value of coinSide is 0, then the function will return "It landed on tails."
    - If the value of coinSide is 1, then the function will return "It landed on heads."

12) Translate the following if statement:

```
if(flowers.equals(“red”)) {
        return “Roses are red.”;
} else if(flowers.equals(“blue”)) {
        return “Violets are blue.”;
} else {
        return “The other colours are not a part of the poem.”;
}
```

13) Suppose that you had three functions named A, B, and C. Function A takes in an integer parameter called x and Function B takes in an integer parameter called y. Function C does not take in any parameters. All of their return value types are integers.

    Function A returns a value of x, function B returns a value of y, and function C returns a value of 1, for whatever values x and y are inside their respective functions.

    If the user passed in a value of x = 10 to function A, a value of y = 35 to function B, what would the following code print out? (What value would be printed into the console?)

```
System.out.println(A()*B() + (C() – A() + B()));
```

14) Suppose that you had three functions named A, B, and C. Function A takes in an integer parameter called x and Function B takes in an integer parameter called y. Function C does not take in any parameters. All of their return value types are integers.

    Function A returns a value of x, function B returns a value of y, and function C returns a value of 1, for whatever values x and y are inside their respective functions.

    If the user passed in a value of x = 10 to function A, a value of y = “forty two” to function B, what would the following code print out? (What value would be printed into the console?)

```
System.out.println(A()*B() + (C() – A() + B()));
```

15) Imagine that someone was to have returned the following String from a function:

```
return 5 + 10 + " is not my favourite number.";
```

What will get returned? Why? Will the result change if I were to have swapped everything around?

```
return 5 + " is not my favourite number. But this is: " + 10;
```

Part II *– Fill-in-the-blank Coding Questions*

For the next 5 questions, please download the respective code from the tutorial course website and fill in the blanks of the missing code based on the instructions below and in the file's coded comments. (The comments you will see *indicate where to add the code*!)

1) *Small String Fun*
   First, download the folder package containing this particular code from the tutorial course website. Save it to a folder – this folder will be set as your Eclipse workspace. Open the folder package and copy the inner folder into the same folder you just saved this zipped folder package in. Then, open up Eclipse and set the folder you saved the folder package in as your Eclipse workspace.

   In Eclipse, open the program folder called "Small String Fun" in the *Package Explorer* panel (on your left hand side). After clicking on the arrow dropdown functionality, two folders will appear. Dropdown the *src* folder, and then click on the dropdown icon again for the *codePkg* package file, and then a file called "StringReturner.java" should appear. Double-click on this file to work on it.

   The goal of this particular lab file is to *return the function's parameter* which is of type String. However, the makeup of the function's header is incorrect – the function needs to return the value of **type String**.

   Your task is to fill in the function code **and** to fix the function's header.

- *RESULTS*
  If I were to have ran your code by calling this function, I should receive back the string I entered in.

  For example, if I ran the following code as my main function code:

```
StringReturner strVar = new StringReturner();

System.out.println(strVar.returnThisString("Hello World!"));
```

  the following result I should receive is:

```
Hello World!
```

2) *Mini Functional Primitive Calculator*

First, download the folder package containing this particular code from the tutorial course website. Save it to a folder – this folder will be set as your Eclipse workspace. Open the folder package and copy the inner folder into the same folder you just saved this zipped folder package in. Then, open up Eclipse and set the folder you saved the folder package in as your Eclipse workspace.

In Eclipse, open the program folder called "Mini Functional Primitive Calculator" in the *Package Explorer* panel (on your left hand side). After clicking on the arrow dropdown functionality, two folders will appear. Dropdown the *src* folder, and then click on the dropdown icon again for the *codePkg* package file, and then a file called "MathematicsFun.java" should appear. Double-click on this file to work on it.

In this Java file, there are only five functions for you to fill in – *subtractTwoNumbers*, *multiplyTwoNumbers*, *divideTwoNumbers*, *addThreeNumbers*, and *calculateExpression*.

Each function will have a number of parameters for you to work with – it is your job to fill in return statements for each function based on the instructions of the ***TODO*** comment. (The *addTwoNumbers* function has been done for you to illustrate what you need to accomplish.)

