|  |  |  |  |
| --- | --- | --- | --- |
| Date | Time | 700031791 | 700037512 |
| 22/10/2021 | 9:45 – 11:10  1 hour 25 mins | Design decisions & began writing unit test | Design decisions & began writing unit test |
| 25/10/2021 | 10:30 – 11:30  1 hour | Driver for testing bags then switched for bag creation functionality | Navigator for testing bags then switched for production code |
| 25/10/2021 | 12:00 – 4:00  4 hours | Implemented unit testing and coded for 2 hours on the pebbleGame class & Bags | Went over testing specifications and coded for 2 hours on pebbleGame began threading. |
| 28/10/2021 | 10:00 – 12:00  2 hours | Tested methods in pebbleGame related to bags and improved functionality as well as verification | Tested methods in pebbleGame and began to write player methods |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Design Decisions:

Initial thoughts-

Project would be most effectively coded through test driven development so we must have a firm grasp of requirements before proceeding. Understanding how to program in a thread safe manor and ensure all created files are working with production code will be essential to smooth running.

Project requires jar with both byte classes and source file, files include:

* Interface for the game
* User output files txt
* Black and white bag csv files
* Nested classes with players (threads and listeners)
* Game set up file (main file)
* Bag creation file

Decided we wanted random numbers between 0 and 25 for the game to work.

Whilst creating some code for the main application PebbleGame we reread the spec and concluded it may be a good idea to have the bags file within that file as a nested class like the players threads to make it easier for testing and overall development.

We decided we wanted to include sufficient verification and used while loops to check that user inputs were of the correct form.

Testing Decisions:

Black bags:

* Length of list (correct number of pebbles in bag)
* All the values of the list are positive integers
* Check all numbers are within the range >25 (this is not a rigorous test as it involves random numbers)
* Check all files are of the right format (list of numbers with commas between them)

White bags:

* Check they are empty at the start of the game

Players:

* Have 10 pebbles unless they are in the process of discarding one
* Output file created when thread is started

Started by testing the bag files themselves, then our next step is to write tests for the functionality of creating the bags correctly. To correctly test our functionality, we need to produce some erroneous test files that would check our testing works, e.g a list containing negatives or floats.

Having created private methods for bags we realised we didn’t have a way to access them in the test whist keeping them separate therefore we need to code our public interface and test the public methods which will call the private ones.

We have created the user interface and can run it, initialising bags from user input, this allows us to test what we have created so far before moving onto player functionality.