Strategy Homework Part 00

Assignment Goals

The goal of this assignment is to:

1. Practice converting UML to code
2. Practice creating abstract
3. Practice creating concrete classes
4. Build familiarity with managing large projects
5. BONUS: Push this code to a github repository

Table of Contents

Figures

Deliverables

We will create four classes:

1. Abstract class called monster
2. concrete class called Imp
3. concrete class called Kobold
4. One class called Driver.java

Turn in a .zip file with all four classes.

Overall Diagram

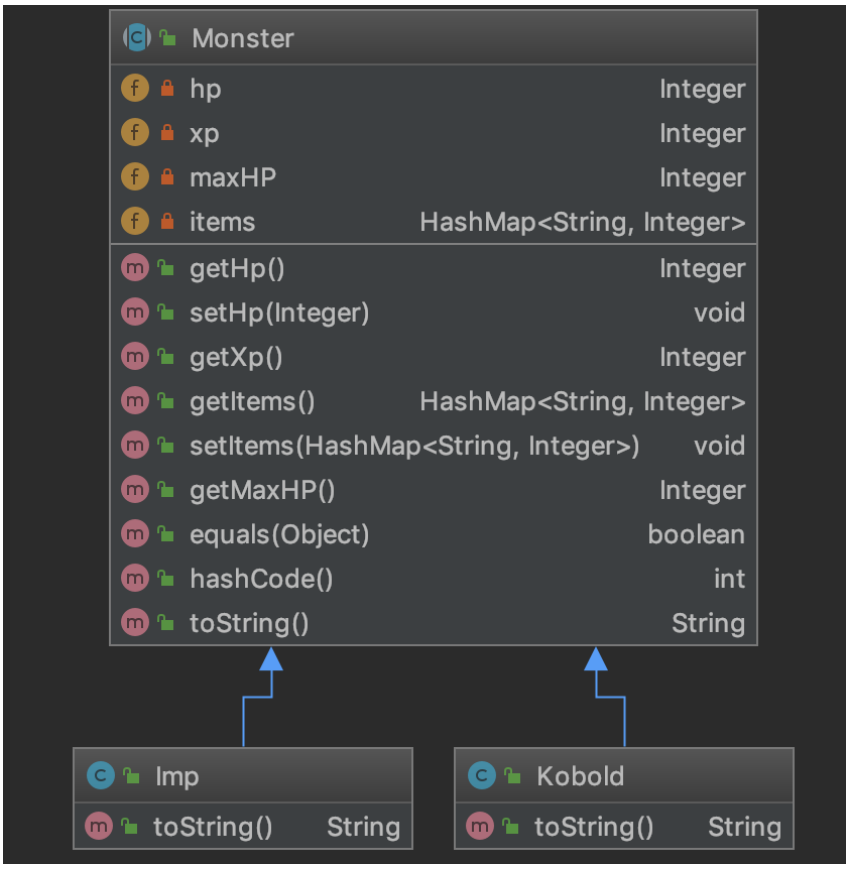

Figure 1: Overall Diagram

As detailed in Figure 1 there are four fields in Monster and several accessors and mutators. Pay special attention to the equals and hashCode() methods.

xp should have a default value of 10

With the exception of the toString method, all of the included methods may be generated using the code completion feature of IntelliJ.

Details of Monster

```
public Monster(Integer maxHP, Integer xp, HashMap<String, Integer> items) {
    this.maxHP = maxHP;
    hp = this.maxHP;
    this.xp = xp;
    this.items = items;
}
```

Figure 2: The Constructor in Monster

The two methods that must be written by hand in Monster.java are the Constructor and the toString().
The constructor is detailed in Figure 2.

The correct formatting of the toString() method should be inferred from the output shown in Figure 5.

Details of the Imp and Kobold classes

```
public class Imp extends Monster {

    public Imp(Integer maxHP, Integer xp, HashMap<String, Integer> items){
        super(maxHP,xp,items);
    }

    @Override
    public String toString() {
        return "Imp has : " + super.toString();
    }
}
```

Figure 3 The completed Imp class

The Imp and Kobold class are identical except for the toString method.
There will be more changes made in a future Strategy homework assignment.

Note the call to super() in the toString method. When not used in as part of a Constructor body, super may be used at any time.

The Driver class and the final output

```java
public class Driver {
    public static void main(String[] args) {
        HashMap<String, Integer> items = new HashMap<>();
        items.put("gold", 5);
        List<Monster> monsters = new ArrayList<>();
        monsters.add(new Imp(15, 20, items));
        monsters.add(new Kobold(1, 5, items));

        for (Monster m : monsters) {
            System.out.println(m);
        }
    }
```

Figure 4: The Driver.java class

The driver class is detailed in Figure 4
The output of the Driver class is shown in Figure 5.

```
Imp has : hp=15/15
Kobold has : hp=1/1
```

Figure 5: The correct output from the Driver file.