For the last function, *calculateExpression*, instead of having to just use the parameters, try to reuse some of the other functions you have written in order to receive full marks. (We want to be lazy computer scientists! I figured this would be the easiest way to expose this laziness to you all!)

- *RESULTS*

If I were to have ran your code by calling any of these functions, your functions must return the desired value based on calculating what the value is from the values I pass into your functions.

For example, if I ran the following code as my main function code:

```
MathematicsFun easyMath = new MathematicsFun();

System.out.println(easyMath.addTwoNumbers(2, 5));

System.out.println(easyMath.multiplyTwoNumbers(10,20));

System.out.println(easyMath.divideTwoNumbers(3, 27));
```

the following result I should receive is:

```
7

200

9
```

3) *A Countdown Program*

First, download the folder package containing this particular code from the tutorial course website. Save it to a folder – this folder will be set as your Eclipse workspace. Open the folder package and copy the inner folder into the same folder you just saved this zipped folder package in. Then, open up Eclipse and set the folder you saved the folder package in as your Eclipse workspace.

In Eclipse, open the program folder called "A Countdown Program" in the *Package Explorer* panel (on your left hand side). After clicking on the arrow dropdown functionality, two folders will appear. Dropdown the *src* folder, and then click on the dropdown icon again for the *codePkg* package file, and then a file called "FastestCountdownEver.java" should appear. Double-click on this file to work on it.

You will see a function called *countdown* that takes in an integer parameter called *num*. Your task is to create an if statement for when *num* reaches the value 0. You want to first print out the value of *num*, and then return that same value after ***using num***.

- *RESULTS*

If I were to have ran your code by calling the countdown function, I would have all my numbers printed out, counting down towards 0, and then I would print out the sum value of all those numbers.

For example, if I ran the following code as my main function code:

```
FastestCountdownEver fce = new FastestCountdownEver();

System.out.println("The sum of all the numbers from the countdown is " + fce.countdown(10));
```

the following result I should receive is:

```
10
9
8
7
6
5
4
3
2
1
0
The sum of all the numbers from the countdown is 55.
```

4) *The Stepping Stone*

First, download the folder package containing this particular code from the tutorial course website. Save it to a folder – this folder will be set as your Eclipse workspace. Open the folder package and copy the inner folder into the same folder you just saved this zipped folder package in. Then, open up Eclipse and set the folder you saved the folder package in as your Eclipse workspace.

In Eclipse, open the program folder called "The Stepping Stone" in the *Package Explorer* panel (on your left hand side). After clicking on the arrow dropdown functionality, two folders will appear. Dropdown the *src* folder, and then click on the dropdown icon again for the *codePkg* package file, and then a file called "SteppingStone.java" should appear. Double-click on this file to work on it.

Your goal is to create a function called *steppedOnStone* that returns a *String value* based on whether or not the person has stepped on the stone. That is, if the person has stepped on the stone, return "*I have stepped on the stone.*" If the person has not stepped on the stone, return "*I did not step on this stone.*" You may use the already defined *hasSteppedOnStone* global variable to help you out.

- *RESULTS*

There are two scenarios here. Let me show you both results.

Scenario 1: If I didn't step on the stone…

```
SteppingStone ss = new SteppingStone();

System.out.println(ss.steppedOnStone());
```

the following result I should receive is:

```
I did not step on this stone.
```

Scenario 2: If I stepped on the stone…

```
SteppingStone ss = new SteppingStone();

ss.personSteppedOnStone();

System.out.println(ss.steppedOnStone());
```

the following result I should receive is:

```
I have stepped on the stone.
```

5) *Passing the Hot Potato*

First, download the folder package containing this particular code from the tutorial course website. Save it to a folder – this folder will be set as your Eclipse workspace. Open the folder package and copy the inner folder into the same folder you just saved this zipped folder package in. Then, open up Eclipse and set the folder you saved the folder package in as your Eclipse workspace.

