Task 1: Deliverables (group)

This section is about the quality of your code and submitted documentation.

Task 1.1: Java code (group)

Task 1.1.1: Functionality (group)

This section covers the quality of the code you submitted. For full marks here, your implementation will need to be robust and stable. It must also meets specification.

Moon map and rocket	Not Done Partial Complete
Yugo Maxx, water pistol, and boss battle	Not Done Partial Complete
Oxygen dispenser	Not Done Partial Complete
Spacesuit and oxygen depletion	Not Done Partial Complete
Game ending	Not Done Partial Complete
Mark for task 1.1.1 : out of 8.0	

Task 1.1.2: Style and readability (group)

All new public and protected methods in the game package should be documented using Javadoc -- code supplied by us does not need to be documented. Javadoc should be informative and buildable.

Variable, attribute, and method names are well-chosen	Poor Okay Good
Adherence to Java style guide	Poor Okay Good
Use of Javadoc	Poor Okay Good
Quality of comments	Poor Okay Good N/A

Readability (Layout, use of whitespace indentation etc.)	Poor Okay Good

Mark for task 1.1.2 : out of 3.0

Mark for task 1.1 : out of 11.0

Task 1.2: Supporting documentation (group)

Design documents are required for the new requirements. If your design for existing parts of the system changed during implementation, e.g. because you needed to refactor, you should have kept its documentation up to date. If not, there was no need to change existing documentation.

Design documentation matches code	Poor Okay Good
Documentation is readable	Poor Okay Good
Notation used is correct	Poor Okay Good
Rationales provided for any refactorings	Poor Okay Good N/A
Spelling, grammar etc. of written documentation is correct	Poor Okay Good

Mark for task 1.2 : out of 4.0

Mark for task 1 : out of 15.0

Task 2: Design quality (group)

Task 2.1: Design of new system components (group)

This section is about the design as embodied in the code rather than the design as written in the documentation.

Repeated code avoided	Poor Okay Good
Use of encapsulation (e.g. implementation hiding, use of private attributes)	Poor Okay Good

Quality of abstractions (classes contain related data and functionality)	Poor Okay Good
Public interfaces are well-designed for new classes/packages	Poor Okay Good
Any refactorings after Assignment 2 improve quality of design	Poor Okay Good N/A

Mark for task 2.1 : out of 4.0

Task 2.2: Integration with existing system (group)

The implementation makes appropriate use of the existing framework of Actors, Items, Actions, etc.

Engine code is unchanged (i.e. classes in edu.monash.fit2099.engine)	Poor Okay Good
New client code uses engine classes well	Poor Okay Good
New client code uses existing game classes well	Poor Okay Good

Mark for task 2.2 : out of 4.0

Mark for task 2 : out of 8.0

Task 3: Use of Git and GitLab (group)

Both partners used git	No Yes
Frequency of commits	Just one A few Many
Quality of commit comments	None Poor Okay Good

Mark for task 3 : out of 2.0

Task 4: Recommendations for change to the game engine (group)

Recommendations must be based on genuine weaknesses of the current design - not simply "I didn't like it".

Problems perceived are clearly described	Poor Okay Good N/A
Proposed changes are clearly described, using UML if appropriate	Poor Okay Good N/A
Proposed changes solve the stated problem	Poor Okay Good N/A
Advantages and disadvantages of proposed changes are considered and explained	Poor Okay Good N/A
Perceived problems and proposed solutions are for the game engine, not specific to the scenario	Poor Okay Good N/A
Spelling, grammar, presentation	Poor Okay Good N/A

Mark for task 4 : out of 5.0

5: Bonus marks (bonus)

Bonus marks available for groups that come up with ideas for new features for the game, and then design and implement them. These need to have been approved beforehand by the Chief Examiner.

Mark for task 5 : out of 3.0