Dilara Mukasheva K23008889

Shipwreck Adventure

– Synopsis of the game -

The game has an underwater concept, in which the aim is to find the Lost Pearl Necklace. The player, who is on the search for the Necklace, has to explore the ship, find/ earn items along the way, meet other characters, who either give or take something from the player, and maintain an oxygen level above 0 to ensure no loss. To win the game you will need to pick up the necklace.

Rooms: *The Deck*, *The Grand Hall*, *The Drawing Room*, *The Game Room*, *The Closet*, *The Dining Room*, *The Grand Suite*, *The Maid Quarters*, *and The Chamber Room* **(9 rooms)**
Characters: *Octopus*, *Oyster*, *Skeleton*, *Shark* **(4 characters)**
Items: *algae*, *fish*, *torch*, *flippers*, *key*, *necklace* **(6 items)**
The item *algae* is considered as non-expendable; this means that when the player uses it, it is placed back to its original position to be picked up and used again. Each item has a weight that adds to the total bag weight.

There are also some interactive features such as a multiplication game that is played in *The Game Room*, or the transporter room (*The Closet*) which transports you into a randomly selected room on the ship. *The Closet* is the only room through which you can enter *The Grand Suite*, where you can find the *Shark*, and receive the *key* to *The Chamber Room*. *The Maid Quarters* and *The Chamber Room*, both require objects (*flippers / key*) to enter it.

- Base task implementations –

The game has at least 6 locations/rooms

Using the Room class, I instantiated and initialised 9 rooms (for full list look back to the synopsis). I've connected the rooms using the setExit() method and also assigned the starting positions of characters and items in each room in the Room constructor. This means that I could get a list of all current occupants and contents of each room throughout the game.

There are items in some rooms. Every room can hold any number of items. Some items can be picked up by the player, others can't

Using the Item class, I instantiated and initialised 6 items (for full list look back to the synopsis). All items can be picked up at some point in the game, nonetheless, the *torch* and the *key* must be earned, they can't just be picked up using the 'pick' command at first. This is specified within the Game class in the processCommand() method.

The player can win. There has to be some situation that is recognised as the end of the game where the player is informed that they have won. Furthermore, the player has to visit at least two rooms to win

The player wins the second the necklace is placed into the bag. This is seen in the play() method in the Game class. At least two rooms will definitely be visited before *The Chamber Room*.

Implement a command back that takes you back to the last room you've been in. The back command should keep track of every move made, allowing the player to eventually return to it's starting room

Using a stack named as previousRooms in the Game class, the program keeps track of the rooms that the player has entered. Using push() when entering a new room, and pop() when going back to the previous room. The stack will always contain *The Deck*, as it is always the starting point of the game.

Add at least three new commands (in addition to those that are present in the base code); The back command will not count as a new command.

New commands: check, drop, pick, give and play (which can only be processed when in *The Game Room*) **(5 new commands)**

Check command can have different second words: weight, room (current items and characters in the game), oxygen, bag (items in the bag), map (layout of rooms). All commands found in the CommandWord and Game class. Implemented using the methods check(), pickItem(), dropItem(), giveItem(), playGame() in Game class.

- Challenge task implementations -

Add at least three characters to your game. Characters are people or animals or monsters – anything that moves, really. Characters are also in rooms (like the player and the items). Unlike items, characters can move around by themselves

Using the Character class, I instantiated and initialised 4 characters. Characters automatically move around the ship, when the player moves to a different room. This is done by allocating a new room to the characters using the move() method in Character class. Check allowedroomsCharacters ArrayList which rooms characters are allowed to be in. *Octopus* (steals all items in bag, if no *torch* present), *Oyster* (increases oxygen level by 10, if fed *algae*), *Skeleton* (with every encounter decreases oxygen level by 5 every time). The *Shark* character is bound to *The Suite*, therefore isn't considered as a 'full character', even though it interacts with the user using the 'give' command.

