DUE DATE: **By the end of your lab session**

CSE-271: Object-Oriented Programming

Exercise #8

Max Points: 20

Name: Jacob Igel

For your own convenient reference – You should first save/rename this document using the naming convention `MUid_Exercise8.docx` (example: `raodm_Exercise8.docx`) prior to proceeding with this exercise.

Objectives: The objectives of this exercise are to:

1. Review the concepts of recursion
2. Trace the operation of recursive methods
3. Appreciate the relationship between iterative and recursive solutions
4. Practice the use of recursion for problem solving

Fill in answers to all of the questions. For some of the questions you can simply copy-paste appropriate text from Eclipse output into this document. You may discuss the questions or seek help from your neighbor, TA, and/or your instructor.

Part #0: One time setup of Eclipse (IDE) – Only if needed

We already configured `Eclipse`'s source formatter and Checkstyle plug-in as part of Lab #1. If your `Eclipse` is not configured (because you are using a different computer) then use the instructions from Lab #1 to configure `Eclipse`.

Part #1: Recursion: Concepts & Tracing

Estimate time: < 50 minutes

Background (from Wikipedia): In computer science, recursion is a method of solving a computational problem where the solution depends on solutions to smaller instances of the same problem Recursion solves such recursive problems by using functions that call themselves from within their own code. The approach can be applied to many types of problems, and hence, recursion is one of the central ideas of computer science.

> The power of recursion evidently lies in the possibility of defining an infinite set of objects by a finite statement. In the same manner, an infinite number of computations can be described by a finite recursive program, even if this program contains no explicit repetitions.
>
> — **Niklaus Wirth**, Algorithms + Data Structures = Programs, 1976

Exercise: Briefly (2-to-3 sentences each) respond to the following questions regarding generic concepts of Graphical User Interface (GUI)

1. What is a key characteristic of a problem that lends itself to using recursion for problem-solving?

   Breaking a larger problem into smaller sub-problems. Then the solutions for the sub-problems can be used to solve the main problem.

2. Briefly describe the two parts of any recursive solution (*i.e.*, base case and recursive case)

   Base Case – Conditional parts of the methods that don't make any further recursive calls. They can also make other non-recursive calls as needed.

   Recursive Part – These parts of the method make recursive calls (self or mutual). This breaks the problem into smaller sub-problems.

3. Run the adjacent recursive `main` method (that will generate an exception) and copy-paste the output below **showing the exception and number of recursive calls accomplished**. Briefly (about 1-to-2 sentences) describe the source of the error.

```
public class Endless {
  public static void main(String args[]) {
    int num = 0;
    if (args.length > 0) {
      num = Integer.parseInt(args[0]);
    }
    System.out.println(num);
    String next = Integer.toString(num + 1);
    Endless.main(new String[] { next });
  }
}
```

```
<terminated> tester [Java Application] /Library/Java/JavaVirtualMachines/jdk-11.0.13.jdk/Contents/Home/bin/java  (Apr 1, 2022, 10:18:22
6366
6367
6368
6369
6370
6371
6372
6373
6374
6375
6376
6377
6378
6379
6380
6381
6382
6383
6384
6385
6386
6387
Exception in thread "main" java.lang.StackOverflowError
        at java.base/sun.nio.cs.UTF_8$Encoder.encodeLoop(UTF_8.java:564)
        at java.base/java.nio.charset.CharsetEncoder.encode(CharsetEncoder.java:576)
        at java.base/sun.nio.cs.StreamEncoder.implWrite(StreamEncoder.java:292)
        at java.base/sun.nio.cs.StreamEncoder.implWrite(StreamEncoder.java:281)
        at java.base/sun.nio.cs.StreamEncoder.write(StreamEncoder.java:125)
        at java.base/java.io.OutputStreamWriter.write(OutputStreamWriter.java:208)
        at java.base/java.io.BufferedWriter.flushBuffer(BufferedWriter.java:120)
        at java.base/java.io.PrintStream.write(PrintStream.java:605)
        at java.base/java.io.PrintStream.print(PrintStream.java:676)
        at java.base/java.io.PrintStream.println(PrintStream.java:812)
        at tester.main(tester.java:8)
        at tester.main(tester.java:10)
        at tester.main(tester.java:10)
```

The method used 6387 recursive calls. This method generates an infinite number of recursion calls and eventually, the computer runs out of stack space and then generates a 'StackOverflowError' Exception.

4. The web-site `StackOverflow.com` is widely used by programmers. Where do you think this name originates from?

   "The website serves as a platform for users to ask and answer questions, and, through membership and active participation, to vote questions and answers up or down similar to Reddit and edit questions and answers in a fashion similar to a wiki."

