HW 05: Pokemon Battles

Release date: Friday, September 23th, 2016
Due date: Friday, October 7th, 2016 – 11:59 pm

This is a 2-week homework and it will count as two homeworks for your final grade.

Goals

- Learn to model a problem using Java classes
- Learn to use selection and repetition to guide the control flow of a program.

Description

In addition to labs, CS180 now has a Pokemon Gym where students can compete against each other. As a participant yourself, you are to write a program that helps predict which Pokemon wins a given battle.

To perform this task, you should develop a class called "Pokemon". Each Pokemon entity is modelled as an Object of this class. A Pokemon has a number of attributes that are represented as data members of the Object that represents this Pokemon. These include the Pokemon's name, its ID, type, base attack power and health power. The Pokemon class has a class method that simulates a battle between two Pokemon Objects and returns the winner.

You are also required to develop a class called "MyUtils" that you will use for the rest of the semester. This class has some utility methods for manipulating Strings.

Details on the Pokemon class and MyUtils class are given in the rest of this document.

I- The Pokemon Class

You are required to develop a public class called "Pokemon". The data members of Objects of this Class are shown in Table 1.

1. Initialization

At the time of the creation of a *new* Pokemon, the values of all its data members should be initialized using the constructor arguments except for the ID. The Pokemon has a static integer that counts the number of Pokemon Objects created so far (see Table 2). The ID of a new Pokemon is set to the value of this counter at the time of the Object creation. The prototype of the constructor is, thus:

```
public Pokemon(String name, String type, int healthPower,
                       double baseAttackPower)
```

The type of the Pokemon can be either Fire, Water, Grass or Electric. The health power of the Pokemon should be either zero or a positive integer value. The base attack power of the Pokemon is a positive value. If the value given in the constructor arguments is invalid for the corresponding attribute, the constructor sets this attribute to its default value. You may assume that the name provided will not be an empty String. Your comparison of the type of the Pokemon given as an argument of the constructor should be case-insensitive; that is if the given String is "ELeCtrIc", it should be accepted. However, you need to save the type in the proper format in which the first character of the type is uppercase and the rest of the characters are lowercase.

2. Accessors and Mutators

Since all data members have private access modifiers, you should supply accessor methods for each data member. The prototypes of these methods are shown in Table 3.

The values of all data members of the Pokemon Object can be changed at any point of time except the name and ID; you are required to supply mutators for the other data members. Table 4 contains the prototypes of the required mutators.

3. Object methods

You are required to implement the Object methods shown in Table 5 in the Pokemon Class. The `toString` method is used to print the values of the Pokemon Object data members. Table 9 shows the result of printing the return value of the `toString` method.

The `isDead` method checks if the healthpower of the Pokemon is greater than zero, in this case `isDead` returns the value false; otherwise, it returns true. The `boostHealthPower` method increases the health power of the Pokemon by the amount given as an argument. The `reduceHealthPower` method reduces the healthpower of the Pokemon by the given amount. Note that the healthpower of the Pokemon should not go below zero. Therefore, if the amount given in `reduceHealthPower` will make the healthpower of the Pokemon negative, the value of the healthpower should be set to zero instead.

4. Class methods

You should implement a class method in the Pokemon class that simulates the battle between two Pokemons and returns information about the winner. The prototype of the method is:

```
public static int battle(Pokemon p1, Pokemon p2)
```

The `battle` method receives reference to two Pokemon Objects in its arguments. This method returns the value 1 if the first Pokemon wins the battle and returns the value 2 if the second Pokemon wins.

The battle between two Pokemons consists of rounds. In each round, each Pokemon punches the other; this punch results in a decrease in the healthpower of the latter by an amount equal to the base attack of the puncher multiplied by a factor that represents the effect of the punch on the other Pokemon. This factor is computed based on the two Pokemon types. Table 7 shows the values of this factor for the different Pokemon types. The two Pokemons battle until the health power of one of them is zero; the other Pokemon will be considered the winner in this case.

