New class created in starwars.entities.actor package

1) Class name: AdmiralAckbar

Create a new character called Admiral Ackbar, who works at the headquarters of the Rebel Alliance which is located on the moon Yavin IV. He needs to stay still at the headquarters and has a 10% chance of saying "It's a trap!" Thus, talking behaviour is added to the list of behaviours for that actor.

2) Class name: DarthVader

Create a new character called Darth Vader. A lightsaber is assigned to him and 3 behaviours are added. We have created a TortureLukeBehaviour() and ChokeBehaviour(). The ChokeBehaviour() is similar to attackNeighbourBehaviour, but instead of attacking, we force choke any actors in the same coordinate as Darth Vader. We could have reused the attackNeighbourBehaviour class for Darth Vader but some complicated conditions and changes has to be added to the code to accommodate the ChokeBehaviour class. Our aim is to make the class as simple and easy to be understand by the user. Thus, we created a new behaviour. The probability shouldn't have been hard coded, but instead a parameter could be passed in like what we did for the attackNeighbourBehaviour class. But since for this assignment, only Darth Vader is using the ChokeBehaviour, it's okay to hard code?

3) Class name: MonMothma

Create a new character called MonMothma. In the act() method, we looped through the whole Yavin IV map grid to check for Luke's presence. Once Luke arrives at Yavin IV, MonMothma will say appropriate line depending on who follows Luke to Yavin IV. If both Princess Leia and R2-D2 arrives at Yavin IV together with Luke then the game will end and the player has won. This is done by accessing the SWWorld setNotRun method which would set the run attribute in SWWorld as false and the program will stop running after another round.

4) Class name: PrincessLeia

Create a new character called Princess Leia. We decided to set Princess Leia's hitpoints as 600 because at 600 maxhitpoint, Princess Leia will have a force level of 6. This indicates that Princess Leia can resist mind control. Thus, she cannot be mind control. This also means that Princess Leia can also attack with a light saber because she is not weak minded.

When Princess Leia meets Luke at the same coordinate, a follow behaviour will be added to her. From then onwards, this class will execute the behaviour. This behaviour will only be added to Princess Leia if and only if Luke is in the same location as Princess Leia. That's what the Boolean meet is used for.

5) Class name: Stormtrooper

Create a new character called Stormtrooper. A blaster is given to every Storm Trooper. 3 behaviours are added to this class. A CallBackUp behaviour class is created for stormTrooper because we thought it's not appropriate to associate the properties of a stormtrooper in the attackNeighbour class. AttackNeighbour behaviour class should be purely used to attack neighbour and not to call back up.

6) Class name: SWRobots

7) Class name: SWOrganicActor

SWRobots class is created based on SWDroidActor class in Spike’s sample solution for assignment 2. We thought it was a good idea to breakdown SWActor into more categories such as SWDroidActor and SWOrganicActor. However, we thought the VanillaActors class is a little redundant as we don’t really understand the reason behind the existence of that class thus we did not implement that class.

8) Class name: SWRobotsInterface

This class is also based on the SWDroidActorInterface class in the sample solution.

New class created in starwars.entities.actor.behaviour package

1) Class name: AttackNeighboursBehaviour
2) Class name: BehaviourInterface
3) Class name: FollowBehaviour
4) Class name: PatrolBehaviour
5) Class name: RepairBehaviour
6) Class name: TalkingBehaviour
7) Class name: TrainerBehaviour
8) Class name: WanderAround

These are the list of behaviours adapted from Spike's sample solution for assignment 2. Small changes have been made so that the code is compatible with our previous code for assignment 2.

9) Class name: CallBackUp

This behaviour class is created mainly for stormtroopers to call backups. It's a very simple class, whereby all it does is at 5% chance it calls the MoreStormTrooper action class which will create a new stormTrooper actor at the same location. This class is created because we thought it's not appropriate to associate the properties of a stormtrooper in the attackNeighbour class. AttackNeighbour behaviour class should be purely used to attack neighbour and not to call back up.

10) Class name: ChokeBehaviour

This behaviour class is created mainly for Darth Vader to choke actors at the same location at 50% chance. This behaviours class is similar to the attackNeighbourBehaviour class. The only difference is that some conditions are added and the ForceChoke action is called instead of the attack action.

*further explanation please refer to Darth Vader class documentation.

