Comprehensive Lab 3

Released on: Wednesday, November 21st
Due on: Wednesday December 5th, 11:59pm
How: submit zip file to Google classroom
What: A zip file named after you (LastNameFirstName-comp-lab03.zip) containing 2 Java files.

Lab Description

Objective 1

Here is **your first mission**. We are presenting to you a little game below: the Factor Maze Puzzle.

Factor Maze Puzzle

Instructions: Start at the top left hand corner, and solve the maze by moving down to the bottom right. If the number you are on is divisible by 2, you can go left. If the number is divisible by 5, you can go right. If the number is divisible by 3, you can go up. And if the number is divisible by 7, you can go down. Only go from one number to the next (never skipping over numbers). You cannot go back on your steps (a cell can only be visited once).

START
↓

225	200	70	14	35	14	35	14	25	14
35	40	6	15	28	15	18	15	20	42
75	20	14	35	40	50	10	14	5	42
35	14	21	21	35	200	2	105	14	21
21	15	12	225	49	35	14	9	21	9
115	50	20	40	77	9	15	14	225	14
28	35	200	14	35	20	50	14	35	6
21	3	35	42	49	20	20	14	105	14
15	50	28	21	5	70	14	21	9	21
5	40	205	300	40	25	15	45	20	42

↓
FINISH

Your job on this mission is to figure out the way through the above table, while following the rules of the maze. As you are going through the above table, you have to draw another table of the same size, with all cells empty, and draw and X in each cell that you visit. As a result, when you are done, we can follow in your "footsteps". Draw this new table below.

Objective 2

Here is **your second mission**.
Now that you have played with the Factor Maze Puzzle (a.k.a. FMP), we are now expecting you to write some code to automate it.

We are providing you a java file: mission2.java, that contains starter code for you to fill in.
There are two methods that you have to complete: buildPath and printPath.

buildPath:

- takes a 2D array of positive integers; and
- returns a 2D array of strings in which you will write the correct path: the correct path is defined by the FMP rules (see mission 1 document to remember the rules).

printPath:

- takes a 2D array of strings that represents the path you need to take; and
- does not return anything: it only prints out this table.

The main method is provided to you. Once you have finished implementing buildPath and printPath, executing mission2 with the table below should result in the following terminal output:

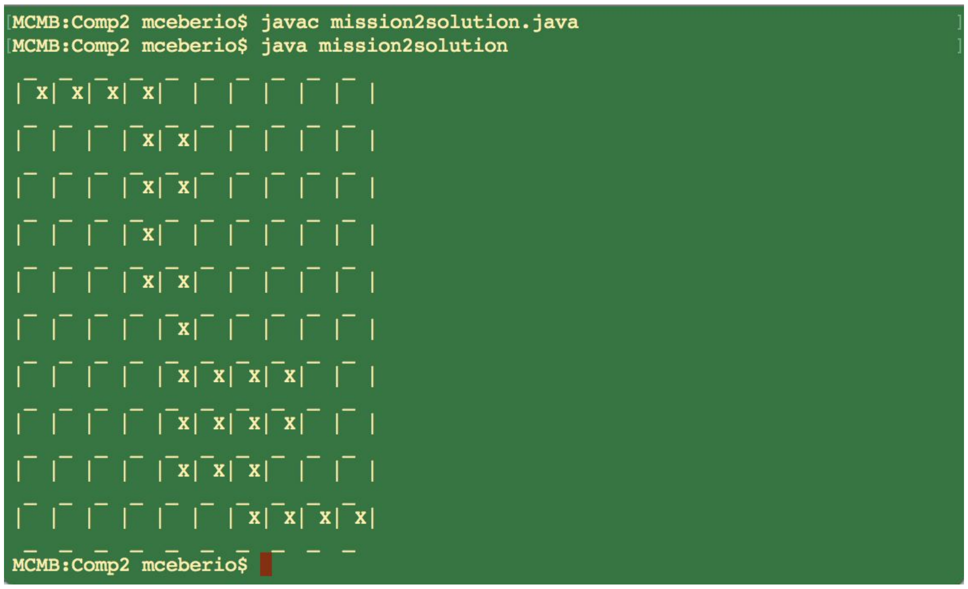
INPUT TABLE:

