Changes to the existing code

Previously in assignment 2, after the player calls for the BuildRocketAction class, they were able to create a rocket which is a furniture on the map. However, the rocket did not have any functionality so the player could not fly to the Moon. We have introduced some changes to our code to add the feature of flying to the moon by creating the RocketToTheMoon class. In the BuildRocketAction class, we have refactored the createRocket method to instantiate a new RocketToTheMoon object.

Furthermore, before this, we created the List of Strings representing the Earth map in the Application class. We have refactored this to create separate classes for the GameMap objects which are EarthMap and MoonMap. The List of Strings for each map are stored in their own classes respectively. The EarthMap and MoonMap both have static methods to get their maps. The EarthMap and MoonMap classes are instantiated in the Application class.
Reasons why we refactored our code:

1. By putting the maps in their own classes, we are obeying the single responsibility principle where the classes are only responsible for their own maps. It is easier for future maintenance.
2. There are classes like RocketToTheMoon and RocketToEarth that use the instance of the maps.

Now that we have two maps in the game (representing the Earth and the Moon), we noticed that enemies and minibosses that are not on the same map as the player can attack the player. This is because, previously in our Distance class, the isAdjacent method compares the distance between the actors and the player irrespective if they're on the same map. The isAdjacent method has been refactored to take this into account. So, now the enemies (Goon and Grunt) and minibosses (Dr Maybe and YugoMaxx) cannot the attack the player unless they're adjacent to the player and they're on the same map. Furthermore, the StunBehaviour class has been refactored to compare the maps between the player and ninja. If they're on the same map, only the ninjas will have the ability to stun the player.

For the safety system, the GamePlayer class has been refactored. The player has an oxygen point counter in their class and if the counter has decremented to 0, the playTurn method adds the Action RocketToEarth.getAllowableActions() so that the player will be transported back to Earth.

In order for the player to quit the game, a new Action class called QuitGameAction has been created. The GamePlayer's class's playTurn method has been refactored to add the QuitGameAction into the list of possible actions for the player to perform.

There are new skills added to the GameSkills enum to identify the oxygen tank, spacesuit, Yugo's unconcious body and state of the water pistol.