

CMPT 213 Sample Midterm

Name: _____

Student Number: _____
(Neatly please!)

Signature: _____

DO NOT TURN THE PAGE UNTIL TOLD TO DO SO

You will have 50 minutes to complete this test once told to begin.

The test has 7 (multi-part) questions totalling 47 points. Points shown in square brackets.

Each question provides space for the answer, and your answer should appear in that space. **Multiple choice must be answered on bubble sheet** (detachable last page). Clarity and legibility count.

No aids are allowed: no crib notes, calculators, smart watches, textbooks, electronic aids, or dictionaries. If you do not understand the meaning of a word, ask the instructor or TA. **Reread the question carefully before asking questions during the exam.**

[10] Multiple Choice, worth 2 mark each.

Circle the one correct answer.

1. Which of the following statements is true about static methods?
 - a) Static methods cannot access static final fields.
 - b) Static methods cannot be private.
 - c) Static methods cannot change instance data.
 - d) Static methods cannot call other static methods.
 - e) None of the above; they are all false.

2. The *best reason* to make a class like `CarCollection` `Iterable` is...
 - a) it makes `CarCollection` immutable.
 - b) it allows `Collections.sort()` to be used on a `CarCollection`.
 - c) it allows client code to use a for-each loop over a `CarCollection`.
 - d) it allows `CarCollection` to reduce its coupling to `Iterator`.
 - e) it allows client code to break `CarCollection`'s encapsulation and access its private fields.

3. To support sorting instances of a class by multiple different sort orders we use:
 - a) `Sortable`
 - b) `Comparator`
 - c) `Object.compareTo()`
 - d) `Comparable`
 - e) Two of the above.

4. CRC cards are useful because they...
 - a) formally record the decisions, and the reasons for these decisions, made during OOD.
 - b) record the steps a user does while using the software.
 - c) show the low-level implementation details of each class, such as method and field names.
 - d) support refactoring code late in development.
 - e) none of the above.

5. Which of the following best describes the argument in the following Java code? Assume `processFile()` is a function which accepts one argument: a `File` object.

```
processFile(new File("C:\output.txt"));
```

 - a) Anonymous object
 - b) Anonymous method
 - c) Anonymous class
 - d) Anonymous field
 - e) Two of the above

Short answer questions. Points as indicated.

6. [10] Analyze the quality of the interface of `Maze`. Provide five (5) different ways that the quality of the interface can be improved, as discussed in lecture. For each improvement, **state the general name of the improvement**, and briefly **explain with an example** in the interface of where it applies. *Do not use the same named improvement more than once.*

```
/**
 * Store a maze of walls for use in a cat-and-mouse game.
 */
public interface Maze {
    // Access size of the maze
    int getWidth();
    int getNumRows();

    // Access the elements in the maze; true indicates a wall.
    boolean[][] getMaze();

    // Count number of cheese pieces the mouse has eaten.
    int getNumCheeseEaten();

    // Checks validity of the row index.
    // Returns 1 for valid, 0 for invalid.
    int checkValidRowIndex(int row);

    // Make the cell a wall, returning if it was already a wall.
    boolean makeWall(int x, int y);
}
```

Improvement name
(use each at most once)

Explain with an example in code of where to apply it

- a) _____: _____

- b) _____: _____

- c) _____: _____

- d) _____: _____

- e) _____: _____

7. An Alien is searching the Universe to find a good Planet. Complete the Universe and Alien classes below to help! Each Planet has a maximum temperature, and each Alien has a maximum temperature he/she likes.
- Planet and PlanetRater are shown on page 7 (which is removable).
 - Include Java annotations as appropriate.
 - You need not write any comments or import statements.
 - **Use intention revealing names.**
- a) [8] Create a well-named new function in the Universe class which finds the “best” planet. Your method must accept one parameter, a PlanetRater object, and returns a Planet which is rated the highest by the PlanetRater. If there are no planets in the Universe, return null. **You must use a for-each loop.**
- Hint: Start by looking at the PlanetRater interface (page 7).*

a) **[8]** Create a well-named new function in the `Universe` class which finds the “best” planet. Your method must accept one parameter, a `PlanetRater` object, and returns a `Planet` which is rated the highest by the `PlanetRater`. If there are no planets in the `Universe`, return `null`.

Hint: Start by looking at the PlanetRater interface (page 7).

```
public class Universe {
    private List<Planet> planets = new ArrayList<>();

    // Create a new Universe, reading it in from the file fileName
    public static Universe makeFromFile(String fileName) {
        // CODE OMITTED; don't write it, assume it's here and works.
    }
}
```

}

- c) [2] State just the name of the design pattern for...
- the method `Universe.makeFromFile(...)`:

- finding the best `Planet` with a `PlanetRater`:

- d) [5] Complete the UML class diagram below, as relating to the code in the rest of this question.
- Draw any anonymous classes (name them “Anon”).
 - Draw all **class relationships** and **decorations** (such as `<< ... >>`) for the classes and interfaces shown.
 - Do *not* show any fields or methods.



Detachable Information Page

You may remove this page but it must be turned in with your exam.

```
/**
 * Store info about a planet.
 */
public class Planet {

    private String name;
    private double maxTempC;

    public Planet(String name, double maxTempC) {...}

    public String getName() {...}
    public double getMaxTempC() {...}
}
```

Code 1: Planet class skeleton (method implementation omitted).

```
/**
 * Rate how desirable a planet is.
 */
public interface PlanetRater {

    // Returns the score for a Planet:
    // larger numbers are more desirable (better).
    double ratePlanet(Planet planet);

}
```

Code 2: PlanetRater interface for rating the desirability of a planet.