In Eclipse, open the program folder called "Passing the Hot Potato" in the *Package Explorer* panel (on your left hand side). After clicking on the arrow dropdown functionality, two folders will appear. Dropdown the *src* folder, and then click on the dropdown icon again for the *codePkg* package file, and then a file called "Hot Potato.java" should appear. Double-click on this file to work on it.

Your task is broken up into two parts:

- to create two global variables that will need to keep track of if the hot potato has been passed around and the user who currently has the hot potato, and
- to create a function called *passTo* that will take in only one parameter that is of type String, called *name*.

A few things to note:

- The first global var is called *hotPotatoHasBeenPassed* and is currently set to false initially. It keeps track of whether or not it has been passed around.
- The second global var is called *userName* and is based on the current user's name that currently has the hot potato.
- In the *passTo* function, for any name passed through in the parameter, set the *hotPotatoHasBeenPassed* to true, because it has been passed to this particular person. In addition, set the *userName* to the name that was passed through the function's parameter.
- In the *passTo* function, if the strings "dropped" or "put on table" has been passed in the *name* parameter, then set the *hotPotatoHasBeenPassed* to false, and in turn set the *userName* as an empty string, which would just be empty quotation marks: "".

➢ ***Results?***

Actually, this would be a good time for you to do a bit of your own testing! :) Just be sure to follow the instructions above in order to get the functionality right.

Part III – Lab Code

Work on all 4 questions of your choice. Please be sure to create the proper package name based on the instructions here and properly name all functions *according to what has been written in the question itself*. **Please read each instruction carefully!** Also, be sure to mark down which questions you have worked on and write/type it on page 2 of this Homework Lab Set package.

1) **GPA Calculator**
   Create a class called *GPA_Calc*. Here, you will create a function called *percent_to_gpa* that takes a percentage mark (which is an integer) as input, and returns the corresponding Grade Point Average value (which is a float). The GPA is calculated as follows:

Percentage Mark	GPA
90-100	4.0
85-89	4.0
80-84	3.7
77-79	3.3
73-76	3.0
70-72	2.7
67-69	2.3
63-66	2.0
60-62	1.7
57-59	1.3
53-56	1.0
50-52	0.7
0-49	0.0

   You can safely assume that I will not give you values outside of the given ranges for testing.

2) Building our own String Representation Function

So far, we have been currently working with the System.out.println() functions to print out all sorts of values. However, I would like to build our own version so that we can see the process of what gets printed out dependant on what I pass into the function's parameter.

Create a class called *StringRepr*. You will implement a function called *my_str* that takes in an ***Object*** as an input, and returns a string representation of that object. However, we don't want to use our own boring old built-in representations in Java, so we will make our own.

- If the input is a string, just return it as is.
- If the object is a Boolean, return "YES" for *true* and "NO" for *false*.
- If the object is an integer:
    - If it's less than 0, return "Negative Number".
    - If it's 0 - 10, return "Small Number".
    - If it's 11 – 99, return "Medium Number".
    - If it's greater than or equals 100, return "Large Number".
- If the object is any other data type, simply return the phrase, "I dunno what this is."

Here is some sample output for you to work with, if I ran this code in a main function:

```
StringRepr sr = new StringRepr();

System.out.println(sr.my_str("Hello world!"));

System.out.println(sr.my_str(-10));

System.out.println(sr.my_str(42));

System.out.println(sr.my_str(true));

System.out.println(sr.my_str('c'));
```

Here are the results:

```
Hello world!

Negative Number

Medium Number

YES

I dunno what this is.
```

When implementing this program, be sure to use

<variable/parameter name> isinstanceof <type>

by replacing the angle brackets with the actual items listed inside.

3) Having a Bit of Fun with Recursion

Create a class called *MiniRecursionFun*. Here, we will be working with a function called *numericalRecursion* that will take in an integer input.

