Design changes and improvements

Listed below in the table are the changes and improvements made to our design. It is further explained below.

Classes	Changes made
Doors and keys	
UnlockDoorAction	- Overridden execute method, the Item key is removed after door is unlocked
LockedDoor	No changes
New types of enemies	
Enemy	- List of behaviours getter method suggested removed. - Added a GamePlayer attribute. - playTurn method uses Distance's isAdjacent method to check if the player is adjacent to the enemy. If true, returns AttackAction. Else, returns SkipTurnAction. Does not use a random Action in actions.
Grunt	- Introduced new final static constants (GRUNT_DAMAGE and GRUNT_HITPOINTS) used for damage and hitPoints.
Goon	- Uses Grunt.GRUNT_DAMAGE - Proposed overridden playTurn method and calling its superclass's getter method for a List of behaviours have been removed
FollowBehaviour	- Removed the distance method - Uses Distance class's static distance method to get the distance between the actor and its target
ShoutInsultBehaviour	- getAction method modified to have a 10% chance to return an Action that shouts an insult
Ninja	No changes
StunBehaviour	- No longer inherits from AttackAction. It inherits from Action as it does not damage the player. - Checks if there is a wall between the actor and the player. If there is a wall even though the distance between them is less than or equals to 5, do not stun. - Removed instanceof GamePlayer check. - In the execute method, the return String when the Ninja stuns the player has been changed. - Uses the Distance's class static distance method to get the distance between the actor and the player.
GamePlayer	No changes
Counter	No changes
MaxCounter	No changes
Distance	- Newly added class.

	- Has static methods; distance and isAdjacent
Q	
Q	- Added an overridden playTurn method to wander around the map. - Added an overridden getAllowableActions method to enable player to talk and give plans.
Doctor Maybe	
DrMaybe	- Added an overridden playTurn method. Calls Distance class's static isAdjacent method to check if its adjacent to the player. If it is, return AttackAction. Else return SkipTurnAction. - Uses Grunt's static variables to get half of Grunt's damage and hitPoints.
Building a rocket	
Rocket Pad	- In Assignment 1, it checked player's inventory if it has rocket body and rocket engine. Now, check if rocket body and rocket engine and placed on the rocket pad location.
BuildRocketAction	- In the execute method, it removes rocket body and rocket pad. - Do not add rocket into the player's inventory. Now, it replaces the rocket pad with Item newFurniture, rocket.

Design changes and improvements

Doors and keys

Class UnlockDoorAction

Design improvement:

In the overridden execute method, the Item key is removed from the player's inventory after the door has been unlocked.

Justification:

This feature has been added so that the player cannot use the same key item to unlock another locked door.

New types of enemies

Class Enemy

Design change:

1. The proposed getter method in Assignment 1 to return the List of behaviours has been removed.

2. Added a GamePlayer attribute called subject.

3. The playTurn method in the Enemy class has been modified so that it iterates through the List of behaviours and returns the first Action of behaviour that is not null. If there are no behaviours in the List or all behaviours are null, it call Distance class's static isAdjacent method to check if the player is adjacent to it. If yes, it will return a newly instantiated AttackAction to attack the player. Else, it will return a newly instantiated SkipTurnAction.

Justification:

1. The getter method proposed in Assignment 1 is unnecessary because the Enemy's subclasses use their superclass's playTurn method. The Enemy's subclasses do not need to get the List of their own behaviours as the behaviour of their classes are controlled by the playTurn method in the superclass. This reduces unnecessary codes for easy maintenance.

2. GamePlayer object is passed through its constructor. This means that the subject is a GamePlayer object and the enemies can only attack the player.

3. The playTurn method returns an Action for the enemy to perform based on its behaviours. It either returns an AttackAction or SkipTurnAction if there are no behaviours in the List or all behaviours are null. We have a made a design to not return a random Action from actions to perform as this requires us to check if the Action is an instance of DropItemAction or PickUpItemAction. Using instanceof does not apply polymorphism in our code which might lead to low maintainability. We avoid using instanceof because it is a code smell.

Class Grunt

Design improvement:

Introduced new final static constants (GRUNT_DAMAGE and GRUNT_HITPOINTS) used for Grunt's damage and hitPoints.

Justification:

Since the damage of Goon is double relative to the damage of Grunt and the damage and hitPoints of Doctor Maybe are half relative to the damage and hitPoints of Grunt, we have introduced static constants.

1. Static constants are used so that Goon and DrMaybe do not have to instantiate a new Grunt object. The constants are accessible through the class and not through an object of the class.
2. It is also declared as final to prevent wrongful assignation.

The Goon and DrMaybe classes use the static constants in their classes. This prevents magic numbers from appearing in our code. It also avoids excessive use of literals and prevents hunting for every place it occurs if there are future changes to our codes.

<u>Class Goon</u>
Design improvement:
1. To get its own damage, it uses Grunt.GRUNT_DAMAGE to get twice of Grunt's damage.
2. The proposed overridden playTurn method in Assignment 1 has been removed and the Goon class does not have to call the getter method for a List of behaviours in its superclass.

Justification:
1. Since Goon's damage is twice of Grunt's, it uses Grunt's static constant. This prevents excessive use of literals and magic number occurences. Furthermore, this improve our code maintainability, as Goon's damage will automatically be updated if changes are made to Grunt's damage.
2. The proposed design with the overridden playTurn method in Assignment 1 has been removed as the Goon class inherits and uses its superclass's playTurn method to implement the behaviour of its class. Hence, it is unnecessary to override its superclass's method. It also does not have to call the getter method in its superclass for a List of behaviours. This implements the DRY concept as it reduces duplicated code.

