Assignment #4

Todorka Dimitrova and Christian Christensen

Fall 2021

1 MOCKITO POWERUPS

Answer the following questions about Mockito. Use code examples in your explanations.

- How do you verify that a mock was called?
- How do you verify that a mock was NOT called?
- How do you specify how many times a mock should have been called?
- How do you verify that a mock was called with specific arguments?
- How do you use a predicate to verify the properties of the arguments given to a call to the mock?

2 AT LEAST ONE

Using TDD, make at least one of the following three tasks, A, B or C. Whatever you choose, include coverage report (e.g. Jacoco) and mutation testing (e.g. PITest), and static analysis (e.g. Findbugs, PMD, CheckStyle).

A: Snake game: Make a classic snake game using TDD. To remind you the (minimum) rules of snake (you can make more features if you like):

- You control a the direction of a continuously moving snake, going up, down, left or right – the snake cannot stop moving.
- At any point in time, there is an apple somewhere on the playing field.
- When the snake's head runs into the apple, the snake's body gets longer.
- The snake dies if it runs into its own body, or a wall (if your game has walls). If the game doesn't have walls, the snake should wrap around (like in Pacman).
- The winning state is to run out of space.

- Choose a point system of your liking. Inspiration:
  - Point(s) added for each apple eaten
  - Point(s) subtracted when starving (e.g. no apple eaten for an amount of time)

B: JSON-parser: Make a JSON-parser using TDD. Find the JSON RFC for reference.

C: Tic-tac-toe: Make a tic-tac-toe game using TDD. It should play against the human player. If you already had the Data Science course, you can try to make an AI with Minimax. Otherwise, simply have the AI make random moves – but the checks for win/loss, and the correct game flow (turn taking, handling human error of giving incorrect placement, etc.) must be implemented.

3 HAND-IN

Hand-in in groups or individually on the date given in peergrade. The handin should be code in a repository or zip-file, and a README.md with the written answers.