ECM1410:Object-oriented programming

CityRescue Coursework (Java / Maven)
Student Instructions

This repository contains starter code only for the CityRescue coursework. You must implement the full system described in the coursework specification PDF.

What you are given

- A Maven project skeleton (pom.xml, src/main/java, src/test/java).
- The public interface CityRescue with all required method signatures.
- Required enums and exception classes.
- A skeleton implementation CityRescueImpl (methods currently throw UnsupportedOperationException).
- A small set of public JUnit tests (not exhaustive).

What you must implement

You are expected to implement all behaviour described in the specification, including (but not limited to):

- A complete working implementation of CityRescueImpl that satisfies the CityRescue interface contract.
- All required domain classes and data structures (e.g., map, stations, units, incidents, etc.).
- Correct state management across operations (statuses, assignments, timers, capacity rules).
- Deterministic simulation/tick behaviour as defined in the spec (movement rules, ordering/tie-breaking rules).
- Exact output formatting where required (spacing, punctuation, ordering).
- Correct exception behaviour (including cases where state must remain unchanged if an exception is thrown).

The starter code intentionally does not include core model classes or solution logic — you must design an appropriate object model.

Project requirements

- Java: 17
- Build tool: Maven

- Testing: JUnit 5

How to run

Option A: Using Maven (recommended). Run these commands from the project root:

```
mvn clean test
```

```
mvn clean compile
```

Option B: Using an IDE (IntelliJ IDEA / Eclipse).

- Open the folder as a Maven project.
- Set the project SDK/JDK to Java 17.
- Run tests via the IDE's JUnit runner or by invoking Maven goals.

Where to write code

- Implement your solution in src/main/java/...
- CityRescueImpl is where you implement the required interface.
- You may create as many additional classes as you want, organised into sensible packages.

Rules: do NOT change the public API

To ensure your solution can be marked automatically:

- Do not change method names, signatures, or required exceptions in the CityRescue interface.
- Do not rename CityRescueImpl.
- Do not change the package names of provided files (unless the specification explicitly permits it).
- Do not delete or modify the provided enums or exception classes.
- You may add your own tests; do not remove the public tests.

If you break the public API, your submission may fail to compile during marking.

Testing guidance

- Public tests are only a small sample; passing them does not guarantee full marks.
- Write your own tests for edge cases, exceptions, ordering/tie-breaking, formatting, and tick sequencing.
- Test both normal scenarios and failure scenarios (invalid operations).

Output formatting

Several methods require exact output formatting. Follow the specification precisely. Small differences (extra spaces, missing commas, wrong ordering) may cause automated tests to fail.

- Prefer building a single formatting helper for units/incidents/status so output is consistent.
- Ensure lists are ordered exactly as specified.
- Include required prefixes (e.g., IDs) and fields exactly as stated in the spec.

Submission checklist

1. Run mvn test and ensure all your tests and the public tests pass.
2. Confirm you did not modify the required public API.
3. Check your output formatting against the specification examples.
4. Verify exception behaviour matches the specification (including no unintended state changes).
5. Ensure your code compiles under Java 17.

Academic integrity

- Do not share your solution code with others.
- Do not post your coursework code publicly (GitHub, forums, etc.).
- You may discuss the specification concepts, but do not share implementable solution code.

Help

If something is unclear or seems inconsistent, feel free to contact **a.chatterjee5@exeter.ac.uk**