First Name: _____ **Last Name**: _____

McGill ID: _____ **Section**: _____

Faculty of Science
COMP-202A - Introduction to Computing I (Fall 2011) - All Sections
Final Examination

December 21, 2011
14:00 - 17:00

Examiners: Daniel Pomerantz [Sections 1 and 3]
Jörg Kienzle [Section 2]

Instructions:

- **DO NOT TURN THIS PAGE UNTIL INSTRUCTED**
- This is a **closed book** examination; only a letter-sized (8.5" by 11") **crib sheet** is permitted. This crib sheet can be single or double-sided; it can be handwritten or typed. Non-electronic translation dictionaries are permitted, but instructors and invigilators reserve the right to inspect them at any time during the examination.
- Besides the above, only writing implements (pens, pencils, erasers, pencil sharpeners, etc.) are allowed. The possession of any other tools or devices is prohibited.
- Answer **all** questions **on this examination paper** and return it. **If you need additional space**, use page 23-25, or the booklets supplied upon request, and clearly indicate where each question is continued. **In order to receive full marks for a question, you must show all work** unless otherwise stated.
- This examination has **26** pages including this cover page, and is printed on both sides of the paper. On page 26, you will find information about **useful classes and methods**. **You may detach page 26 from the examination if you wish.**

1	2	3	Subtotal
/10	/9	/6	/25

4	5	Subtotal
/15	/5	/20

6	7	8	9	Subtotal
/10	/10	/25	/10	/55

Total
/100

Section 1 - Short questions

[10]

1. In each of the following you should write an expression or short piece of Javacode. (10 points)

   - Given a `boolean b1` and a `boolean b2` write a boolean expression indicating whether at least one of them is true.

   - Given that the two legs (i.e. not hypotenuse) of a right triangle have lengths `A` and `B` (`doubles`), write a Java expression to calculate the length of the hypotenuse in terms of `A` and `B`. (Remember that if the two legs of a right triangle have length A and B, then the length of the hypotenuse H can be calculated using the formula $H^2 = A^2 + B^2$)

   - Given an `int[] array` of size 3, write a boolean expression indicating whether exactly 2 values in the array are even.

   - Given an `ArrayList<String> words` write a boolean expression to check if the first `String` in `words` has the contents "Hello." If `words` is `null` or the size of `words` is 0, then your expression should evaluate to be false. (Note: If you can't write this as just one expression, you can write a full method instead for full credit)

   - Write a for loop to print all integers from 1 to 100.

[9]

2. In this section you will be shown several classes. Inside the code for each class will be a question. You should provide your answer to the question inside the chart.

   Many of the questions will ask you whether a variable has changed or not. In cases that the variable being asked about is a primitive type, you should provide your answer based on the data stored in the primitive type. In cases that the variable being asked about is a reference type, you should provide your answer based on the data stored *at the Object linked to by the reference type*.

   For example, in the first snippet, the question asks about the variable x defined in the main method. Since x is a primitive type, you should answer YES if you think the variable x has had its value changed after the method methodOne() returns.

   All your answers should go in the chart below: (9 points)

Part	Answer to question
a	x changed (circle one)? YES NO
b	y changed (circle one)? YES NO
c	a changed (circle one)? YES NO
c	b changed (circle one)? YES NO
d	a changed (circle one)? YES NO
d	b changed (circle one)? YES NO
e	words changed (circle one)? YES NO
f	Values in array of array referred to by y:
g	What is printed to the screen?

