Author: Gervais Pierre.

## Theme: Salvaging of derelict ships in space.

The idea of the theme is that you are a solo space salvager that goes to different derelict ships scattered across the endless void of space. You are accompanied by your ship's Ai which will manage some functions for you while you go exploring (i.e., playing the game).

Your goal is to find as many valuable objects as possible. They will be scattered around the derelict and will add up in "credits". The credits will make up your high score for the game.

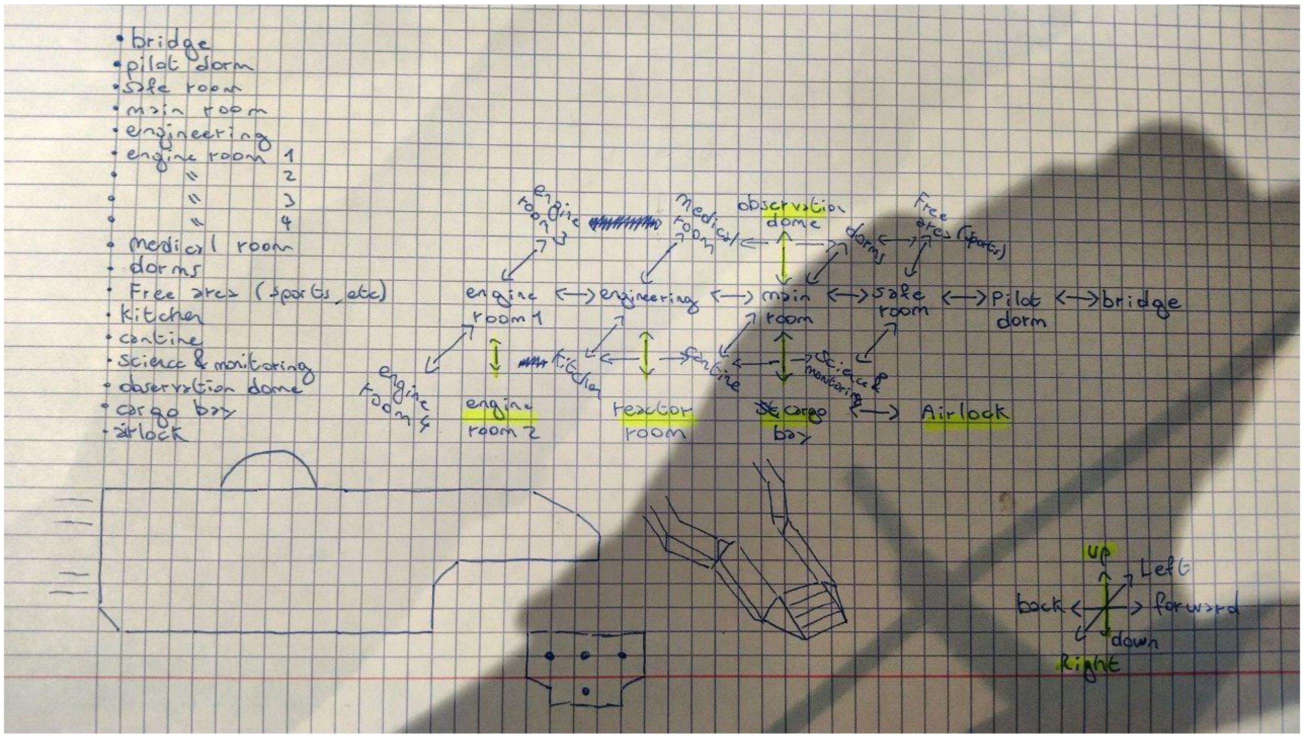
Your character will be limited in inventory space by volume (in L).

In order to "finish" the game the player will have a limited time mechanic that will follow the turn-based actions you can take. In other words, each action you take will reduce your timer until it reaches zero and you will be "returned to the ship", the game ends.

The timer will be life support level in %.

Some rooms could have differences and bring about some dangers that could make you lose the game (die) or gain some benefits like extra time or credits.

## Layout of the game world.



## Gameplay information.

You are a space runner that goes salvaging in the great void of space. You find a derelict and your job is to find anything of value that you can later sell on for credits. Though beware derelicts can contain many dangers as well as benefits.

The player will then go from room to room starting in an Airlock room exploring the ship and looking for anything of value that will help them get more credits.

Once the player runs out of time they will return to their ship (the game ends) and the total credits they acquired will be their final score.
If the player dies on the other hand it will also end the game cutting their score to 0 though it'll still show how much you've earned until the player's untimely demise.