In order to find the attack factor used in the `battle` method, you should implement a method called `getAttackMultiplier.` The prototype of this method is:

```
public static double getAttackMultiplier(Pokemon attacker, Pokemon
                                defender)
```

This method encodes the information in Table 7. Based on the type of the attacker and the defender, it returns the value of the effect of a punch from the attacking Pokemon on the defending Pokemon.

You should also implement a method called “`battleOracle`”. This method finds the winning Pokemon if, **hypothetically**, two Pokemons entered a battle. This method does not have any side-effect on the health power of any of the two Pokemons. The prototype of the `battleOracle` method is:

```
public static int battleOracle(Pokemon p1, Pokemon p2)
```

Table 6 contains all the class methods that should be present in the Pokemon class. A description on each method is provided in the same table.

II- MyUtils Class

You are required to implement a class called “`MyUtils`”. This class will contain utility methods that you will use throughout the semester. `MyUtils` contains a method called `isNumeric`. The prototype of this method is:

```
public static boolean isNumeric(String str)
```

Given an input String, this method checks if this String represents a number. This number could be an integer or a floating point number. `isNumeric` returns a boolean value that is true if the String is numeric, and false otherwise. An empty String is a not considered a number; therefore, if `isNumeric` method is called on an empty String, the return value should be false.

To check if the given String is a number, you should loop on all the characters of this String and check if the character is either a number or a period. If there is a period in the String, it should be followed by a number.

In the `MyUtils` class, you should also provide a method that takes a String, and converts all its characters to lowercase except the first character that should be uppercase. The prototype of this method is:

```
public static String formatStr(String str)
```

Table 8 contains all the methods that should be implemented in the `MyUtils` class.

III- The main method

You are required to implement a main method that reads from the user the data of two Pokemons and shows the winner of the battle between them. You will need to perform input validation making use of the MyUtils class *(i.e. the input for the health and attack needs to be a number)*. Table 10 contains a demo of a correct program. Declare the Scanner or JOptionPane object in the main method, **do not make them instance variables.**

Object Member Definition	**Default Value**	**Description**
`private String name`	--	Contains the name of the Pokemon
`private int ID`	--	Represents the ID of the Pokemon
`private String type`	"Fire"	Represents the Pokemon type which would be either Fire, Water, Electric or Grass
`private int healthPower`	0	Represents the health power of the Pokemon
`private double baseAttackPower`	1	Represents the value by which the Pokemon can reduce another Pokemon's health in 1 hit

Table 1: Pokemon Object data members

Class Members Definition	**Description**
private static int NUM_POKEMONS = 0	Contains the number of Pokemon Objects created so far. Used as the id of a newly created Pokemon

Table 2: Pokemon Class members

Accessor Method Prototype	**Description**
`public String getName()`	Returns the name of the Pokemon
`public int getId()`	Returns the ID of the Pokemon
`public String getType()`	Returns the type of the Pokemon
`public int getHealthPower()`	Returns the current healthpower of the Pokemon
`public double getBaseAttackPower()`	Returns the attack power of the Pokemon

Table 3: Accessors in the Pokemon Class

Mutator Method Prototype	**Description**
`public boolean setType(String type)`	Sets the type of this Pokemon. The type provided should be either Fire, Water, Grass, or Electric. If the type provided is incorrect, the method returns false and no changes are made to the Object members data.
`public boolean setHealthPower(int healthPower)`	Sets the health power of this Pokemon to the given value. If the value provided in the method arguments is not a positive numeric value or zero, this method returns false; otherwise, this method returns true.
`Public boolean setBaseAttackPower(double baseAttackPower)`	Sets the base attack power of this Pokemon to the given value. If the value provided in the method arguments is not a positive numeric value, this method returns false; otherwise, this method returns true.

Table 4: Mutators in the Pokemon Class

Method Prototype	**Description**
`public String toString()`	Composes a String Object that contains the data stored in the members of the Pokemon Object.
`public boolean isDead()`	Returns true if this Pokemon’s health power is equal to zero. Returns false otherwise.

