Software Development CA

Continuous assessment

Developers’ candidate numbers:

Thomas Mack - 163034

Pedro Pereira - 157665

Both developers agreed that the final mark be allocated on a 50:50 split

# Index

[Index 2](#_Toc87455843)

[Development Log 3](#_Toc87455844)

[Design 4](#_Toc87455845)

# Development Log

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Date | Time | Duration | Observer | Driver | Signatures | |
| 14/10/2021 | 14:00 | 2 hours | - | Pedro/Thomas | 157665 | 163034 |
| 18/10/2021 | 16:30 | 1 hour | Pedro | Thomas | 157665 | 163034 |
| 19/10/2021 | 11:30 | 3 hours | - | Pedro/Thomas | 157665 | 163034 |
| 21/10/2021 | 14:00 | 1 hour | Thomas | Pedro | 157665 | 163034 |
| 22/10/2021 | 17:00 | 3 hours | - | Pedro/Thomas | 157665 | 163034 |
| 25/10/2021 | 18:30 | 2 hours | - | Pedro/Thomas | 157665 | 163034 |
| 28/10/2021 | 14:00 | 2 hours |  | Pedro/Thomas | 157665 | 163034 |
| 01/11/2021 | 14:00 | 4 hours | - | Pedro/Thomas | 157665 | 163034 |
| 02/11/2021 | 10:00 | 8 hours | - | Pedro/Thomas | 157665 | 163034 |
| 04/11/2021 | 10:00 | 3 hours | - | Pedro/Thomas | 157665 | 163034 |
| 08/11/2021 | 14:00 | 5 hours | - | Pedro/Thomas | 157665 | 163034 |
| 09/11/2021 | 9:00 | 10 hours | - | Pedro/Thomas | 157665 | 163034 |
| 10/11/2021 | 9:00 | 4 hours | - | Pedro/Thomas | 157665 | 163034 |

A brief explanation of some of the aspects of the development log.

Other than some of the shorter work sessions the development was able to be performed simultaneously by both developers in a driver role. By driver role we mean that both developers were actively changing the code and not watching the other do said action.

This is due to the multiple collaboration tools that were available to and were used by us during the process of development of which we will highlight *GitHub* which allowed us to share our work between our devices and a feature of *Intellij IDEA*, Code With me. This feature allowed us to make changes to a file at the same time with the changes registering in each other’s computer.

For these reasons, we have placed both developers on the role of driver in most of the work sessions since, essentially, after planning how to do the work every session was spent developing this work by both developers, simultaneously.

# Design

To develop this pebble game, it was necessary to first think of what entities existed within the game, and the conclusion was that there we 2, the players participating in the game and the bags that hold the pebbles.

In light of this, the Player and Bags classes were created as nested classes inside the PebbleGame class. Since both Player and Bags are classes inside the pebble game class, we had to choose whether to have them be static nested classes or an inner class.

For the Player class it was chosen that it would be an inner class since methods inside of it need to access instances of classes outside of it, namely the bags class, as can be seen in the example below:



Said line of code would not be possible if the Player class was static since it would not be able to access the bags, an instance of the bags class declared in the PebbleGame outerclass.

For the Bags class it is a similar case to the Player class where an inner class was chosen because of its ability to access instances of classes outside of it, only this time it necessary for the bags class to access an instance of itself declared in the outer class PebbleGame, as is illustrated in the example below:



Said line of code would not be possible if the Bags class was static since it would not be able to access the bags, an instance of the bags class declared in the PebbleGame outerclass.

The bag class holds a list of all the black bags and another list of all the corresponding white bags. This is done in a way where the white bag that is paired with the black bag is the same index in both lists, so black bag A will be index 0 and its corresponding white bag, bag X, is index 0 too, making it easier to access and modify the bags.

After this initial design decisions, we had to lay out what interactions between the players and the bags would have to occur in the game without interruption, meaning these would have to act as atomic actions, they are as follows:

* Player stores a random pebble from one of the three black bags in his own player bag;
* Check if the removal of the pebble makes the black bag empty, if so transfer all pebble from the corresponding white bag to the black bag, leaving the black bag full with its initial pebbles and the white bag empty;
* Player discards a pebble from his bag and it’s stored in the white bag that corresponds to the black bag the player took a pebble from;

Each one of these actions consists of smaller sub actions, all of which endanger the game as they can ruin its functionality due to the possibility of multiple players simultaneously picking the same pebble, for example.

This is where the second phase of our design process commences, where we have to make sure that the actions and the sub actions that compose them don’t interfere with each other.

Thread safety