Tower Carts Report

Tyler Gamble (99623422) Yang Zhang (37594777)

Structure of the Application

In this project, we developed a tower game application based on Java and JavaFX. The project structure is clear and covers multiple modules. Enumerations include the Difficulty, ItemType, material, and Type enumerations. These define in-game difficulty, item types, and material properties. The core game logic is mainly centered around an element package that includes the Cart, Player, Rounds, Tower, and Upgrade classes. These classes are responsible for managing the interaction of the player, shopping, rounds and tower. Player interaction of the interface is handled through her GUI controllers such as his FailScreenController, GameScreenController, and MainWindow in the GUI package. These controllers implement direct user interaction and interface updates. Additionally, the application defines a time listener function using his TimerListener interface. This makes time management more flexible and easier to maintain and expand. The game's management logic is coordinated by his GameManager and ItemsManager classes in the manager package. These two management classes not only improve code reusability, but also simplify game status tracking and item management. The model package's Counter class provides basic counting functionality, and the service package provides services including CounterService and MarketService, which are responsible for timing functionality and market transaction processing, respectively.

This project adopted the MVC design pattern to clearly separate the game's business logic (Model), user interface (View), and data control logic (Controller). This structure not only makes the code clearer, but also greatly improves program maintainability and extensibility. We make extensive use of interfaces such as his TimerListener. This provides a consistent way to handle time-driven events, increase application flexibility, and provide separation between modules. With these design decisions, our project ensures application performance and stability while providing convenience for functionality expansion and subsequent maintenance.

The classes communicate with each other using the manager classes ItemManager, and GameManager. These manager classes hold the other classes within them and interchange the barbies between the different classes. ItemManager is associated with Item which has the subclasses tower and upgrade. GameManager is associated with Player, Rounds, Carts, CounterService, MarketService and is also associated with ItemManager to make it easier passing it to the gui controllers. The gui classes are launched from within the MainWrapper class and are passed GameManager, to manage their variables. GameScreenController and GameManager use the interface TimerListener to update their variables whenever one second passes in the CounterService class, allowing the game to take place over time.

Unit Test Coverage

This project uses the JUnit framework for unit testing to verify the functional correctness and stability of each class and method. We pay particular attention to testing core game logic such as the GameManager and Player classes to ensure that game state management and player interactions are accurate. Coverage reached approximately 57%, but we found that some error handling code and some boundary conditions were not sufficiently tested, so we will need to add more test cases in the future to improve coverage.

Thoughts, Feedback and Brief Retrospective

In this project, we found the game development process using JavaFX and SceneBuilder to be very interesting, as they allow us to create different graphical user interfaces. SceneBuilder's visual design capabilities let us intuitively build complex interfaces and see the effects of layout adjustments in real time. However, effectively combining these beautiful GUI interfaces with backend logic is a big challenge. MVC design patterns should be deeply studied and applied to ensure high cohesion and low coupling of the code, especially when it comes to ensuring that user interface events correctly trigger the corresponding backend processing logic. Additionally, our team used gitlab for code management and version control, and while this greatly facilitated team collaboration and code sharing, it was difficult to manage branches and manage merge conflicts during the early stages of the project. We also faced some challenges such as processing. This project led us to focus on creating clear commit messages and conducting regular code reviews and merges to avoid potential problems. Overall, through this project, we not only learned a lot about JavaFX's specific technology and GUI design, but also gained a deeper understanding of the software development process and project management.

Effort spent hours and Contribution Percentage

	Coding	Testing	Documentin g	Total Hours	Contribution
Tyler Gamble	36	8	26	70	50%
Yang Zhang	20	28	12	60	50%