`public void boostHealthPower(int healthPower)`	Increases the value of the healthpower of the Pokemon by the value given in the method's argument.
`public void reduceHealthPower(int healthPower)`	Decreases the healthpower of the Pokemon by the value given in the method's argument. If the value given is greater than the value of the healthpower of the Pokemon, the healthpower of the Pokemon is set to zero.

Table 5: Object methods in the Pokemon class

Method Prototype	**Description**
`public static int battle(Pokemon p1, Pokemon p2)`	Returns 1 if the first Pokemon is the winner and 2 if the second Pokemon is the winner. This method has a side effect on the healthPower of each Pokemon.
`public static double getAttackMultiplier(Pokemon attacker, Pokemon defender)`	Returns the factor by which a punch, represented by the baseAttackPower, from the attacker Pokemon will have an effect on the healthpower of the defender Pokemon.
`public static int battleOracle(Pokemon p1, Pokemon p2)`	Returns the result of a battle between p1 and p2. This method is different from the `battle` method in that it does not have a side effect on the Object members of any of the Pokemons. It is rather used to find the winner in a hypothetical battle between the two Pokemons.

Table 6: Class methods in the Pokemon class

Attacker Type	**Defender Type**	**Factor**
Grass	Grass	½
Grass	Electric	1
Grass	Water	2
Grass	Fire	½
Electric	Electric	½
Electric	Water	2
Electric	Fire	1
Electric	Grass	½

Water	Water	½
Water	Electric	1
Water	Fire	2
Water	Grass	½
Fire	Fire	½
Fire	Electric	1
Fire	Water	½
Fire	Grass	2