   Source: https://en.wikipedia.org/wiki/Stack_Overflow

5. Consider the adjacent recursive `mystery` method.

```
int mystery(int [] array, int s, int e) {
  if (s == e)
     return array[s];
  int rest = mystery(array, s + 1, e);
  if (array[s] > rest)
     return array[s];
  return rest;
}
```

   a. What is the stopping case for the recursive method?

```
if (s == e)
```

For the following sub-questions, assume that variable `A` is an integer array with values `{3, 8, 2, -10, 5, 7, -11}`.

b. What happens when you execute `mystery(A, 2, 4)`? For this problem trace the recursive calls by illustrating parameters to each call to the `mystery` method in each row of the table below. Copy-paste the first row and update argument-values and notes suitably (add more rows to the table if needed).

Method call (with arguments)	Notes (briefly describe logic performed)
`Mystery(A, 4, 4)`	Returns 5
`Mystery(A, 3, 4)`	array[3] (i.e, -10) is not > rest (which is 5). So returns 5.
`mystery(A, 2, 4)`	array[2] (i.e, 2) is not > rest (which is 5). So returns 5.

c. What is the final return value from the method call `mystery(A, 0, 6)`?

8

d. What is the final return value from the method call `mystery(A, 3, 2)`?

Stack Overflow Error Exception

6. Study the recursive `mystery` method shown below and answer the following questions.

```
int mystery(int num) {
  if (num == 0) {
    return 0;
  } else {
    return num+mystery(num-1);
  }
}
```

a. What is the return value from `mystery(5)`?

15

b. What is the return value from `mystery(0)`?

0

c. What is the return value from `mystery(-5)`?

Stack Overflow Error Exception

7. Recursion and iteration are related in that all recursive solutions can be converted to iterative solutions. Convert the following methods between iterative or recursive versions by suitably implementing the missing version

Iterative	Recursive

int factorial(int n) {
int fact = 1;
for (int i = 2; (i <= n); i++, fact += (i-1)) {}
return fact;
}
	int factorial(int n) {
	return (n < 2) ? 1 : (n * factorial(n - 1));
	}

public int reverse(int n) { int rev = 0;	public int reverse(int n) { if (n < 10) {

```
  for (; (n > 0); n /= 10) {
    rev = (rev * 10) + (n % 10);
  }
  return rev;
}
// Note: Math.log10 is handy for recursion!
```

```
return n;
}
int digit = (int) Math.log10(n);
return ((n % 10) * Math.pow(10, digit)) + reverse(n / 10);
}
```

8. Complete the following recursive method that recursively computes the value of the expression `x[0]*y[0] + x[1]*y[1] + … + x[num]*y[num]`. If `num` is larger than the length of the arrays then it should throw an `IllegalArgumentException`.

```
int arrayproduct(int[] x, int[] y, int num) {
    if (num >= x.length || x.length != y.length) {
       throw new IllegalArgumentException("Different array length");
    int result = x[num] * y[num];
    if (num > 0) {
        result += arrayproduct(x, y, num - 1);
    }
return result;

}
```

Part #2: Practice developing recursive methods

*Estimate time: < 7 minutes * 5 methods = 35 minutes*

Background: Recursive thinking for problem-solving requires practice.

Setup: First, you need to setup a Java project in Eclipse, download the starter code and add them to your Java project. In this part of the exercise, you are given the following set of classes:

File Name	***Description***
`RecursionTest.java`	A `JUnit` class for testing. **Do not modify this class**.
`RecursionPractice.java`	In this class, you should use the comments to suitably implement the 5 methods.

Exercise: Using recursion, implement the 5 methods in the `RecursionPractice.java` class.

- Use the Javadoc comments at the beginning of each method to help you implement each method.

- Remove the dummy return statement in each method.
- After implementing a method, you may use the supplied `JUnit` tests in the `RecursionTest.java` class for testing your implementation.

Canvas-CODE Testing: When you upload your solution to Canvas-CODE the following output will be expected from your program.

Expected output:

```
Running tests in RecursionTest
.....

OK (5 tests)
Finished. Result: Failures: 0. Ignored: 0. Tests run: 5
```

Part #3: Submit to Canvas via CODE plug-in

Estimated time: < 5 minutes

Exercise: You will be submitting the following files via the Canvas CODE plug-in:

1. This MS-Word document saved as a PDF file – **Only submit PDF file**.
2. The Java source file `RecursionPractice.java` that you modified in this exercise.

Ensure you actually complete the submission on Canvas by verifying your submission (after you submit)