CSC207: Project Update

Edward, Terry and Yan

November 29, 2021

1 Design (Nov 28/2021)

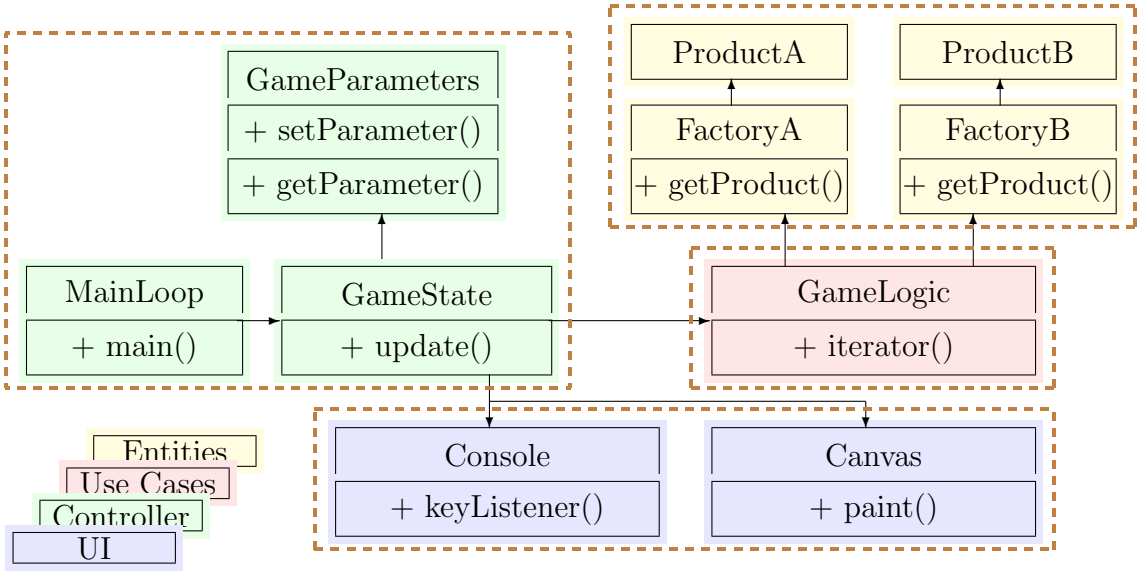

Figure 1: Missile Mayhem packages and calling dependencies.

2 Weekly Update

2.1 Packaging & Dependency Injection (Yan)

All classes are now in one of four folders corresponding to four layers: Entities, Logic, Controller, and UI. This was a major effort because of unexpected

dependencies created by `JPanels` (products) since they contain key listeners. A solution was found in **Dependency Injection**. At runtime a `JPanel` gains access to `GameParameters` that is passed to it as a parameter.

2.2 Interface (Edward)

To solve the issue of products with key listeners, class `GameParameters` was implemented as an interface between `GameState` and `JPanels`. It is simply a list of various parameters describing the state of the game. The advantage of this interface is that `JPanels` update `GameParameters` without a need for knowledge how these update are used and how they affect the other parts of the program.

2.3 Serialization (Terry)

Class `ScoreBoard` has been created to store and fetch game users and their scores. A text file contains a list of two fields separated by a colon. Method `write()` records the current score if higher than an existing score. Method `read()` reads the text file and creates an `ArrayList` of tuples which are of type `Tuple` and contain a string for the username and an integer for the score. Class `ScoreBoard` also contains a method to sort the tuples by the score. This will allow the program to display top five players at the end of each game session.

3 Bugs and fixes

3.1 Maps cannot be sorted

Even when elements are put into a map in an ordered fashion, the map will rearrange them by `key` within a tree structure. The order by `value` cannot be restored. Our solution was to abandon `Map` and create a custom `ArrayList<Tuple>`.

3.2 JFrame cannot manage multiple components

Both `JFrame` and `jFrame.getContentPane()` have the same methods to add, revalidate, remove, and repaint. Experimentally, we have established a sequence that works.

```
jFrame.setVisible(true);
jFrame.getContentPane().removeAll();
jFrame.getContentPane().add(jPanel);
jFrame.getContentPane().add(jPanel);
jFrame.getContentPane().revalidate();
jFrame.revalidate();
jFrame.repaint();
```

4 TODO

Functionality is implemented. The game runs from the beginning to the end smoothly, with pauses allowed at any time during the game. The focus now should be on code cleanup.

- For classes /** @author @version @since */
- For methods /** brief <p>long description <p>@param @return */
- Identify what information crosses boundaries
- Complete design patterns
- Look for compliance and non-compliance with SOLID principles
- Export tests from desktops to Intellij on GetHub. This may turn out to be difficult because test cases for `JPanels` were large and heavily modified.
- Final Report