START
↓

225	200	70	14	35	14	35	14	25	14
35	40	6	15	28	15	18	15	20	42
75	20	14	35	40	50	10	14	5	42
35	14	21	21	35	200	2	105	14	21
21	15	12	225	49	35	14	9	21	9
115	50	20	40	77	9	15	14	225	14
28	35	200	14	35	20	50	14	35	6
21	3	35	42	49	20	20	14	105	14
15	50	28	21	5	70	14	21	9	21
5	40	205	300	40	25	15	45	20	42

↓
FINISH

EXPECTED OUTPUT:



```
MCMB:Comp2 mceberio$ javac mission2solution.java
MCMB:Comp2 mceberio$ java mission2solution

|¯x|¯x|¯x|¯x|¯ |¯ |¯ |¯ |¯ |¯ |

|¯ |¯ |¯ |¯x|¯x|¯ |¯ |¯ |¯ |¯ |

|¯ |¯ |¯ |¯x|¯x|¯ |¯ |¯ |¯ |¯ |

|¯ |¯ |¯ |¯x|¯ |¯ |¯ |¯ |¯ |¯ |

|¯ |¯ |¯ |¯x|¯x|¯ |¯ |¯ |¯ |¯ |

|¯ |¯ |¯ |¯ |¯x|¯ |¯ |¯ |¯ |¯ |

|¯ |¯ |¯ |¯ |¯x|¯x|¯x|¯x|¯ |¯ |

|¯ |¯ |¯ |¯ |¯x|¯x|¯x|¯x|¯ |¯ |

|¯ |¯ |¯ |¯ |¯x|¯x|¯x|¯ |¯ |¯ |

|¯ |¯ |¯ |¯ |¯ |¯ |¯x|¯x|¯x|¯x|
 ¯  ¯  ¯  ¯  ¯  ¯  ¯  ¯  ¯  ¯
MCMB:Comp2 mceberio$
```

Objective 3

Here is **your third mission**.
In Missions 1 and 2, you have built some experience with Factor Maze Puzzles. With this 3rd mission, you are going to integrate the code you wrote for Mission 2 into a new type called FMP.

This new type, FMP, allows you to bundle together the following pieces of data (attributes):

- A puzzle: the table that contains positive integers;
- A path: a table of strings that contains the path from start to finish, if it exists; and
- A Boolean value that informs of whether there is a path between start and finish.

In this mission, you have to put together the code of this new type FMP.

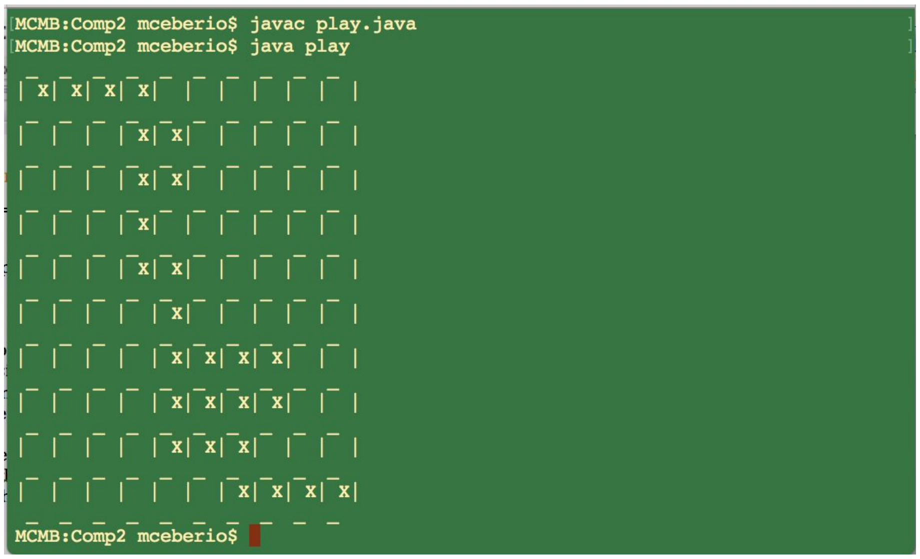
We are providing you a java file: FMP.java, that contains starter code for you to fill in. We are also giving you the file called play.java that makes use of your new type.