(a)
```
public class ReferenceTest1{
     public static void methodOne(int x) {
          x++;
     }

     public static void main(String[] args) {
          int x = 0;
          methodOne(x);
          //IS THE DATA STORED OR REFERRED TO BY x THE SAME? (ANSWER IN CHART)
     }

}
```

(b)
```
public class ReferenceTest2 {
     public static void methodTwo(int[] x) {
          x = new int[10];
          x[0]++;
     }

     public static void main(String[] args) {
          int[] y = {1, 2, 3, 4, 5};
          methodTwo(y);
           //IS THE DATA STORED OR REFERRED TO BY y THE SAME? (ANSWER IN CHART)
     }
}
```

(c)
```
public class ReferenceTest3 {
        public static void methodThree(int[] x, int[] y) {
              x[0]++;
              x = new int[5];
              y = x;
              y[0]++;
        }

        public static void main(String[] args) {
              int[] a = {1,2,3};
              int[] b = {3,4,5};
              methodThree(a,b);
             //IS THE DATA STORED OR REFERRED TO BY a THE SAME? (ANSWER IN CHART)
             //IS THE DATA STORED OR REFERRED TO BY b THE SAME? (ANSWER IN CHART)
        }
   }
```

(d)
```
public class ReferenceTest4 {
        public static void methodFour(int[] x, int[] y) {
              x[0]++;
              x = y;
              x[0]++;
        }

        public static void main(String[] args) {
              int[] a = {1,2,3};
              int[] b = {100,101,102};
              methodFour(a,b);
             //IS THE DATA STORED OR REFERRED TO BY a THE SAME? (ANSWER IN CHART)
             //IS THE DATA STORED OR REFERRED TO BY b THE SAME? (ANSWER IN CHART)
        }
   }
```

(e)
```
public class ReferenceTest5 {
        public static void methodFive(String[] x) {
              String first = x[1];
              first = "Love";
        }

        public static void main(String[] args) {
              String[] words = {"I", "Hate", "Computers"};
              methodFive(words);
             //IS THE DATA STORED OR REFERRED TO BY words THE SAME? (ANSWER IN CHART)
        }
   }
```

(f)
```
public class ReferenceTest6 {
        public static void methodSix(int[][] x) {
              x[0] = new int[2];
              x[0][0] = 0;
              x[0][1] = 1;
              x = new int[10][5];
              x[0][3] = 3;
        }

        public static void main(String[] args) {
              int[][] y = { {1,2,3}, {2,4}};
              methodSix(y);
             //WHAT ARE THE VALUES IN THE ARRAY OF ARRAYS
             //REFERRED TO BY y AT THIS POINT? (ANSWER IN CHART)
        }
   }
```

(g) Suppose I define a type `Point` which contains two properties, `x` and `y` . The `Point` class has both getters and setters defined on it which do the obvious thing of setting and getting x and y.

Remember that the `get(int index)` method defined on an `ArrayList` will return a reference to the `index` element of the `ArrayList`.

```
import java.util.ArrayList;
public class ReferenceTest7{
      public static void methodSevenA(ArrayList<Point> coordinates) {
         Point first = coordinates.get(0);
         first.setX(200.0);
      }

     public static void methodSevenB(ArrayList<Point> coordinates) {
         Point second = coordinates.get(1);
         second = new Point(5.0,100.0);
         second.setX(100);
      }

      public static void main(String[] args) {
            ArrayList<Point> coordinates = new ArrayList<Point>();
            coodinates.add(new Point(1.0,2.0));
            coordinates.add(new Point(3.0,4.0));

            methodSevenA(coordinates);
            methodSevenB(coordinates);

            System.out.println(coordinates.get(0).getX());
            System.out.println(coordinates.get(1).getX());
            //WHAT IS PRINTED TO THE SCREEN? (ANSWER IN CHART)
      }
}
```

[6] 3. What is printed to the screen when the following program is run? (6 points)

```
public class AliasTest {
      public static void main(String[] args) {
            int[] foo = {1,2,3,4,5};
            int[] bar = {20,21,22,23};
            int[] chocolate = bar;

            bar[0] = 30;
            chocolate[1] = chocolate[0] + foo[0];

            System.out.println("Phase 1");
            System.out.println(bar[0]);
            System.out.println(chocolate[0]);
            System.out.println(foo[0]);
            System.out.println(bar[1]);

            bar = foo;
            foo = chocolate;
            foo[3] = foo[2] + foo[1];

            System.out.println("Phase 2");
            System.out.println(foo[1]);
            System.out.println(foo[2]);
            System.out.println(foo[3]);

            foo = new int[10];
            foo[0] = 3;

            System.out.println("Phase 3");
            System.out.println(chocolate[0]);
            System.out.println(chocolate[1]);
            System.out.println(chocolate[2]);
            System.out.println(chocolate[3]);

            //WHAT IS PRINTED TO THE SCREEN IN THIS RUN?
      }
}
```

What is printed to the screen when the above program is run?

