Introductory Java Programming

Queen Mary University of London
School of Electronic Engineering and Computer Science

Course Code: EBU4201

Lab Sheet 5: Inheritance and Abstract Classes

---

1. The sub-questions below are about a Java application that deals with *monsters*.

   i) Create a class called **`Monster`** (to be stored in file **`Monster.java`**), such that:

   - every monster has a **`name`** *and*,
   - every monster can attack and move; i.e., the class **`Monster`** has two methods, **`attack()`** and **`move()`**.

   A generic monster **`attack()`** method should return a random integer value between **1** and **5**. It should also print the following message to the command line:

   `"NAME, of type CLASS_TYPE, attacks generically: X points damage caused."`

   where:

`NAME`	= monster name
`CLASS_TYPE`	= class (i.e., type) of monster[1]
`X`	= random integer in the specified range

   The monsters' generic **`move()`** method should be defined as follows:

```
public void move(int direction) {
  switch(direction) {
    case 1:
      System.out.println(this.name + "is moving 1 step NORTH.");
      break;
    case 2:
      System.out.println(this.name + "is moving 1 step EAST.");
      break;
    case 3:
      System.out.println(this.name + "is moving 1 step SOUTH.");
      break;
    default:
      System.out.println(this.name + "is moving 1 step WEST.");
      break;
  }
}
```

[1] **Hint**: This can be accessed via the method call **`this.getClass();`**.

ii) A **`Dragon`** is a type of **`Monster`** that attacks by either breathing fire or scratching (this is a generic attack). It attacks by breathing fire **30%** of the time and generically the rest of the time[2]. When it attacks by breathing fire, it causes between **1** and **50** points of damage; in this case, it also prints a statement to the console, like the generic attack, that includes the *name*, the *class type*, the *attack type* and the *damage done*.

Name this program **`Dragon.java`**.

iii) A **`Troll`** is also a kind of **`Monster`**, but trolls cannot be named **Saul** or **Salomon**. If the user of your program attempts to give a wrong name to a troll, your program should print an error message and name the troll **Detritus**.

Name this program **`Troll.java`**.

iv) Your Java application should run with the class **`TestingMonsters`**, which should be stored in the **`TestingMonsters.java`** file provided in QMplus.

2. For this question, you will be using the three Java classes you wrote for *Q1* (i.e., **`Monster`**, **`Dragon`** and **`Troll`**), together with the **`TestingMonsters`** class that was provided to you in QMplus. In addition, we will also be assuming that there is no longer such a thing as a *generic* monster. Make the **`Monster`** class **`abstract`** and:

   i) All monsters should still have a **`name`**; however, they should also have an instance variable called **`spAttackProbability`** with a default value of **0.2**.

   ii) All monsters should only inherit the generic methods **`attack()`** and **`move()`**, but the generic **`attack()`** method should be modified so that no child class can *override* it[3]. In particular,

   a. The generic **`attack()`** method should now use either a generic mode of attack or a special mode of attack (via a call to method **`specialAttack()`**). This decision is based on generating a random number and checking to see whether it is less than the value of **`spAttackProbability`**.

   b. If the generated random number is less than **`spAttackProbability`**, then the monster should attack via a call to the **`specialAttack()`** method. Otherwise, the monster should attack in a generic way.

   The **`move()`** method remains unchanged, with the same behaviour as described in *Q1*.

   iii) All monsters must have a special power (via a call to method **`specialAttack()`**). A **`Dragon`**'s special power is to breath fire, whereas a **`Troll`**'s special power is to hit with a stick (resulting in a damage of between **1** to **15** points). There is no longer a default method for **`specialAttack()`**; however, every subclass of **`Monster`** should be forced to provide a method for **`specialAttack()`**.

[2] **Hint**: A superclass is accessed with the **`super`** keyword.

[3] **Hint**: The **`attack()`** method's definition in the **`Monster`** class must now be declared as: **`public final int attack() {`**. The keyword **`final`** will be explained in detail, during **Teaching Week 3**.

iv) Provide two constructors for each kind of monster:
   - One constructor that sets up the instance variable **`name`**.
   - Another constructor that sets up both instance variables i.e., **`name`** and **`spAttackProbability`**.

v) After all these changes, you will notice that the provided **`TestingMonsters`** class will no longer work. Determine the reason for this and fix the code in this class so that you can run your modified Java application.

Ensure that all your programs contain both internal comments and *Javadoc* comments.