WIX1002 Fundamentals of Programming
Lab Report 3

1. MapleStory is an awesome side-scrolling MMORPG where fascinating lore and epic adventures unravel. You will play the role of the legendary warrior **Aran**, who courageously sealed the almighty Black Mage, a powerful being possessing a different ideology with us so we must unreasonably define him as the villain.

   The **Aran** class will have such *properties and* behaviours:

   - An empty constructor initializing **Aran's**:
     - *level* as 300 [-1 < *level* < 301]
     - *jobAdvancement* as -1 [-1 ≤ *jobAdvancement* < 6]
     - private static integer [6] *jobAdvMap* specifying which *level* should **Aran** be initialized according to his *jobAdvancement*

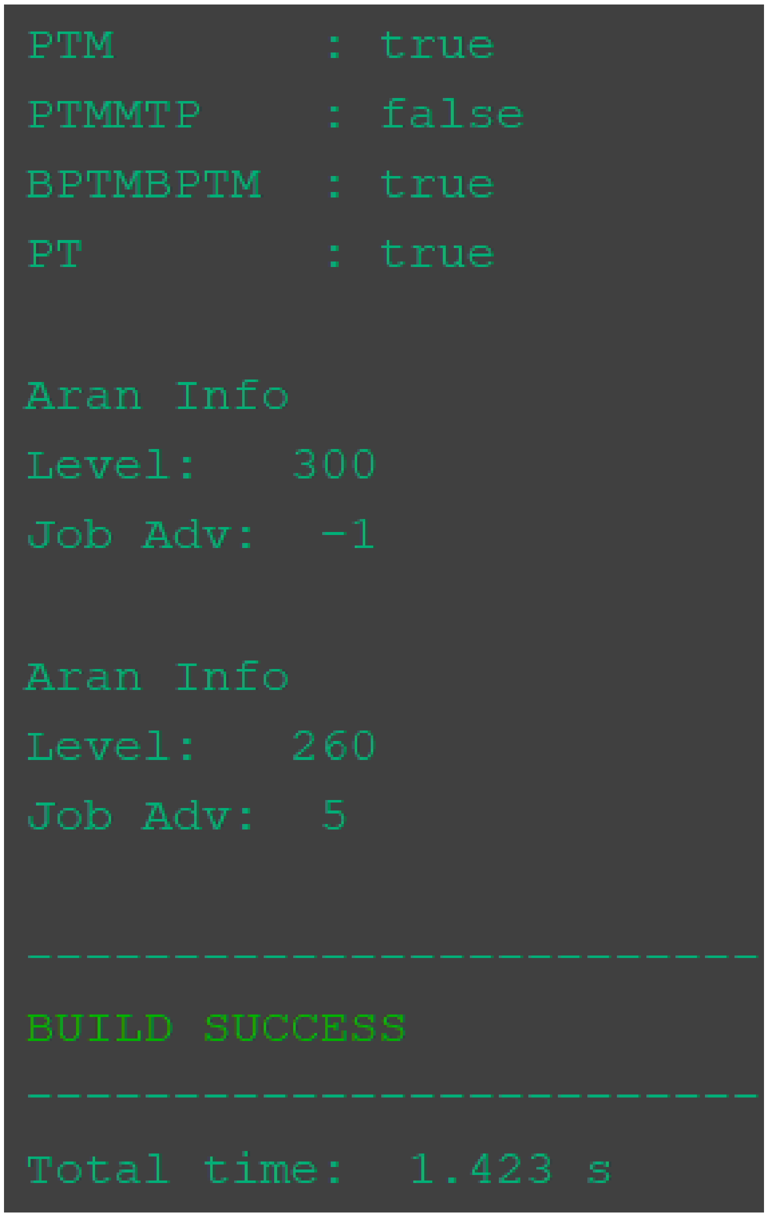
   - A constructor accepting an integer parameter for *jobAdvancement* and allocate an appropriate *level* according to *jobAdvMap*.

   - The *level* accessor and mutator method where *jobAdvancement* should be adjusted accordingly too, if necessary, and restricts their magnitude according to the assigned ranges.