Completing file FMP.java:
The attributes of this new type are given to you.
Constructor: implement the default constructor along with another one that takes a table of integers as its unique parameter.
Note: the constructor that takes a table as input should initialize the attribute puzzle with this table, but also initialize the table of Strings path to a 2D array of the same size as puzzle, filled with the
Setters: only one – a setter for the table of integers (attribute called puzzle).
Getters: one per attribute.

Aside from the usual constructors, getters, and setters, we are asking you to design and implement the following additional methods:

- buildPath
- printPath

These two methods use the code you wrote in Mission 2. Just make sure that your implementation of FMP.java allows to execute play.java as follows:



```
MCMB:Comp2 mceberio$ javac play.java
MCMB:Comp2 mceberio$ java play

|¯x|¯x|¯x|¯x|¯ |¯ |¯ |¯ |¯ |¯ |

|¯ |¯ |¯ |¯x|¯x|¯ |¯ |¯ |¯ |¯ |

|¯ |¯ |¯ |¯x|¯x|¯ |¯ |¯ |¯ |¯ |

|¯ |¯ |¯ |¯x|¯ |¯ |¯ |¯ |¯ |¯ |

|¯ |¯ |¯ |¯x|¯x|¯ |¯ |¯ |¯ |¯ |

|¯ |¯ |¯ |¯ |¯x|¯ |¯ |¯ |¯ |¯ |

|¯ |¯ |¯ |¯ |¯x|¯x|¯x|¯x|¯ |¯ |

|¯ |¯ |¯ |¯ |¯x|¯x|¯x|¯x|¯ |¯ |

|¯ |¯ |¯ |¯ |¯x|¯x|¯x|¯ |¯ |¯ |

|¯ |¯ |¯ |¯ |¯ |¯ |¯x|¯x|¯x|¯x|
 ¯  ¯  ¯  ¯  ¯  ¯  ¯  ¯  ¯  ¯
MCMB:Comp2 mceberio$ 
```

Objective 4

Let's not forget: as your last mission for this lab, it is a BONUS POINTS activity!

Here is **your fourth (and last) mission**.
In this last mission, you are asked to revisit the method buildPath from your file FMP.java: you are now going to implement a method that does the same thing as buildPath, but recursively. You will call this method buildPathR.

Here is how it goes.
When you are building a path through a table, you are building it starting at the top left corner, make one move, and then building again a path with the same rules, but from a different location on the 2D array.
So roughly speaking,

Building a path through a table from the top left corner
=
Building a path through the same table but from another cell
(the one you reach in one step from the top left corner)

In other words:
Building a path through a table from a given location =

- We are done if the given location is the bottom right corner (base case)
- We are done if there is no available move (base case)
- Otherwise: we take the move instructed by the puzzle's rules AND THEN we build a path through the same table from the new location (closer to reaching the end of the path or to being stuck) (recursive call)

In java, it will go as follows:

```
public void buildPathR() {
        path[0][0]="X";
        this.buildPathAux(0,0);
}

public void buildPathAux(int i, int j) {
         // -- Base cases --
        // this is where your code goes
        // -- Recursive calls --
        // and here too
}
```

What do you have to turn in?

3 Java files, **play.java, mission2.java, and FMP.java** and word doc for mission 1 activity all in a zip folder.

Criteria for grading:

- **[10 pts] Well commented code explaining what you are doing (clear and grammatically correct) along with correct formatting (properly tabbed code).**
- **[5 pts] Objective 1**
- **[45 pts] Objective 2**
- **[40 pts] Objective 3**
- **[bonus: 20 pts] Objective 4**