INFORMATICS LARGE PRACTICAL COURSEWORK 2

Theodor Amariucai

STEPHEN GILMORE, PAUL JACKSON
SCHOOL OF INFORMATICS
UNIVERSITY OF EDINBURGH

Table of Contents

1. Software architecture description

This section provides a description of the software architecture of your application. Your application is made up of a collection of Java classes; explain why you identified *these classes* as being the right ones for your application. Identify class hierarchical relationships between classes: which classes are subclasses of others?

My application comprises 7 different Java classes: App, Game, Map, Drone, Stateless, Stateful, Position and Direction. I identified each one of them as having a separate, well-defined purpose within the architecture, and I believe this is the correct layout when having to address the 4 principles of Object-Oriented Programming: Inheritance, Encapsulation, Abstraction and Polymorphism.

The 7 classes mentioned above each encapsulate a different. The data is hidden from the user, with only a few public methods available for other classes to interact with. Class cohesion is low because

Considering the hierarchical relationships between classes, Stateless and Stateful are both subclasses of abstract class Drone. This is in accordance with their slightly distinct logic: they bear different variables and methods as their strategy is different.

2. Class documentation

Provide concise documentation for each class in your application. Explain each class as through providing documentation for a developer who will be maintaining your application in the future. Java design patterns?

The **App** class defines the entry point in the program: it has the main class in which command line arguments are parsed accordingly, and then passed over to the Game class which will handle other logic.

The **Game** class defines

The **Map** class defines

The **Drone** class defines

As part of the Drone class, I implemented the Singleton design pattern to ensure that only a single instance of the Drone class will ever be instantiated. This is a safe and straightforward way of complying with the logic of the project: under no circumstances should the Drone class have multiple instances in the same game. To enforce this in a single-threaded application such as PowerGrab, it is sufficient for the Drone class to implement the field and method as below:

```java
private static Drone instance = null;

static Drone createInstance(Position position, long seed,
boolean submissionGeneration) {
    if (instance == null || submissionGeneration)
        instance = new Stateless(250, 0, position, seed);
    return instance;
}
```

The **StatelessDrone** class defines

The **StatefulDrone** class defines

The **Position** class defines

The **Direction** class is an enumeration of 16 cardinal points. In addition, it provides two static variables and a static method randomDirection().

3. Stateful drone strategy

This section explains the strategy which is used by your stateful drone to improve their score relative to the stateless drone. You should explain what is remembered in the state of the stateful drone and how this is used to improve the drone's score.

This section of your report should contain two graphical figures (similar to Figure 6 and Figure 7 in this document) which have been made using the http://geojson.io website, rendering one flight of your stateless drone and one flight of your stateful drone on a PowerGrab map of your choosing. It can be any of the available PowerGrab maps, but make sure that the same map is used for both the stateless drone and the stateful drone.

The maximum page count of your project report is 15 pages with:
title pages
table of contents
references
appendices
and all other material included in the page total.

References

There are no sources in the current document