   - **Aran** has gotten her head frozen for a century so she forgot how to properly perform her awesome combos. Program a static boolean isValid(String input) method to return true for such conditions:
     - Input is a string only consisting of characters 'B', 'P', 'T' and 'M'
     - 'P' can only be followed by 'T' and 'T' can only be followed by 'M'
     - 'B' may or may not exists only before 'P'

   - A toString method that returns all information as demonstrated in the sample output.

Sample output



```
PTM         : true
PTMMTP      : false
BPTMBPTM    : true
PT          : true

Aran Info
Level:    300
Job Adv:   -1

Aran Info
Level:    260
Job Adv:   5


-------------------------
BUILD SUCCESS
-------------------------
Total time:  1.423 s
```

2. In the immersive world of Pokemon, where battles and strategies abound, a critical element comes into play—type effectiveness. Pokemon are categorized into diverse types, each with its unique strengths and weaknesses. This diversity adds layers of complexity to battles, as certain types are more effective against others, with strategic advantages that can turn the tide of a battle.

   Note that the type of Pokemon will buff the strength of Pokemon while combating. For example, when a Flame-type Pokemon engages in combat with a Grass-type Pokemon, the system considers the elemental dynamics. In this scenario, the strength of the Flame-type Pokemon is enhanced, as its strength is multiplied by a factor, typically 1.5.

Flame : Grass	Grass : Water	Water : Flame
1.5 : 1.0	1.25 : 1.0	1.4 : 1.0

   You should create a class `Pokemon` to store information about each Pokemon, including their name, type and strength. Include the necessary constructor, accessor, and mutator methods for each variable, if required.

   You are required to create another class `PokemonSortingSystem`, which is a vital component of our Pokemon management system, designed to efficiently organize and evaluate Pokemon based on their combat strength and provide valuable insights to trainers and researchers. First, sort the Pokemon based on their strength in ascending order. Then, a method, `determineWinner(String name)`, has to be created to offer trainers an invaluable resource by assessing type effectiveness and returning an array of Pokemon names that can triumph over a specified opponent. Type effectiveness buffs, intricately designed, will be considered in determining the winner of combat.

Test Program

```
public class PokemonTest {
     public static void main(String[] args) {
             Pokemon moltres = new Pokemon("Moltres",
     "Flame", 85.0);
             Pokemon servine = new Pokemon("Servine",
     "Grass", 60.0);
             Pokemon charmander = new Pokemon("Charmander",
     "Flame", 92.0);
             Pokemon pansage = new Pokemon("Pansage",
     "Grass", 55.0);
             Pokemon araquanid = new Pokemon("Araquanid",
     "Water", 74.0);
             Pokemon flareon = new Pokemon("Flareon",
     "Flame", 65.0);
             Pokemon squirtle = new Pokemon("Squirtle",
     "Water", 63.0);
             Pokemon wooper = new Pokemon("Wooper",
     "Water", 42.0);

             Pokemon[] PokemonList = {moltres, servine,
     charmander, pansage, araquanid, flareon, squirtle,
     wooper};

             PokemonSortingSystem sortingSystem = new
     PokemonSortingSystem(PokemonList);

             System.out.println("List of Pokemon after
     Sorting: ");
             for (Pokemon pokemon : PokemonList) {
                 System.out.println(pokemon);
             }

             String opponentName = "Squirtle";
             String[] winners =
     sortingSystem.determineWinner(opponentName,
     PokemonList);

             System.out.print("Pokemon effective against "
     + opponentName + ":");

             for (String winner : winners) {
                 System.out.print(winner+", ");
             }
```

```
            }
        }
```

Sample Output

```
List of Pokemon afterSorting:
Wooper
Pansage
Servine
Squirtle
Flareon
Araquanid
Moltres
Charmander

Pokemon effective against Squirtle: Charmander Araquanid
Servine Pansage
```

Lab Report

Prepare a report to solve the above problems. The report should contain all the sections as below for each question:

No	Section	Description
1	Problem	Description on the problem
2	Solution	Explanation on how to solve the above problems
3	Sample Input & Output	A few sets of input and output (snapshot)
4	Source Code	Java Source Code

Requirements

1. Group Assignment (4-5 students per group)
2. Cover page that includes all student matric number and full name.
3. Font: Times New Roman 12, Line Spacing: 1 ½ Spacing
4. Submit to Spectrum according to your OCC.