<u>Class Distance</u>
```
public class Distance
```

This is a newly added class. The Distance class is responsible for finding the distance between two actors based on their locations. It also checks if the actors are adjacent to each other.

How it works?
There are two methods added.
1. A static distance method
   Returns the distance between the actors based on their locations.
2. A static isAdjacent method
   Returns a boolean indicating if the actors are adjacent by calling its own static distance method.

Design rationale:
The methods have been declared as static so that the methods are accessible through the class and not through an object of the class. Hence a new Distance class does not have to be instantiated every time we need to find the distance between two actors.

Why has this new class been added?
We realised that there are 4 different classes (FollowBehaviour, StunBehaviour, DrMaybe, Enemy) that rely on the distance between the actor and the player. To prevent duplicated codes from 4 similar distance methods in different classes, we created a Distance class. This uses the DRY principle and makes maintenance easier if there are any future changes.

Class FollowBehaviour

Design improvements:

The distance method has been removed and it now uses the Distance class's static distance method to get the distance between the actor and its target.

Justification:

Refer to Distance class.

Class ShoutInsultBehaviour

Design change:

The getAction method has been modified so that there is a 10% chance it'll return an Action in its own class that shouts an insult. If the random probability is less than or equals to 0.10, it will return an Action that shouts an insults, else, it will return null.

Justification:

Our proposed design in Assignment 1 does not work as it always returns an Action that shouts insults. The ShoutInsultBehaviour class should be responsible for its own properties of implementing the 10% chance of shouting an insult.

Class StunBehaviour

```
public class StunBehaviour extends Action implements ActionFactory
```

Design changes:

1. The StunBehaviour class no longer inherits from AttackAction.
2. In the getAction method, it checks if there is a wall between the actor and the subject. From our understanding, the wall does not have to be on the same line as the Ninja and the player for the Ninja to not attack. For example, in these cases, the Ninja will not attack.

```
. n . . .              . n . . .
. . .#@        or      #####
. . . . . .            #. .. .@
```

3. Removed instanceof GamePlayer check.
4. In the execute method, if the stun action is successful, the GamePlayer's setPlayerStunned method is called and it returns a String that the Ninja stuns the player with a stun powder bag.
5. Uses the Distance class's static distance method to get the distance between the Ninja and player.

Justification:

1. It inherits from Action as it does not damage the player.
2. This checking has been added to prevent the Ninja from throwing the stun powder bag to the player when there is a terrain that blocks thrown objects (ie: Wall, LockedDoor objects) between the player and ninja because the stun powder bag cannot go through.
3. Using instanceof is a code smell and does not apply polymorphism in our code.
4. In the previous assignment, we returned a String with the damage inflicted but since stun action does not damage the player, we have changed the return String to the correct statement.
5. Refer to Distance class.

Q

Class Q

Design improvements:

1. Added an overridden playTurn method.
2. Added an overridden getAllowableActions method.

Justification:

1. This method allows Q to implement its behaviour which is wander behaviour. Hence, Q can wander around the map on every turn.
2. The getAllowableActions method is overridden to talk and give plans. The GivePlansAction class is called when the GamePlayer has rocket plans in its inventory. In order to talk, it calls the TalkAction class.

Doctor Maybe

Class DrMaybe

Design improvements:

1. Added an overridden playTurn method. It uses the Distance class's static isAdjacent method to check if its adjacent to the player. If it is, return AttackAction. Else return SkipTurnAction.
2. To get its own damage and hitPoints, it uses Grunt.GRUNT_DAMAGE and Grunt.GRUNT_HITPOINTS to get half of Grunt's damage and hitPoints.

Justification:

1. By overriding the playTurn method, DrMaybe will not move and will only attack if it's adjacent to the player.
2. Since DrMaybe's damage and hitPoints are half of Grunt's, it uses Grunt's static constants. This prevents excessive use of literals and magic number cases which improve our code maintainability, as DrMaybe's damage and hitPoints will automatically be updated if changes are made to Grunt's damage.

Building a rocket

Class RocketPad

Design Change:

In the previous design, it checks if the player's inventory has items with the skills GameSkills.BUILDROCKETBASE and GameSkills.BUILDROCKETTOP to call the BuildRocketAction class. The newly designed allowableAction method will check if rocketBody item and rocketEngine item are placed on the rocketPad. If yes, it adds BuildRocketAction into actions.

Justification:

Our design has been changed because we have found that our initial design does not work as the player is not required to place the rocket body and rocket engine on the rocket pad before building the rocket. Furthermore, in our initial design, we proposed that the player is allowed to build the rocket while standing on the rocket pad. We discovered that this approach does not work as the BuildRocketAction description is only printed in the menu when the player is adjacent to the rocket pad.

Class BuildRocketAction

Improvements added:

1. The execute method removes rocketBody and rocketEngine from the rocketPad.
2. In the previous assignment, we treated rocket as an Item object that can be added into the player’s inventory. Now, an improvement has been made in the execute method whereby the rocket pad terrain will be replaced with a new Item furniture called rocket when the action is executed.

Justification :

1. These improvements were added in conjunction with the changes made in RocketPad class. The rocketEngine and rocketBody should be removed from the rocketPad once the rocket is created. This is to prevent GamePlayer from creating multiple rockets from the same set of rocketEngine and rocketBody.
2. We realised that the rocket is not an item that should be carried in the player’s inventory so we classify rocket as a furniture that appears on the map. Hence, when building the rocket, the rocket pad will be replaced by a rocket to show the player that it has been successfully built.