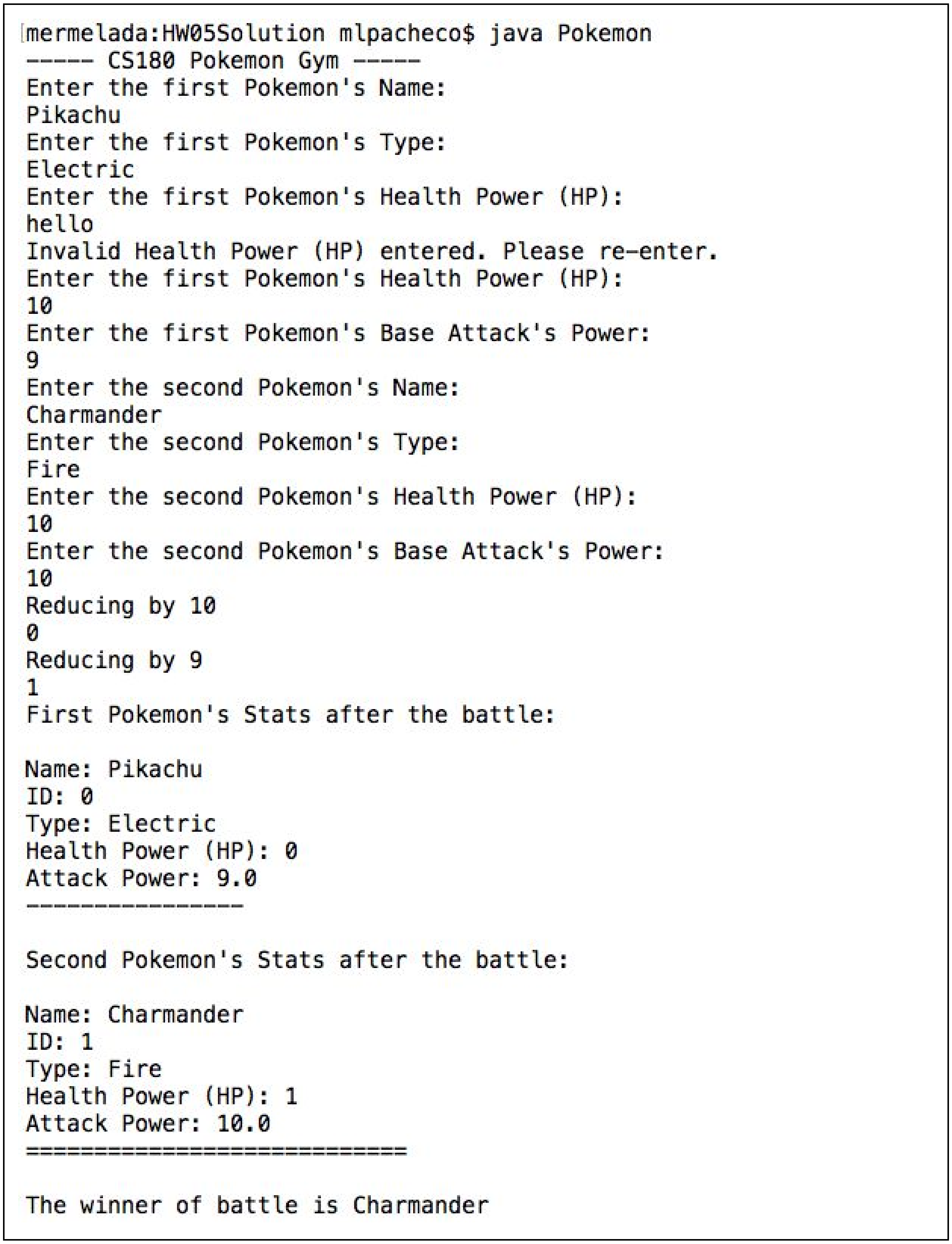
Table 7: Attack Factor

Method Prototype	**Description**
`public static boolean isNumeric(String str)`	Returns true if the String given as an argument is a number. This String could contain an integer or a floating point number. Returns false otherwise.
`public static String formatStr(String str)`	Creates a new String Object. This String contains the same characters as the String given as an argument. The method converts these characters to lowercase except the first character which should be in uppercase. The result of calling formatStr(“fIrE”) is the String “Fire”

Table 8: Methods in MyUtils class

```
Name: Pikachu
ID: 101
Type: Electric
Health power: 1
Base attack power: 10
```

Table 9: Result of printing the result of calling the toString method on a Pokemon Object



```
mermelada:HW05Solution mlpacheco$ java Pokemon
----- CS180 Pokemon Gym -----
Enter the first Pokemon's Name:
Pikachu
Enter the first Pokemon's Type:
Electric
Enter the first Pokemon's Health Power (HP):
hello
Invalid Health Power (HP) entered. Please re-enter.
Enter the first Pokemon's Health Power (HP):
10
Enter the first Pokemon's Base Attack's Power:
9
Enter the second Pokemon's Name:
Charmander
Enter the second Pokemon's Type:
Fire
Enter the second Pokemon's Health Power (HP):
10
Enter the second Pokemon's Base Attack's Power:
10
Reducing by 10
0
Reducing by 9
1
First Pokemon's Stats after the battle:

Name: Pikachu
ID: 0
Type: Electric
Health Power (HP): 0
Attack Power: 9.0
----------------

Second Pokemon's Stats after the battle:

Name: Charmander
ID: 1
Type: Fire
Health Power (HP): 1
Attack Power: 10.0
============================

The winner of battle is Charmander
```

Table 10: Demo of a correct program

Rubric

This homework is worth 100 points. The Pokemon class is worth 80 points and the MyUtils class is worth 20 points. The points breakdown is as follows.

- Pokemon class (80%)
    - Constructor (10%)
    - Accessors (10%)
    - Mutators (10%)
    - Object methods (25%)
    - Class methods (25%)
- MyUtils class (20%)
    - `isNumeric` (15%)
    - `formatStr` (5%)

Submission Instructions

Submit all your files to Vocareum by following these instructions. Keep in mind that only your last submission will be considered.

Good Luck