Debugging and Short Programming Questions

[15]

4. In this question, you will be shown a snippet of code. You will then be shown the output from either the compiler or the Java virtual machine and be asked what the error is and how to fix it. In each case, your answer should include the line number of the problem, a one or two sentence explanation as to why the issue occurs, and a specific fix for the problem. Your fix should mention the exact line or lines of code you'd add/change/delete.

   Total for this question (5 parts): (15 points)

   (a) Consider the following class:

```
1  import java.util.Scanner;
2  public class HockeyStandings {
3      //This method should calculate the percentage of games in which the team
           lost.
4      public static double computePercentageLosses(int numWins, int numLosses) {
5            return numLosses / (numWins + numLosses) * 100;
6      }
7
8      public static void main(String[] args) {
9            Scanner reader = new Scanner(System.in);
10           System.out.println("Enter the number of wins for the Maple Leafs");
11           int numLeafWins = reader.nextInt();
12           System.out.println("Enter the number of losses for the Maple Leafs");
13           int numLeafLosses = reader.nextInt();
14
15           System.out.println("Enter the number of wins for the Canadiens");
16           int numCanadiensWins = reader.nextInt();
17           System.out.println("Enter the number of losses for the Canadiens");
18           int numCanadiensLosses = reader.nextInt();
19
20           double mapleLeafsRecord = computePercentageLosses(numLeafWins,
                 numLeafLosses);
21           double canadiensRecord = computePercentageLosses(numCanadiensWins,
                 numCanadiensLosses);
22
23           if (canadiensRecord > mapleLeafsRecord) {
24                System.out.println("Of course the Leafs did worse.");
25           }
26           else {
27                System.out.println("Wow, that hasn't happened for a long time!");
28           }
29     }
30 }
```

   The point of the program is to ask the user to enter numbers representing the wins and losses of the Maple Leafs and Canadiens in a season.

   The code compiles successfully and when you run the program, you enter the values:

   `2 80 40 42`

   into the keyboard. You expect that the program will output the message `Of course the Leafs did worse` since they had fewer wins. However, to your surprise, the output of the program is:

   `Wow, that hasn't happened for a long time!`

   Why is this the case and fix the error?

(b) Consider the following classes:

```
1  public class House
2  {
3        private String[] rooms;
4        public House() {
5               String[] rooms = new String[3];
6               rooms[0] = "living room";
7               rooms[1] = "kitchen";
8               rooms[2] = "bedroom";
9        }
10
11       public void printRoomList() {
12              for (int i=0; i < rooms.length; i++) {
13                     System.out.println(rooms[i]);
14              }
15       }
16 }
```

```
1  public class Block
2  {
3        public static void main(String[] args)
4        {
5                 House myHouse = new House();
6                 myHouse.printRoomList();
7        }
8  }
```

The program compiles, but when you run the program, you get the following exception:

```
Exception in thread "main" java.lang.NullPointerException
        at House.printRoomList(House.java:12)
        at Block.main(Block.java:6)
```

The point of the method `printRoomList` is to print all the Strings in the house to the screen. Why does it give the error and write the code to fix the error while successfully printing all the rooms to the screen?