Extend the parser to recognise three-word commands. You could, for example, have a command give bread dwarf to give some bread (which you are carrying) to the dwarf

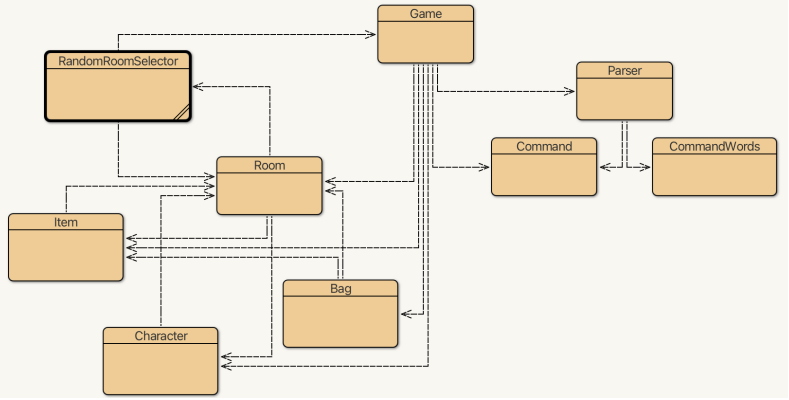
I extended the parser by adding a third word to the class - constructor, getCommand() method. This is most often used in the giveItem() method in Game class, as it requires 3

words. For example: 'give *oyster algae*' or 'give *shark fish*'. The other command methods only use 1-2 word commands, and reject 3 word commands.

Add a magic transporter room – every time you enter it you are transported to a random room in your game

The magic transporter room is *The Closet*. I've created the RandomRoomSelector class which selects a room from an allowed room list, and generates a random number from 0 - allowedRoomList.size() and finds the room at that index position. This is then used to change the current room from *The Closet* to the room randomly generated.

- Code quality considerations –



Coupling and Cohesion

From the class diagram you can see the loose coupling between the 'right hand side' and the 'left hand side'. This shows there are no dependencies among them, meaning that they operate independent of one another. There is high cohesion in the 'right hand side', however this was necessary as Room, Item, Character and Bag are very dependent on one another in the game. If I would have redone the project, I would use more classes and split up the work that each class does and make the cohesion looser, and decrease coupling even more.

Responsibility - driven design

I believe that I've considered a responsibly driven design when creating different classes. This means that each class takes care entirely of a certain aspect of the program. For example, I've ensured that the Bag and Item class are entirely separate, to ensure that each 'specialises' on its own thing, and does it well.

Maintainability

I used variables to denote constants such as oxygen, maximum weight, names of rooms. This means that if in the future, I would need to change the maximum weight from 8 to 10, I would only need to change it in one place (in the variable maxWeight), instead of returning back to every place I have quoted it to be 8. This saves time, as well as ensures no human error on my part.

- Walk through the game (step-by-step) –

Throughout the game ensure that oxygen level is always above 0, go back to the dining room, pick algae and feed it to the oyster using the 'give oyster algae'. You can do that endlessly. There is also an element of luck when encountering an oyster, so seize it! Skeleton will decrease it by 5 all the time; keep an eye out using the 'check oxygen' command.

go east (in *The Grand Hall*)
go south (in *The Game Room*)
play
win the torch by answering the question; automatically placed into the bag
now you can add any item to the bag and ensure that octopus won't steal
back (in *The Grand Hall*)
back (leads you back to *The Deck*)
pick *fish*
go east (in *The Grand Hall*)
go east (in *The Dining Room*)
pick *algae*
*when encounters *oyster*, now can use *algae* increase your oxygen level*
back (in *The Grand Hall*)
go north (in *The Drawing Room*)
pick *flippers*
Following combination is used until payer reaches *The Grand Suite*:
    go east (randomly allocated the room)
    back (in *The Drawing Room*)
*in *The Grand Suite**
give shark fish
if weight above 5, drop items to make it no more than 5
*automatically places *key* into the bag*
go north (in *The Dining Room*)
go east (in *The Maid Quarters*)
*player stays in room because of the *flippers**
go east (in *The Chamber Room*)
*player can enter room because of the *key**
pick *necklace*
if weight above 5, drop items to make it no more than 5
WIN is displayed