# Gregory Prosper COP4331

# Problem 1

## Fib.java

**package** q1;

**import** java.util.InputMismatchException;

**import** java.util.Scanner;

/\*\* Computes the Fibonacci Series \*/

**public** **class** Fib {

/\*\* Constructor for Fib

\* **@param** f0 is the first number in series

\* **@param** f1 is the first number in series

\*/

**public** Fib(**int** f0, **int** f1)

{

**this**.f0 = f0;

**this**.f1 = f1;

}

/\*\* computes F(n) using an \*\*\*iterative\*\*\* algorithm

\* **@param** n is the index for the series. Series is printed to index n

\* **@return** returns F(n)

\*/

// use instance variables that store F(0) and F(1).

// check parameter and throw exception if n < 0. Don't worry about arithmetic overflow.

**public** **int** f(**int** n){

**this**.f0 = 0;

**this**.f1 = 1;

**if** (n < 0) **throw** **new** IllegalArgumentException("n must be positive");

**if** (n == **this**.f0) **return** **this**.f0;

**if** (n == **this**.f1) **return** **this**.f1;

**int** sum = 0;

**for** (**int** i = 2; i <= n; i++){

sum = **this**.f0 + **this**.f1;

**this**.f0 = **this**.f1;

**this**.f1 = sum;

}

**return** sum;

}

/\*\* computes F(n) using the \*\*\*recursive\*\*\* algorithm

\* **@param** n is the index for the series. Series is printed to index n

\* **@return** returns F(n)

\*/

// use instance variables that store F(0) and F(1).

// check parameter and throw exception if n < 0. Don't worry about arithmetic overflow.

**public** **int** fRec(**int** n) {

**if** (n < 0) **throw** **new** IllegalArgumentException("n must be positive");

**if** (n == 0){

**return** 0;

}

**if** (n == 1){

**return** 1;

}

**return** fRec(n-1) + fRec(n-2);

}

/\*\* main function

\* **@param** args the command line arguments, stored in a String array

\*/

**public** **static** **void** main(String[] args)

{

**try**{

// get numbers F(0) and F(1) from args[0] and args[1].

// use either the Scanner class or Integer.parseInt(args[...])

// you must handle possible exceptions !

Scanner in = **new** Scanner(System.*in*);

System.*out*.print("Enter F(0): ");

**int** arg1 = in.nextInt();

System.*out*.print("Enter F(1): ");

**int** arg2 = in.nextInt();

// get n from args[2]:

System.*out*.print("Enter F(n): ");

**int** n = in.nextInt();

in.close();

// create a Fib object with params F(0) and F(1)

Fib obj = **new** Fib(arg1,arg2);

// calculate F(0), ..., F(n) and display them with System.out.println(...) using

// the iterative methode f(i)

System.*out*.printf("F(%d) using iterative method\n",n);

**for** (**int** i = 0; i <= n; i++){

System.*out*.print(obj.f(i));

System.*out*.print(" ");

}

System.*out*.print("\n");

// calculate F(0), ..., F(n) and display them with System.out.println(...) using

// the recursive methode fRec(i)

System.*out*.printf("F(%d) using recursive method\n",n);

**for** (**int** i = 0; i <= n; i++){

System.*out*.print(obj.fRec(i));

System.*out*.print(" ");

}

}

**catch** (IllegalArgumentException ex){

System.*out*.println(ex);

System.*out*.println("\nTry again.");

*main*(args);

}

**catch** (InputMismatchException ex){

System.*out*.println(ex);

System.*out*.println("\nTry again.");

*main*(args);

}

}

// instance variables store F(0) and F(1):

**int** f0;

**int** f1;

}

# Problem 2

## Greeter.java

**package** q2;

**public** **class** Greeter{

/\*\*

Constructs a Greeter object that can greet a person or

entity.

**@param** aName the name of the person or entity who should

be addressed in the greetings.

\*/

**public** Greeter(String aName){

name = aName;

}

/\*\*

Greet with a "Hello" message.

**@return** a message containing "Hello" and the name of

the greeted person or entity.

\*/

**public** String sayHello(){

**return** "Hello, " + name + "!";

}

/\*\*

Takes the Greeter parameter and replaces this name with the other.

