Version 1.0

winter 2019

syst 17796

Deliverable 3

## description

In Deliverable 2 you committed to a design for your game and you will have received feedback on the design. The first step for Deliverable 3 is to incorporate that feedback so you have a firm UML class diagram of the final game so that you can begin to code. Of course, the quality of your submission also needs to be verified by testing the final product against the original requirements. This will be achieved by creating and executing JUnit tests as well as manual tests for any features that cannot be testing at the unit level.

The same standards for groupwork, professional writing style and citations apply to all deliverables. If you have questions, you can refer to the project description or ask your instructor. **The project requirements are not to be reduced for groups smaller than 4 students. Any groupwork conflicts will be dealt with using the contract from Deliverable 1.**

To begin, you will update your UML Class Diagram, incorporating the feedback you received in Deliverable 2. **You should include methods and attributes for each class**.

Next, complete the code to satisfy the rules of the game (requirements) and the use cases you defined (scope). Ensure the final version is checked in to your Git repository.

After the code is finished, take the fully developed use cases from Deliverable 2 and analyze them to produce unit tests for your code. Make sure you have “good, bad and boundary” tests for each requirement.

Create your tests using JUnit and code them so that they verify the good, bad and boundary cases of each method or unit (remember it won’t always be easy to divide your code neatly into methods). For any requirements that cannot be verified at the unit level, write manual test scripts (a list of manual steps you will take to verify the requirement). Marks will be awarded for proper design of tests as well as your source code so pay proper attention to the naming of the tests and the use of setup and teardown methods where applicable.

Finally, produce a Test Results Report that lists the requirement you are validating, the test that covers it and whether that test passed or failed on your final run (be honest, this is easily double-checked). An example of a test results report is below but you can modify it if you wish.

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Use Case | Test Method (ClassName.methodName) | Status (Date) |
| Player can draw a card from the deck | “Regular Play” | TestPlayGame.testDrawCard() | Pass (June 26, 2018) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Submission

Please submit **one PDF document per group as well as the access credentials (URL pointing to your repo and make sure the Instructor has read access before submission) to your Git Repository**. This means that the document should be professionally organized and have a uniform style throughout. It should look as though it came from one team, not 4 separate students. Please note that instructors may choose to run your submission through TurnItIn or compare the submission with other students from other sections for academic integrity purposes. Please take the time to properly cite your sources.

**You should also include a zip export of your project including the source code for your game and for your tests, labelled with your group name.**

### rubric

| Item | Criteria | Points | Weight |
| --- | --- | --- | --- |
| Class Diagram | Updated to reflect feedback from Deliverable 2, methods and attributes included. Notationally correct and solid design as described in Deliverable 2 Design Document. | 10 | 1 |
| Source Code | Code is completed and conforms to the design specified by the class diagram. Code is functionally correct. Code follows standard coding conventions for comments, naming and indentation | 10 | 3 |
| Source Code Design | The Design produced follows the principles of OOD studied in the course and provides flexibility, reusability and efficiency | 10 | 2 |
| Test Code + Scripts | JUnit tests are completed and pass. JUnit tests validate the one thing they are written to validate (each). JUnit tests conform to test standards taught in class. Manual tests scripts are included for any requirements that are unable to be validated using JUnit | 10 | 2 |
| Quality | JUnit tests cover the scope of the project as defined by the students in Deliverable 2 (use cases). Test results report is accurate and shows good, bad and boundary cases for all requirements | 10 | 2 |

## Final Document Layout

One PDF with the following sections denoted using page numbers, headers and a table of contents:

* Updated Class Diagram with methods and attributes
* Reference to the Git repository containing your source code
* Reference to the Git repository location (directory) of your JUnit tests
* Any manual test scripts
* Test Results Report

**AND** One zip archive containing the source code for your game and your JUnit tests, labelled with your group name.