Whenever the player finds an item it will either give credits or more time for them to explore the derelict.

## Rooms, Characters, Items.

Rooms: Airlock, Cargo Bay, Main Hall, Observation Dome, Safe Room, Pilot Dorms, Bridge, Sports Room, Dorms, Medical Room, Engineering, Cafeteria, Kitching, Science & Monitoring, Reactor Room, Engine Room 1, Engine Room 2, Engine Room 3, Engine Room 4.
Characters: Ship AI, The Player.
Items: Digi Pad, Damaged Crate, Intact Crate, the rest is TBD. (Way to recharge life support and one to increase inventory capacity).

## Win and Lose conditions.

Timer runs out and you are still alive (back at ship) ➔ Win
You return to the ship early ➔ Win (neutral)
You die while exploring ➔ Lose

You can measure yourself to others with your final credit score.

## Game challenges.

Limited inventory space for the player when carrying multiple items.

Future time limit (Life Support). (May have to return to ship before it runs out or die).

Potential future dangers that could lead to player death.

## To Do List.

- More potential story details.
- Items, Characters.
- File system for saving external data (like potential game save system and custom room layouts).

## Changes over time (top to bottom in chronological order).

- Initial.

- Added getExit() command to remove duplicate code left from initial project (TP3.x).
  printWelcome() and goRoom() now incorporate this function.

- Added "Above" & "Below" capabilities to the Room class.
  Added attribute "aBelow", "aAbove".
  Changed foRoom() to incorporate "Below" and "Above" features.
  Added "Below" and "Above" to getExit() return type String.
- Changed initial getExit() to printLocationInfo().
  Added "You are in : " + CurrentRoomDescription before Exit part. In : "
  Updated the calls for printLocationInfo() where it used to be called by getExit().

- Added getExitString() in Room class.
  Changed part of the code in printLocationInfo() since it would have used duplicate code from getExitString().

- Changed all direction attributes to a direction hashmap. (Instantiated the hashmap in room constructor).
  Changed setExits() to accommodate the change in attribute type.
  Added a getExit() to make exit data retrievable for other functions like : getExitString() and goRoom().

- Changed getExitString() to use keySet() from import.

- Added getLongDescription() in Room class.

  Changed printLocationInfo() to use getLongDescription() to avoid duplicate code.
- Changed "break;" in Game class in the goRoom() function to a "return;" in order to avoid sending two types of error text.
- Added "look" command that will give the player information on the room they are in (uses printLocationInfo()).

- Added “eat” as Use command (doesn’t do anything rn but will be used latter once inventory system is up.
- Added “show commands” & “show all” as GetCommandListString() in CommandWords then as “getValidCommands” in Parser to be then used in the “help” command. Also changed the Valid command words table to a hashmap and added a second compartment to the keywords with a small description of what the command does for later use if needed.
- Proper Rooms have been added base on the current design thus removing old test rooms.
- Changed the help command to be able to take a second word and return specific information related to said command.
- Changed Game class to Game Engine and Added mostly void UI class and Game class.
- Created basic GUI window to interface with.
- Finalized GUI class to be fully integrated into the code. (Terminal is now only debug information). Added Programmer Art pictures for all rooms too.
- Created Item class. Integrated Item class in Room class.
- Created Inventory class (Items), that can hold multiple items. Did not integrate yet into other classes.
- Added Test command and Back command.
- Created Player class and integrated Inventory class into Room and Player Class.
- Re-did all code to be more streamline with current classes and to remove duplicate code in some areas.
- Added Take and Drop commands. Player can now pick up an item from a room and drop it in another.

## Current Build.

Classes consist of:

- Game: Initial setup between classes for them to work together.
- UserInterfaceController: GUI through which the player will interact with the game. (Contains: Text Log, Text Field, Enter Button and Image Display).
- GameEngine: Central class managing all data related to the game. (Commands, piping data between Classes, etc. . .).
- Player: Class storing and managing all data related to the player.
- Room: Class storing and managing all data related to rooms.
- Inventory: Class storing and managing all data related to inventories.
- Item: Class storing and managing all data related to items.
- Parser: Class managing text inputs and parsing it into command objects.
- Command: Class managing all command data.
- CommandWords: Class checking if Text information fed to Parser is valid or not.

## Used Resources.

Currently all code and ideas are made by me or used with current Zuul exercises (Zuul Book). And by reading Javadoc documentation from the Oracle website.