ECM2414 Pair Programming CA

Tomas Premoli, Lucas Proudhon-Smith, mark split 50:50

# Development Log

* A development log, which includes date, time and duration of pair programming sessions, and which role(s) developers took in these sessions, with each log entry signed by both members (using your candidate number as your signature). This part of the document should be no more than one side of A4.

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Lucas contribution** | **Tomas contribution** | **Signed** |
| Oct 15, 2021 | Helped setup GitHub repo, discussed how to go about doing the coursework. | Setup GitHub repo, discussed how to go about doing the coursework. | Lucas: 700074221  Tomas: |
| Oct 17, 2021 | Created and worked on UML to plan methods and classes. | Worked on UML to plan methods and classes, started work on basic outline of the program. | Lucas: 700074221  Tomas: |
| Oct 18, 2021 | Added documentation, worked on generation of bags, general bag methods, general code fixes and optimisation. | Added documentation, worked on generation of bags, general bag methods, worked on main game method. | Lucas: 700074221  Tomas: |
| Oct 20, 2021 | Worked on takeRandomPebble(), getTotalPebbleWeight() and bag creation, general code fixes and optimisation. | Worked on takeRandomPebble(), getTotalPebbleWeight(), improved encapsulation for PebbleGame, general code fixes/optimisation. | Lucas: 700074221  Tomas: |
| Oct 25, 2021 | Worked on player constructor, getRandomBlackBag(), swapPebble(), getCounterPart(). General code fixes/optimisation, documentation | Worked on error handling, logging player moves to a file and swapPebble(). General code fixes/optimisation, documentation. | Lucas: 700074221  Tomas: |
| Oct 26, 2021 | Worked on swapContents(), threading aspects of code, general code fixes/optimisation | Fixed file writing, worked on swapContents(), threading, general code fixes/optimisation. | Lucas: 700074221  Tomas: |
| Oct 29, 2021 | Worked on write up | Implemented run(), implemented exit on ‘e’ input, changed swapPebble(int pebbleWeight) for swapRandomPebble() for better encapsulation between objects. | Lucas: 700074221  Tomas: |

# Design choices

## Production code

Timeline

Description automatically generated with low confidenceTo start off with, after an initial discussion and quick read through of the project spec, we decided to make a rough UML so we could outline the basic structure of the code, figure out which methods we’d need and further understand the specification provided.

As we continued working on the project, we went slightly off the original UML’s plan as we figured out more efficient/better ways to create the program. We also found several issues with the UML due to parts of the spec that we had misunderstood, such as changing the swapPebble(int pebbleWeight) to swapRandomPebble() as originally we thought that a player would need to input the pebble they wanted to swap.

In the final program, there were 4 main classes:

* Main
  + A class with methods to:
    - Run the actual game, get user inputs for pebble file locations and player count
    - Create player threads
    - Generate bag objects
* PebbleGame
  + A class which holds the attributes for:
    - An atomic Boolean value used to check if the game is finished or not
    - A lastBag attribute so the program knows what the last bag drawn from was
    - A hashmap of bags and an arraylist of player objects.
  + Contains methods to get/set attributes, create black bags and end the game
  + Also contains a nested ‘Player’ class which holds attributes for
    - An int playerID
    - An array of pebbles that the player currently has
    - A string which stores the location of the player’s output file
  + The Player class contains all the necessary methods needed for a player thread.
* Bag
  + A class which holds attributes for:
    - A hashmap of Bags
    - A bagType value
    - The string of the fille location where a bags content is stored
    - An arraylist of a bag’s pebbles
    - The bag’s corresponding character (bagName)
  + The class also holds the bagType enum used to distinguish between black and white bags
  + The class holds all the methods needed to work with a bag and it’s contents
* PebbleErrors
  + A class used for processing errors including:
    - IllegalPlayerNumberException - thrown when the provided player number is not positive.
    - IllegalBagTypeException - thrown when a method is used on a bagType that the method cannot be used on.
    - NotEnoughPebblesInFileException - thrown when there aren't at least 11x the number of valid pebbles as number of players.
    - NegativePebbleWeightException - thrown when a pebble weight in a file is negative.
    - Empty Bag Exception - thrown when a player attempts to draw from an empty bag.

## Testing

* A document detailing the design choice and reasons with respect to your tests of your production code. You may use either of the JUnit 4.x or 5.x frameworks, but you should explicitly detail which framework you are using in your document. This part of the document should be no more than three sides of A4.