11) Class name: TortureLukeBehaviour

This behaviour class is created to because there is a list of things that Darth Vader would do when he meets Luke at the same location. Thus, we thought it would be a good idea to put all those conditions in a behaviour class instead. In this class, Darth Vader will attempt to Luke to the dark side at a 50% chance. This can be done by changing Luke's team to EVIL in the BetrayTeam action class. In that class, another condition is set. If Luke has been trained, it's force ability level will increase and there will be 75% chance that Luke can resist that attempt and his team will remain as GOOD. Darth Vader will have another 50% chance to attack Luke when he is not attempting to turn Luke to the dark side.

New class created in starwars.entities package

1) Class name: Falcon

This is the entity that could bring actors from one world to another, basically it’s to allow actors to travel from one map to another according to their choice. The Falcon entity class has 3 affordances – FlyToDeath, FlyToYavin, FlyToMain. When the Flacon is created on each map, the affordance to travel to its’ own map is removed. This is done in the SWWorld under the initializeMap method.

New class created in starwars.actions package

1) Class name: BetrayTeam

This action class is to change actor’s team. The target and the team desired to be changed to is passed in as parameters. This is to fulfil the specification of Darth Vader turning Luke to the dark side. Luke will be passed in as the target and team.EVIL will also be passed in. Then we will change that actor’s team by using the setTeam() method in SWActor. We have decided to set the condition that this class can only be executed by actors with force level of 9 and above.

2) Class name: FlyToDeath
3) Class name: FlyToMain
4) Class name: FlyToYavinIV

We have decided to create 3 flying class instead of putting all of them in one because for an entity to have 2 command choice, it has to have 2 affordances. For example, an actor can be both mind controlled and attacked, 2 commands will appear in the enter command for player to choose, this is because that actor has those 2 affordances. Thus, we decided to have 3 action classes each fly to one of the three maps. The falcon entity will have all three affordances. When initiating the falcon on the map, we removed the affordance for flying to its’ own map.

In the act() method for all these 3 classes, we would use the SetMyGrid(String) method to change the map to be displayed, then place the actor and whoever following the actor in the new map at the location (0,0) where the other falcon on the other map will be at.

5) Class name: ForceChoke

This action class is similar to attack action class. The difference is that the takeDamage is hard coded to be 50 hitpoints and we are not checking if the target is of the same team as the actor or not. This ForceChoke action will not cause any damage on the actor itself that is executing the action. Only actors with a force level of above 8 can execute this action.

6) Class name: MoreStormTrooper

This class is designed for stormTroopers to create more stormTroopers at the same location. Every time this action is executed, another new stormTrooper actor will be created in the same location as the stormTrooper that called this action.

Significant edits done in the starwars package

1) Class name: Application

In this class, we have changed the condition of the while loop. Originally, the condition was while(true), now we have changed it to while(test) whereby test is a Boolean gotten from the SWWorld class. If that test == false, the game will end and the program will print out an end game String also gotten from the SWWorld class.

2) Class name: SWWorld

There are many new methods created in this class, but there are a few significant methods added that has to be explained. The getRun() and setNotRun(), is the getter and setter methods for the Boolean run which determines whether is it game over or not. Once the setNotRun() method is called, the game will end within the next few rounds, because it takes at least one round for the program to check the condition. The setEndGame(String) and getENDGame() is to pass Strings to the Application class to be printed after the game ends. This Strings differs and it depends on why the game end. For example, when Princess Leia dies, the program will stop running and prints “Game Over! Princess Leia has died!”, but when Luke dies, the program will stop running and prints “Game Over! Luke has died!”.

The getGrid() and SetMyGrid(String) methods are the setter and getter for the myGrid attributes. This is used set the map to be displayed when the player is travelling from one world to another.

3) Class name: SWGrid

We added new integer parameters x and y in the constructor so that the height and width of the map won’t be hard coded anymore. The getMapName() method is also added because each map will be assigned a name so that we could identify which map is which.

Modification done in the engine code under the edu.monash.fit2099.simultor.space package

1) Class name: World

2 new methods have been added to this class. A Boolean attribute “run” is added which determines whether the game ends or not. The getEndGame() method returns the Boolean value while the setDontrun() method set the boolean value to false. Both of this methods are called in the SWWorld class.