(c) Consider the following class:

```
1  public class StudentGrades {
2      private int[] grades;
3
4      //loads values into grades from a file
5      public void initializeFromFile(String filename) { .... }
6
7      //gets the total number of grades entered
8      public int getNumRecords() { .... }
9  }
```

```
1  public class WebCT  {
2       public static void main(String[] args) {
3           StudentGrades grades = new StudentGrades();
4           grades.initializeFromFile("records.tsv");
5           System.out.println("There are " + grades.length + " grades in that
                file.");
6       }
7  }
```

The goal of the program is to load the files and then print the total number of grades in the file. However, when you try to compile the above classes, you get the following error:

```
WebCT.java:5: cannot find symbol
symbol  : variable length
location: class StudentGrades
        System.out.println("There are " + grades.length + " grades in that file.");
```

Explain why the error occurs and write the code to fix it (while maintaining the intended functionality).

(d) Suppose there is a class `BookStore` and inside this, is a method with the following header:
`public boolean stockWarehouse(String filename) throws NotEnoughFundsException`

You then write the following code inside the class `ShoppingMall`:

```
1  import java.util.Scanner;
2
3  public class ShoppingMall {
4  ...
5      public initialize void initialize(String[] args) {
6             BookStore indigo = new BookStore();
7             Scanner reader = new Scanner(System.in);
8             System.out.println("Please enter the file with the list of contents to
                   buy.");
9             String filename = reader.nextLine();
10            indigo.stockWarehouse(filename);
11     }
12 }
```

However, you get the following error:

```
ShoppingMall.java:10: unreported exception NotEnoughFundsException; must be
 caught or declared to be thrown
           indigo.stockWarehouse(filename);

1 error
```

1)Explain why the error occurs and write the code to fix it.

2)What code would you add if you wanted to change it so that the user was able to continue entering a filename until the method `initializeInventoryFromFile` could successfully be exceuted?

(e) Suppose I have the following class definition:

```
1  //A TelevisionShow is a new type that consists of a String[] for the actors
2  //a String for the name, and a String for the network
3  public class TelevisionShow {
4      private String[] actors;
5      private String name;
6      private String network;
7
8      //this method will return whether or not the property
9      //actors contains actorName or not
10     public static boolean hasActor(String actorName) {
11         for (int i=0; i < actors.length; i++) {
12             if (actors[i].equals(actorName)) {
13                 return true;
14             }
15         }
16
17         return false;
18     }
19 }
```

Unfortunately, when I try to compile this code, I get the following error:

```
TelevisionShow.java:11: non-static variable actors cannot be
referenced from a static context
        for (int i=0; i < actors.length; i++) {
                         ^
TelevisionShow.java:12: non-static variable actors cannot be
referenced from a static context
            if (actors[i].equals(actorName)) {
                ^
2 errors
```

What does this error mean and how can I fix it?

Recursion Question

[5]

5. In the following question, you will be asked to program the following using recursion. If you have an iterative solution, you may use it, but the most you can get for the question will be half credit. (5) points)

   Consider the following mathematical function `h` which operates on integers.

   $$h(n) = \begin{cases} 1, & \text{if } n \leq 2 \\ h(n-1) + n & \text{if } n > 2 \text{ and n is even} \\ h(n-2) + 2 * n & \text{if } n > 2 \text{ and n is odd} \end{cases}$$

   In Java, use recursion to write a method `h` which takes as input an `int n` and returns `h(n)`.

```
public static int h(int n) {




}
```

Total marks for Section 2: **20**

General notes that apply to all questions in the following section, unless otherwise stated:

- `null` **references**: You may assume that for every method you write, none of the parameters it accepts can be a `null` reference; in other words, your methods do not have to handle cases in which their parameters are `null` references. Similarly, you may also assume that none of the arrays your methods take as parameters **contain** `null` references.
- **Using the methods or classes that you are asked to define**: when writing your solution to a question in this section, you may assume that the methods and classes in all other questions in this section have been successfully implemented, *even if you have not even attempted these other questions*.
- You are responsible for remembering to import classes as necessary.
- You may implement the specification below in almost any way you like. However, as in your assignments, you cannot add any **public** methods or member variables to your classes beyond the specification, but you may add **private** helper methods or member variables if you wish.
- Even if you are short on time, do not leave any questions blank. You will get points for things such as class headers, method headers, etc, if you write them.