**@param** other Greeter object

\*/

**public** **void** swapNames(Greeter other) {

**this**.name = other.name;

}

/\*\*

Returns new Greeter object with string qualifier + this object's name

**@param** qualifier string parameter

**@return** Greeter object

\*/

**public** Greeter createQualifiedGreeter (String qualifier) {

**return** **new** Greeter(qualifier + " " + **this**.name);

}

**private** String name;

}

## GreeterTester.java

**package** q2;

**public** **class** GreeterTester {

/\*\* main function

\* **@param** args the command line arguments, stored in a String array

\*/

**public** **static** **void** main(String[] args) {

Greeter g = **new** Greeter("World");

Greeter g2 = g.createQualifiedGreeter("Beutiful");

System.*out*.println(g2.sayHello());

g.swapNames(g2);

System.*out*.println(g.sayHello());

}

}

# Problem 3

## DataAnalyzer.java

**package** q3;

**import** java.util.ArrayList;

**import** java.util.LinkedList;

/\*\* Object to analyze data \*/

**public** **class** DataAnalyzer {

/\*\* Constructor for DataAnalyzer

\* **@param** numList LinkedList with numbers to be analyzed

\*/

**public** DataAnalyzer(LinkedList<Integer> numList) {

Integer nums[] = numList.toArray(**new** Integer[numList.size()]);

**for** (**int** i = 0; i < nums.length; i++){

**this**.list.add(nums[i]);

}

}

/\*\* Calculates minimum number in list

\* **@return** minimum number in list

\*/

**public** **int** min() {

**int** num = **this**.list.get(0);

**for** (**int** i = 0; i < **this**.list.size(); i++){

**if** (num > **this**.list.get(i)){

num = **this**.list.get(i);

}

}

**return** num;

}

/\*\* Calculates maximum number in list

\* **@return** maximum number in list

\*/

**public** **int** max() {

**int** num = **this**.list.get(0);

**for** (**int** i = 0; i < **this**.list.size(); i++){

**if** (num < **this**.list.get(i)){

num = **this**.list.get(i);

}

}

**return** num;

}

/\*\* Calculates average number in list

\* **@return** average number in list

\*/

**public** **double** average() {

**double** total = 0;

**for** (**int** i = 0; i < **this**.list.size(); i++){

total += **this**.list.get(i);

}

**return** total / (**double**) **this**.list.size();

}

ArrayList<Integer> list = **new** ArrayList<>();

}

## DataAnalyzerTester.java

**package** q3;

**import** java.io.FileNotFoundException;

**import** java.io.PrintWriter;

**import** java.io.UnsupportedEncodingException;

**import** java.util.InputMismatchException;

**import** java.util.LinkedList;

**import** java.util.Scanner;

/\*\* Class to Test DataAnalyzer \*/

**public** **class** DataAnalyzerTester {

/\*\* main function

\* **@param** args the command line arguments, stored in a String array

\*/

**public** **static** **void** main(String[] args) {

**try**{

Scanner in = **new** Scanner(System.*in*);

PrintWriter writer;

LinkedList<Integer> nums = **new** LinkedList<Integer>();

System.*out*.println("How many numbers do you wish to analyze: ");

**int** numbersToAnalyze = in.nextInt();

**for** (**int** i = 0; i < numbersToAnalyze; i++){

nums.add(in.nextInt());

}

DataAnalyzer analyzer = **new** DataAnalyzer(nums);

System.*out*.print("\nMin: ");

System.*out*.println(analyzer.min());

System.*out*.print("Max: ");

System.*out*.println(analyzer.max());

System.*out*.print("Average: ");

System.*out*.println(analyzer.average());

System.*out*.println("Enter File Name to save as. (omit .txt)");

**try** {

writer = **new** PrintWriter(in.next() + ".txt", "UTF-8");

writer.print("\nMin: ");

writer.println(analyzer.min());

writer.print("\nMax: ");

writer.println(analyzer.max());

writer.print("\nAverage: ");

writer.println(analyzer.average());

writer.close();

} **catch** (FileNotFoundException e) {

System.*out*.println(e);

System.*out*.println("\nTry again.");

*main*(args);

} **catch** (UnsupportedEncodingException e) {

System.*out*.println(e);

System.*out*.println("\nTry again.");

*main*(args);

}

**finally**{

in.close();

}

}

**catch** (InputMismatchException e){

System.*out*.println(e);

System.*out*.println("\nTry again.");

*main*(args);

}

}

}

# Problem 4

What is the value of x after the following code is executed? Explain what happens.

int x = 0;

try {

Greeter g1 = new Greeter("Alice");

Greeter g2 = new Greeter("Alice");

if (g1.sayHello() != g2.sayHello()) {

g2 = null;

}

x = 1;

System.out.println(g2.sayHello());

x = 2;

} catch (IOException ex) {

x = 4;

} catch (NullPointerException ex) {

x++;

} finally {

x++;

}

X is 3 after the code is executed. g1 and g2 both return the same string after sayHello() is called so g2 = null is not executed. There is no IOException or NullPointerException thrown so X = 4 or X++ is never executed. Finally will execute every time so X++ is executed. X equals 3 at the end.