SENG201 Assignment Report

Ryan Hirst – 13325087

Ben John – 46370065

Application Structure

The structure of the game is centred around the Game Environment class. This is the point where the GUI, the behaviour, and any objects communicate. This was done to allow the easy integration of other classes, and GUI screens into the application. All the information needed (the player team, the market, and the stadium) can be accessed, and easily displayed to the user. The Sled and Contestant classes both inherit the TeamMember class. These classes then get access to basic getters and setters, as well as key variables (such as name, value, and attributes). The TeamMember class is generic and allows for various types of attributes and modifiers to be added to its' members (both enumerators). We did this as it allows additional team member types to be easily added in the future (such as coaches and managers) with individual sets of attributes, modifiers, and behaviour. Both the Sled and Contestant classes use Enumerators for

A BaseTeam has also been implemented, which is extended by the PlayerTeam and ComputerTeam classes. This BaseTeam provides all the basic functionality that all teams need such as: An ArrayList of active contestants, reserve contestants, and basic getters and setters. This base class also uses a separate Item enumerator as the source of Item definitions. The PlayerTeam also implements the TeamBehaviour interface. While for this version of the application it is redundant, it could be used later to implement behaviour for persistent computer teams that level up with the player. The Utils class is used throughout the application and provides methods to generate random contestants and sleds, and a method to write a 2D array to a Map via a stream (used to convert 2D object arrays to attribute - integer maps). The random generation methods are used throughout the program therefore, a static class was created to allow for easy access of the method throughout the program, without having to create a new object.

Both TeamMember extensions (Sled and Contestant) have separate Attribute and Modifier Enumerators. The TeamMember extensions can have a unique set of static attributes and modifiers, which allows for consistency throughout the application. In future versions, these attributes and modifiers could also have unique behaviour, instead of just storing a value.

Unit Test Coverage

Overall, the application has a very low-test coverage (16.3%). However, most of the untested code is in the GUI classes of the application. When these classes are removed, the test coverage is 51.2%. This is still relatively low, as much of the testing occurs in the Market, Stadium, and TeamMember classes. These classes contain most of the behaviour and are worth testing to ensure that the behaviour works as expected. Many of the other classes largely contain getters and setters, which for this project would not be worth writing unit tests for.

Testing for our application mainly occurred in manual testing. The GUI itself was tested extensively to ensure that all the functions worked as expected, and that the edge cases were accounted for (such as entering a blank name into the team set up screen).

Feedback

Our thoughts are that the project was about as good of a project it could've been within the scope of the course. The topic of the project kept it interesting and allowed us to develop something that we personally were interested in. The size of the project was appropriate and was a reasonable amount of work to complete within the timeframe. The project specification was the main problem with the development of the application. We found that many of the requirements were particularly vague and left too much to interpretation.

Reflection

Overall, we found that the project went well. We were able to complete the application within the timeframe and fulfil the requirements. The game works well, and due to the design, would be easy to develop further. However, we found that our management skills were lacking. While the application works as intended, it is not as polished as we would like. The GUI is barebones and is not aesthetically pleasing. The layout is basic and while it is functional, it could be improved significantly to improve the user experience. For example, the current system displays one contestant, and has the player switch through them. A better design would have all the contestants displayed on a grid, so the player could view all their players at the same time. In future, we would manage our risks much more effectively, specifically completing all our objectives within the timeframe. For future projects we would employ more effective time management techniques, such as the use of Gantt charts, to ensure that we complete all our objectives on time.

Hours Completed

Ryan Hirst – 50 hours completed.

Ben John – 45 hours completed.

We (Ryan Hirst and Ben John) have agreed that we have both contributed 50% to the project.