Programming

Writing Montreal Maps

In the following questions, you will write classes that allow one to suggest a route to get from one point to another on a metro or train line. Note that for all of these questions, we will make the simplifying assumption that the earth is flat and that everything resides in 2 dimensions. (There were metros before Columbus and Galileo, right?)

[10]

6. Write a class `MetroStation` in which you define a new type `MetroStation`. (10 points)

   A `MetroStation` is defined to have the following private properties:

   - `String name` This represents the name of the metro station (e.g. Atwater, McGill, etc)
   - `double xPosition` This represents the x location of the metro station.
   - `double yPosition` This represents the y location of the metro station.

   An `MetroStation` object should have the following public methods/behaviors defined on it.

   - A constructor that takes as input a `String stationName`, a `double x`, and a `double y`. The constructor should initialize the properties `name`, `xPosition`, and `yPosition` to these three values respectively.
   - A method `getName()` which returns the name of the `MetroStation`.
   - A method `getXPosition()` which returns the x location of the `MetroStation`
   - A method `getYPosition()` which returns the y location of the `MetroStation`
   - A method `equals(MetroStation other)` which returns a `boolean` representing whether the `name` property of the `this MetroStation` is equal to the `name` property of `other`.

USE THIS PAGE IF YOU NEED ADDITIONAL SPACE.

[10]

7. Write a class `MetroLine` in which you define a new type `MetroLine` (10 points)

   A `MetroLine` is defined to have the following private properties:

   - A String representing the colour of the line (e.g. blue, red, green, etc. No numbers. It's Montreal, not Manhattan!)
   - A collection of `MetroStation` This collection can be either a built in array, an `ArrayList<MetroStation>`, or any other kind of collection you feel would satisfy the requirements of the class. The collection must be able to define an ordering from "beginning" to "end" of the `MetroLine`. It does not matter which part you consider the "beginning" and which part is the "end" as long as it is consistent. (We recommend using `ArrayList<MetroStation>` but you are free to do otherwise if you feel it will be easier.)

   A `MetroLine` should have the following methods defined:

   - A constructor which takes nothing as input and performs any initialization necessary to maintain your collection of stations.
   - A method `addStation` which takes as input a `MetroStation` and adds it to the end of your collection. The method can assume the station is not already part of the collection and should return void. Note that the order the `MetroStation`s are added is the same as the path the trains will travel on (or it could be the reverse path as well).
   - A method `hasStation` which takes as input a `MetroStation` and returns a `boolean` representing whether the metro line has a `MetroStation` with the same name or not.
   - A method `getLineColour` which returns the colour of the line.
   - A method `getFirstStop` which returns the `MetroStation` representing the first stop of the metro line. If there is no such station (i.e. if the collection of `MetroStation` is empty), your method should return null.
   - A method `getPriorStop` which takes as input a `MetroStation` and returns the stop prior to the `MetroStation` passed as input. You may assume that the `MetroStation` passed as input is indeed in the collection of `MetroStations` If the current station is the first stop on the `MetroLine` then this method should return null. You may assume that there are no "loops" in the `MetroLine` That is, no station is present on the same line twice.
   - A method `getNextStop` which takes as input a `MetroStation` and returns the stop after the `MetroStation` passed as input. You may assume that the `MetroStation` passed as input is indeed in the collection of `MetroStations`. If the current station is the last stop on the `MetroLine`, then this method should return null. You may assume that there are no "loops" in the `MetroLine` That is, no station is present on the same line twice.

   For the above methods, you may assume that there is no aliasing issue. That is to say, you can use == to compare the references, or if you are using a type such as `ArrayList` to maintain your list of stations, you can use the `indexOf` or `contains` method defined on an `ArrayList`.

USE THIS PAGE IF YOU NEED ADDITIONAL SPACE.

