CSCI-605 Advanced Object-Oriented Programming Concepts

Homework 3: Predator

1 Introduction

Dutch and his team of elite soldiers are tasked by the CIA to rescue a group of officials being held hostage in an enemy base deep in a Central American jungle. The base is protected by a heavily armed force of insurgent guerrillas. Intelligence shows that the hostages are being held in a narrow cave inside the base with only one entrance and no exit - lined up from the front to back in standing position and unable to reorder themselves.

The rescue mission works as follows. One by one a soldier will leave their bunker and enter the nearby enemy base. Upon entering an alarm is activated which causes the first guerilla to leave their line and investigate. The guerilla engages the soldier and one of two scenarios play out:

1. The soldier is victorious over the guerilla and the guerilla is vanquished. In this case, the soldier rescues the one hostage at the head of the cave and escorts them to a nearby chopper that whisks them away to safety. Next, the soldier returns to the back of the bunker to await their next rescue attempt.
2. The guerilla is victorious and the soldier is vanquished. In this case, the one hostage attempting to be rescued adds themselves back to the head of the cave (effectively pushing the other hostages further into it) and the guerilla returns to the end of the line to await their next encounter.

Unbenownst to the soldiers, an undetected alien spaceship has crashed into the same jungle area. One lifeform survived the crash - a ruthless creature known as the Predator. The Predator is the ultimate fighting machine. It is a beast who knows no mercy and hunts for game using its superior innate alien camoflauge and thermal tracking abilities.

In this homework, your will implement a simulation for the rescue mission in which each soldier will try to rescue a hostage and take them to the chopper. During the mission, a soldier will fight againts a member of the guerrilla and the predator.

Who, if anyone, will "get to the choppa!"?

1.1 Goals

This homework helps students to gain experience working with:

- Interfaces and Classes
- Method overriding
- Polymorphism

1.2 Provided Files

1. The **class_diagram.png** illustrates the program's design.
2. The Javadoc documentation for the classes here.

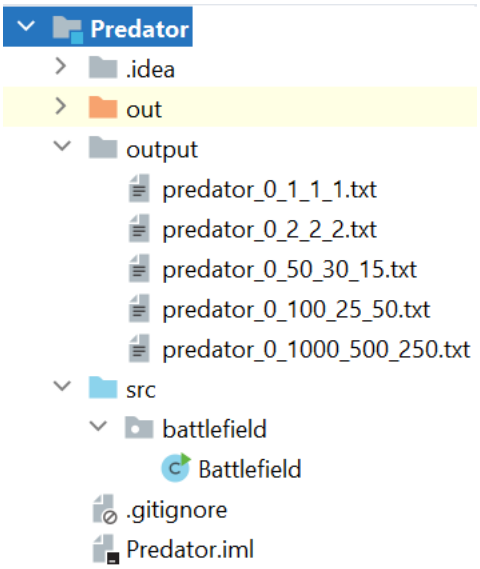
2 Implementation

2.1 Starter Code

With this homework, and all future ones, you will need to use a Version Control System (VCS) to maintain your code. VCSs are used throughout all computing fields to allow multiple people to collaborate on shared software, and to prevent the loss of work if something is accidentally deleted. For this course GitHub will be as the VCS. Please see the installation guide here for instructions on setting it up.

Note: You must use your RIT username for your GitHub account. If you already have an account set up that doesn't use that name, you will need to create a new account for use with this course.

When you get to the **Working on Your Assignment** section you should use the following GitHub classroom link to create and clone your repository. Your project structure should look like this:



The `output` folder contains sample runs. The numbers on the file names (e.g., 0 1 1 1) are the values of the command-line arguments used when running the program (see the Program Specifications below).

2.2 Design

You have been provided with the design for this homework. Notice that both the class diagram and the documentation only include the public state and behavior. Take some time to familiarize yourself with the design of this battle simulation. Make sure you understand the roles and responsibilities of each class and how they interact with each other.

2.3 Program Specifications

Your implementation must meet the specifications below:

2.3.1 Player

All of the "actors" in the simulation e.g. Hostage, Guerrilla, Predator and Soldier, implement the **Player** interface. These are used to produce the victory and defeat messages that are called out by the players. Please, check the Java documentation provided for more information about the victory and defeat messages of each player.

Note that the `Player` interface and all the classes that implement it must be created in the package `players`.

2.3.2 Hostage

Every hostage is identified by a unique id. Hostage's are held up in the enemy base awaiting rescue from a soldier who defeats a guerilla. They all speak English.

2.3.3 Guerrilla

Every member of the guerrilla is identified by a unique id. The guerilla's guard the enemy base and engage the soldiers who enter and try to rescue a hostage. A guerrilla's guard has a chance of 20 points to defeat a soldier. All the guerrillas speak Spanish.

2.3.4 Soldier

The soldiers are attempting to rescue the hostages from the enemy base and escort them to the chopper. In honor of lead soldier Dutch, who is Austrian, all the soldiers speak German. To enter the "umlaut" character you can cut and paste it from the javadoc, or you can use the unicode string "ŭ00FC" in your code.

Also note that whenever you encounter a message that contains curly braces, e.g. {`soldier`}, it means you should invoke the Soldier's `toString()` method.

2.3.5 Predator

There is only one predator in the game. The predator has different chances to defeat a soldier and a hostage that escapes from the enemy base.

The chances of defeating a soldier is 75 points, while the changes of defeating a hostage is 50 points (see the class diagram).