In the same function, if the number is a number that is divisible by 3, I would simply add the number by 2 and then return the function call with that number inside. For example, if the name of my parameter is called *num* and *num* **is divisible by three**, then the following recursive code would be:

```
return numericalRecursion(num + 2);
```

In this function, create an if statement where if the number is an even number, multiply the number by 2 and then return the function call with that number inside. For example, if the name of my parameter is still called *num* and *num* **is an even number**, then the following recursive code would be:

```
return numericalRecursion(num * 2);
```

However, if the parameter numerical value is greater than 100, then simply return the value. This return statement should be placed at the very beginning of the function since it will be easier to work with. Again, if the name of my parameter is called *num*, and *num* **equals 102**, for example, then we want to just execute the following return statement:

```
return num;
```

Just like in Part II, specifically in one of the code fill-ins, we do not need to have a loop to accomplish all of this.

4) Sevens Out

You will write a program to play a dice game called Sevens Out. It involves rolling two dice and using the sum of the dice. If you roll a sum of seven the game is over. Otherwise, you will get the sum of the dice added to your score. **If you roll the same number on both dice** (this is called *rolling doubles*), you will get double the score.

First, create a class called *SevensOut*. Then, create a global variable called *sum* (this will keep track of the current score) that will equal to 0 initially.

Then, write a function called *rollDice* that **does not take any parameters but will return the** *sum* **value**. Here, you will implement the rules of the game (listed above).

Here is a sample output of when I play this game. Based on a main function I have written, here is the first run results:

```
2, 1 This roll = 3. SCORE = 3
```

Here is the second run results:

```
3, 2 This roll = 5. SCORE = 5
```

Here is the third run results:

```
5, 5 This roll = 20. SCORE = 20
```

Here is the last run results:

```
6, 1 This roll = 7. YOU LOSE.
```

Notice how the program shows what dice were rolled on each turn. It also shows how many points you got on that roll. Also, take note of when you get double the points for rolling doubles (5 and 5 gives 20, 3 and 3 gives 12, and so on so forth). It also shows the current score so far in that game. When you re-run the game and receive a score of 7 (in this case, it rolled a 6 and a 1), it will print out "YOU LOSE". This is not an interactive game – you will need to keep on running the Java file with the main function to play the game.

To properly code this, you will need the help of a random integer variable. First, right after the "package" line, type in the following import statement into your code.

```
import java.util.Random;
```

The random variable will have the form

int <variable name> = new Random().nextInt(<value range>);

where this value range will be from 0 to whichever value you enter in, minus 1 however.

To illustrate how it works, imagine the following code:

```
int someRandomNumber = new Random().nextInt(10);
```

The variable *someRandomNumber* will randomly choose a number from 0 to 9, NOT 10. That will be the assigned number for that variable. If the Random class function chose 7, then the variable *someRandomNumber* will be assigned 6 due to the range (the value 7 is actually 6 in the range, when you count 0 as the first number).

To get rid of the problem of NOT choosing the actual value 10, we would need to add 1 to it.

```
int someRandomNumber = new Random().nextInt(10) + 1;
```

This way, we can only access the range of numbers of 0 to 9 first, randomly receive a number from the Random class function, and then add 1 to it. So, if the randomizer has chosen 9, then 9 + 1 will give you 10, thus assigning the variable *someRandomNumber* to be 10.

Armed with this knowledge, you can use the following code to roll a random die value:

```
int die1 = new Random().nextInt(6) + 1;
```

SOURCE: https://www.mkyong.com/java/java-generate-random-integers-in-a-range/

Submission *– Instructions on how to submit code + worksheets*

Upon finishing your Homework Lab Set, you will need to submit the following:

- These worksheets/document with answers, with your name filled out
- Your code for Part II and Part III

You may submit your worksheets/document with answers through email (please send it to evanchen.cs@gmail.com) or through Slack (you **must submit it to me personally on Slack, do not share it with the whole channel**). You can also hand it to me in person, if you wish to do so.

HOWEVER. You must submit your code on a platform called Github. (If you have not signed up on Github, please do so here: https://github.com). You must share with me your Github handle (otherwise known as your Github username) and I will create a repository with you. ***Please do not put your code in a zip file; just upload everything to Github via "drag-and-drop" option.***

All of the code is due at 11:59 pm on July 20, 2018, while the worksheets are due at 5 pm on July 20, 2018.

Failure to hand it to me on time will result in a 20% reduction on this homework lab set.