[25]

8. Write a class `SubwaySystem` in which you define a new type `SubwaySystem`. A `SubwaySystem` consists of a private `ArrayList<MetroLine> lines`. No methods other than the constructor and the `addLine` method should modify this array list. Your class should have the following non-static methods: (25 points)

   - Write a constructor `SubwaySystem` that takes nothing as input and initializes the private property `lines` to be an empty array list.
   - Write a method `addLine` which takes as input a `MetroLine newLine` and adds `newLine` to the array list `lines`. The method should return void.
   - Write a method `getDistanceTravelled` which takes as input an `ArrayList<MetroStation> path` and returns a `double` representing the total distance travelled along a path that goes from the first `MetroStation` in the arraylist, to the second `MetroStation` to the third, and so on, until the end of the array list. For each `MetroStation` along the journey, it should calculate the distance between the `MetroStation` and the prior one.

     If the `ArrayList<MetroStation> path` contains less than 2 values, your method should return 0.0. You may assume that the variable `path` is not null.

     For example, if the `ArrayList<MetroStation> path` contains five `MetroStation` you code should calculate the sum of the distance between the first and second stations, the second and third stations, the third and fourth, and lastly the fourth and final stations.

     Hint: To calculate the distance between two points, you can use a method defined in the appendix in the `MapUtilities` class.
   - Write a method `findNearestStation` which takes as input a `double x` and a `double y` and returns the `MetroStation` which is nearest to the `x` and `y`. To do this, it should look through all the `MetroStation`s of all the `MetroLine`s inside of `this.lines` and find the `MetroStation` that is the smallest distance away. You can use the same appendix method as in the previous question.

     You may assume that there is at least one `MetroLine` defined in the property `lines`. You may also assume that every `MetroLine` in `lines` has at least one `MetroStation` in it.

     Hint: There is no method such as `getAllStations` defined on a `MetroLine` object. This is since providing this method would reveal a lot about the implementation of the `MetroLine` class and would differ depending on the kind of collection you are using to store it. However, you can simulate this sort of method by using the `getFirstStop()` and `getNextStop()` methods to access every entry of the ArrayList. Remember the condition that will cause `getNextStop()` to be null.
   - A method `getNeighboringStations` which takes as input a `MetroStation station` and returns an `ArrayList<MetroStation>` The `ArrayList<MetroStation>` should consist of all neighboring stations to `station` (possibly many if the station is on many lines). For example, if a station is the 3rd stop on the blue line, the 6th stop on the red line, and the 1st stop on the green line, then the method should return an `ArrayList<MetroStation>` containing the `MetroStation` that is the 2nd stop on the blue line, the 4th stop on the blue line, the 5th stop on the red line, the 7th stop on the red line, and the 2nd stop on the green line.

     Remember, that a `MetroStation` will not necessarily be on every `MetroLine`. However, you may assume that the `MetroStation` is on at least one `MetroLine`.

   Finally, we are now going to write 2 methods that will help you to find a path from one `MetroStation` to another `MetroStation`. The first method will be accessible from outside the class and take as input the 2 `MetroStation` arguments you want. The second method will be a helper method that is private and will take as input a 3rd argument.

   - First, write a method `findPath` that takes as input a `MetroStation start` and a `MetroStation finish`. This method should simply call the `findPathHelper` method by passing to it as input `start`, `finish` and a new, empty `ArrayList<MetroStation>` (i.e. an array list with nothing in it). It should return the value returned by the method `findPathHelper`.
   - Write a *private* method `findPathHelper` that takes as input a `MetroStation start`, a `MetroStation finish`, an `ArrayList<MetroStation> partialPath` and returns an `ArrayList<MetroStation>` that contains a full path that goes from `start` until `finish`. If no such path exists without requiring "doubling back" (i.e. go from A to B and then back to A), then the method should return null.