2.3.6 Bunker

The bunker is where the soldiers are stationed. The bunker can be viewed as a queue (first soldier in is the first soldier out).

2.3.7 EnemyBase

The soldiers enter the enemy base from their bunker. The guerrillas are waiting in a guard line which is another queue. The hostages are stored in a cave which is a stack.

When the guerrilla at the front of the line faces the soldier, they use the `Battlefield.nextInt` method to roll the dice and determine a victor. If the soldier is victorious they exit the base with the hostage at the head of the cave. Otherwise, the guerrilla re-enters the back of the guard line.

2.3.8 Chopper

The chopper takes both hostages and soldiers escape to safety. It has a limited capacity of 6 passengers. Each time the chopper is full, it has to fly away to rescue the current passengers and then return back to take on more passengers.

The storage of passengers in the chopper is a stack.

2.3.9 Battlefield

Battlefield is the main class which runs the entire simulation. You are given the main method and the dice rolling method. The program expects four integers on the command line, e.g.:

```
$ java Battlefield #_seed #_hostages #_soldiers #_guerrillas
```

`#_seed` is the seed for the random number generator to simulate the die rolling.
`#_hostages` is the number of initial hostages in the simulation.
`#_soldiers` is the number of soldiers at the bunker.
`#_guerrillas` is the number of guerrillas in the enemy base.

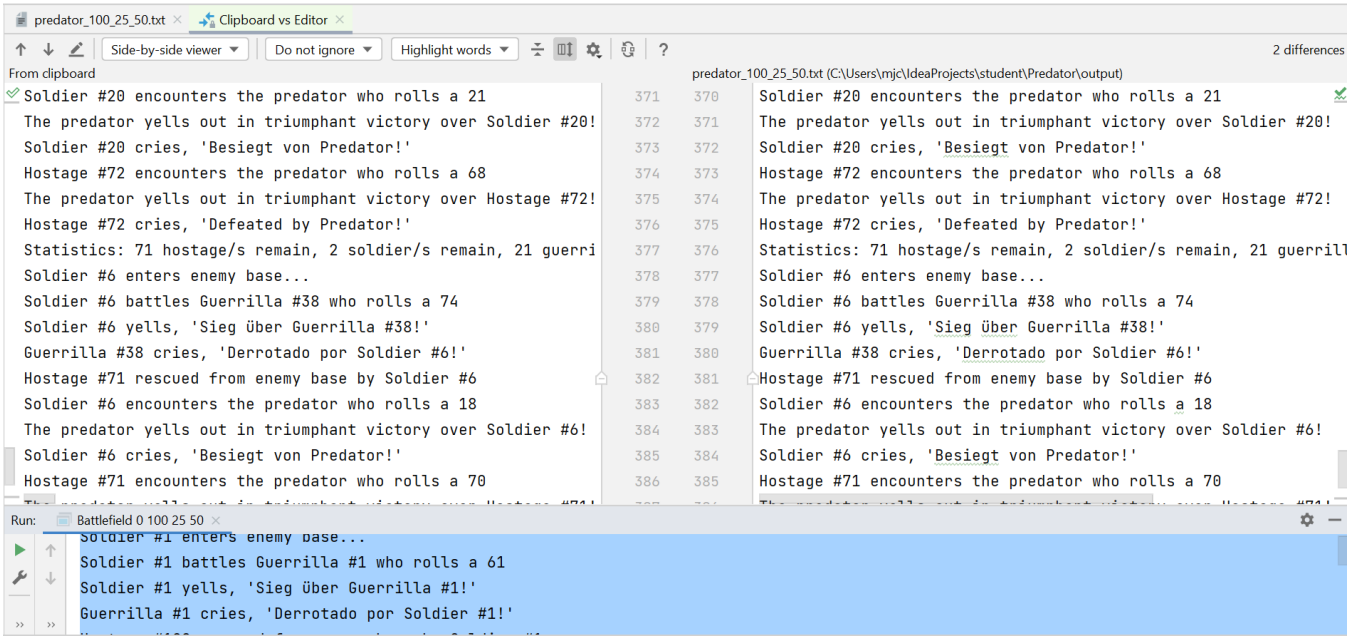
Notice that while the predator may be temporarily defeated by a soldier or hostage, it can never be truly defeated. The predator is around and ready to battle every soldier and hostage that comes out of the enemy base.

If there are no more hostages left in the enemy base, any soldiers left in the bunker may safely board the chopper in order from the bunker.

2.4 Comparing Output

The supplied outputs have a lot of text and can be very confusing to compare your output to. IntelliJ provides a utility so you can compare a text file to the clipboard in a visual manner.

1. Open one of the output files in the IntelliJ editor window.
2. Run the program with your desired command-line arguments.
3. Select all the text in the console and copy it to the clipboard.
4. In the solution output editor window right click and select **Compare with Clipboard**.
5. A new editor window will open highlighting the differences if there is any.



The ultimate goal here would be to make sure the output matches exactly.

3 Submission

You will need to submit all of your code to the MyCourses assignment before the due date. You must submit your `src` folder as a ZIP archive named "`hw3.zip`" (if you submit another format, such as 7-Zip, WinRAR, or Tar, you will not receive credit for this homework).

4 Grading

The grade breakdown for this homework is as follows:

- Design: 15%
- Functionality: 75%
- Code Style, Commenting and Version Control: 10%