General

- **TAs:** A submission that does not contain files about this exam (e.g., the student submitted solution to a homework), or one that only includes the starter code, do not meet any of the specifications of this rubric (under any of the "required," "satisfactory," or "complete" categories). In such a case, please check this item to flag the submission and immediately inform the instructor.
- **TAs:** A submission where the starter code is modified in ways that were not allowed or one that employs Java's built-in data structures (other than those permitted), or data structures that are not yet covered in this course will not meet any of the specifications related to the question affected by this folly.

Required

1. **Spec:** True/false answers are correct except for at most two statements.
2. **Spec:** MCQ answers are correct except for at most three questions.
3. **Spec:** For Question-3 (short answer), Part I, the points made in the answers are on the right track.
   TAs: This specification is met even if the answer is too brief/abstract or contains partially incorrect, contradictory, or missing points.
4. **Spec:** For Question-3 (short answer), Part II, the points made in the answers are on the right track.
   TAs: This specification is met even if the answer is too brief/abstract or contains partially incorrect, contradictory, or missing points.
5. **Spec:** For Question-4 (unit testing), the code exhibits a good understanding of unit testing.
   TAs: This specification is met even if the code contains syntax or minor logical errors.
6. **Spec:** For Question-5 (implementation), the code encompasses a reasonable attempt at implementation.
   TAs: This specification is met even if the code contains syntax or minor logical errors.

Satisfactory

1. **Spec:** True/false answers are correct except for at most one statement.
2. **Spec:** MCQ answers are correct except for at most two questions.
3. **Spec:** For Question-3 (short answer), Part I, the points made in the answer are adequate.
   TAs: This specification is met even if the answer contains minor issues/inaccuracies.

However, it is not met if it contains contradictory points, or it is (unnecessarily) too long to the point it becomes a burden to read and find if it contains the expected answer.

4. **Spec:** For Question-3 (short answer), Part II, the points made in the answer are adequate.
   TAs: This specification is met even if the answer contains minor issues/inaccuracies. However, it is not met if it contains contradictory points, or it is (unnecessarily) too long to the point it becomes a burden to read and find if it contains the expected answer.

5. **Spec:** For Question-4 (unit testing), the code exhibits a solid understanding of unit testing.
   TAs: This specification is met even if the code contains minor syntax errors. However, it is not met if it has logical errors.

6. **Spec:** For Question-5 (implementation), the code showcases a solid attempt at implementation.
   TAs: This specification is met even if the code contains minor syntax errors or fails to account for a minor edge case. However, it is not met if it has obvious logical errors.

7. **Spec:** For Question-6 (implementation), the code encompasses a reasonable attempt at implementation.
   TAs: This specification is met even if the code contains syntax or minor logical errors.

Complete

1. **Spec:** MCQ answers are correct except for at most one question.

2. **Spec:** For Question-5 (implementation), the implementation is correct.
   TAs: This specification is most likely met if the corresponding auto-tests are passed. However, please ensure to check the code.

3. **Spec:** For Question-6 (implementation), the code showcases a solid attempt at implementation.
   TAs: This specification is met even if the code contains minor syntax errors or fails to account for a minor edge case. However, it is not met if it has obvious logical errors.

4. **Spec:** The submission exhibits good practices for writing readable code (e.g., consistent indentation, descriptive naming, etc.), and good programming style (e.g., code is organized in a modular fashion, methods are not too long, helper methods/classes are made private, etc.)
   TAs: You may use the output of Checkstyle, but take into account, the student is writing the code in a limited time, under the pressure of an exam.