     To do this, your method should do the following:

     – If `start` is contained in the `ArrayList<MetroStation> partialPath` passed in as input, then your method should return null. (Note that the `contains` method defined on an `ArrayList` will properly handle this as we defined the `equals()` method on a `MetroStation`.
     – If `start` and `finish` are the same based on the `equals()` method you defined on `MetroStation`s, then your method should return `partialPath`
     – If neither of the above are true, then you should do the following:
       (a) Compute an `ArrayList<MetroStation>` of possible places you can get to from `start`. (Call this $neighbors$. You can of course, use the method `getNeighboringStations` and assume it works correctly even if your code has a mistake in it.)
       (b) For each `MetroStation station` in $neighbors$ do the following:
         i. Create a duplicate copy of the `ArrayList<MetroStation> partialPath`. (See a method in the appendix for this.) Call this copy $duplicatePath$

ii. Add the station `start` to this duplicate copy at the end of the array list `duplicatePath`

iii. Calculate the path from the station `station` until `finish` by using recursion to call the method `findPathHelper`, but this time with input of `station`, `finish`, and the `duplicatePath`. Store this result into a variable `ArrayList<MetroStation> pathResult`.

iv. If the returned value is not null, then calculate the total distance travelled on this path using the method `getDistanceTravelled` that you wrote above.

Your method should return whichever `pathResult` had the smallest associated distance travelled (or null if all pathResults were null).

USE THIS PAGE IF YOU NEED ADDITIONAL SPACE. (Note that there is one more question after this!)

USE THIS PAGE IF YOU NEED ADDITIONAL SPACE. (Note that there is one more question after this!)

[10]

9. Finally, you should write a program `MontrealMaps` which will suggest a path to go along the metro to get from one place to another. The only required method for `MontrealMaps` is a main method but of course you are allowed to write others.

   Your program should do the following. For any library methods that you use, make sure that you are properly importing things and handling any exceptions they may throw. (10 points)

   - Using an appropriate library method, you should create a `SubwaySystem` object. If this method fails to initialize, you should print the message `Error, not enough funding.`
   - Use a `Scanner` and print statements to read the current x and y position of the user.
   - Use a `Scanner` and print statements to read the x and y position that the user wants to go to.
   - Find the nearest MetroStation to both the start position of the user and his final destination.
   - If the 2 metro stations have the same name (i.e. are the same metro station) then print `It is fastest to walk there.`
   - Otherwise, call the method `findPath` from the previous part to find a path from these 2 metro stations.
   - If the method returns null, then print `Sorry, there is no path on the metro, you'll have to walk`
   - Otherwise, print the path using the metro using the method `printPath` defined in the `MapUtilities` class in the appendix.

USE THIS PAGE IF YOU NEED ADDITIONAL SPACE.

USE THIS PAGE IF YOU NEED ADDITIONAL SPACE.

USE THIS PAGE IF YOU NEED ADDITIONAL SPACE. CLEARLY INDICATE WHICH QUESTION(S) YOU ARE ANSWERING HERE.

USE THIS PAGE IF YOU NEED MORE ADDITIONAL SPACE. CLEARLY INDICATE WHICH QUESTION(S) YOU ARE ANSWERING HERE.

USE THIS PAGE IF YOU NEED EVEN MORE ADDITIONAL SPACE. CLEARLY INDICATE WHICH QUESTION(S) YOU ARE ANSWERING HERE.

SUMMARY OF JAVA STANDARD LIBRARY METHODS FOR SELECTED CLASSES

- `String` (package `java.lang`) Methods:
  - `public boolean equals(Object anObject)`: Compares this `String` to `anObject`.
  - `public boolean equalsIgnoreCase(String anotherString)`: Compares, ignoring case considerations, this `String` to `anotherString`.
  - `public int compareTo(String anotherString)`: Compares this `String` to `anotherString` lexicographically; returns a negative value if this `String` occurs before `anotherString`, a positive value if this `String` occurs after `anotherString`, and `0` if both `String`s are equal.
  - `public int compareToIgnoreCase(String anotherString)`: Compares, ignoring case considerations, this `String` to `anotherString` lexicographically; returns a negative value if this `String` occurs before `anotherString`, a positive value if this `String` occurs after `anotherString`, and `0` if both `String`s are equal.
  - `public char[] toCharArray()`: Converts this `String` to a new character array.
- `File` (package `java.io`) Methods:
  - `public FileSstring pathname)`: Creates a new `File` instance that corresponds to the given pathname.
- `Scanner` (package `java.util`) Methods:
  - `public Scanner(Inputstream source)`: Constructs a new `Scanner` that produces values scanned from the specified input stream.
  - `public Scanner(File f)`: Constructs a new `Scanner` that produces values scanned from the specified File
  - `public double nextDouble()`: Scans the next token of the input as a `double`.
  - `public boolean nextBoolean()`: Scans the next token of the input as a `boolean`.
  - `public int nextInt()`: Scans the next token of the input as an `int`.
  - `public String nextLine()`: Advances this `Scanner` past the current line and returns the input read.
  - `public boolean hasNextLine()`: Checks whether there are further lines left to scan.
- `PrintStream` (package `java.io`) Methods:
  - `public void print(boolean b)`: Prints `boolean` value `b`.
  - `public void print(double d)`: Prints `double` value `d`.
  - `public void print(int i)`: Prints `int` value `i`.
  - `public void print(Object o)`: Prints `Object` `o`.
  - `public void print(String s)`: Prints `String` `s`.
  - `public void println()`: Terminates the current line by writing the line separator string.
  - `public void println(boolean b)`: Prints `boolean` value `b` and then terminates the line.
  - `public void println(double d)`: Prints `double` value `d` and then terminates the line.
  - `public void println(int i)`: Prints `int` value `i` and then terminates the line.
  - `public void println(Object o)`: Prints `Object` `o` and then terminates the line.
  - `public void println(String s)`: Prints `String` `s` and then terminates the line.
- `ArrayList` (package `java.util`) Methods:
  - `public boolean ArrayList<type>()`: Creates a new `ArrayList<type>`
  - `public void add(type t)` : Appends the specified element to the end of this list.
  - `public void add(int index, type t)`: Inserts the specified element at the specified position in this list. item `public void addAll(Collection<type> c)`: Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's Iterator.
  - `public boolean contains(Object o)`: Returns true if this list contains the specified element.
  - `public type get(int index)`: Returns the element at the specified position in this list.
  - `public int indexOf(Object o)`: Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element. It searches for the object by calling the .equals() method defined on the Object.
  - `public boolean remove(Object o)`: Removes the first occurrence of the specified element from this list, if it is present.
  - `public int size()`: Returns the number of elements in this list.
- `Math` (package `java.lang`) Methods:
  - `public static double pow(double a, double b)`: Returns the value of `a` raised to the power of `b`.
  - `public static double sqrt(double a)`: Returns the correctly rounded positive square root of `double` value `a`.
  - `public static double random()`: Returns a `double` value with a positive sign, greater than or equal to `0.0` and less than `1.0`.
  - `public static double exp(double a)`: Returns Euler's number e raised to the power of `double` value `a`. (base e) of `double` value `a`. of `double` value `a`.

DESCRIPTION OF METHODS ESPECIALLY PROVIDED FOR THE PROGRAMMING SECTION OF THE EXAM

- `MapUtility` Methods:
  - `public static double distanceBetweenPoints(double x1, double y1, double x2, double y2)`
    Input: 4 doubles representing 2 points
    Output: The distance between the 2 points using pythagorean theorem
  - `public static ArrayList<MetroStation> duplicate(ArrayList<MetroStation> stations)`
    Input: An array list of MetroStations to duplicate
    Output: Returns a duplicate copy of the arraylist via a shallow copy. That is, the individual `MetroStations` in each different `ArrayList` will still be the same, but there will be a copy of the `ArrayList` itself.
  - `public static SubwaySystem generate() throws IOException`
    Input: a none
    Output: a constructor that generates a SubwaySystem Object. Throws an NotEnoughFundsException when it fails to generate it.
  - `public static void printPath(ArrayList<MetroStation> path)`
    Input: An `ArrayList<MetroStation>` representing a path from the first station until the end
    Output: None
    Side effect: Prints to the screen the path.