SE 4810 Project

In this project you are going to model and implement the world described below by using Object-Oriented Programming

The Game of Planet Trade

Planet trade is a turn-based multiplayer game where each player is playing as space merchant while travelling between the planets of a galaxy. The following items are describing the various features of the game universe and are supposed to be covered by your model.

General Objects	
R-1	A blackhole is something that explodes and creates a galaxy
R-2	A galaxy contains a set of planets that are certain distance away (in terms of lightyear) from each other.
R-3	At each planet there is market where some supply of commodities are bought and sold
R-4	Each supply of commodity has (different at each market) : • Current supply amount, • Unit buy price • Unit sell price
R-5	A commodity has a name, unit volume and decay ratio.
R-6	A player has: • name • current money • spaceship • current planet
R-7	A Planet has: • Name • Market (described at Item-3) • Unit fuel price • Spaceship parking price per turn
R-8	A spaceship has : • name, • buy price • a list of cargo, • capacity in terms of volume, • speed in terms of light year per turn, • current fuel • fuel capacity • fuel usage per lightyear
R-9	A cargo contains commodity and the quantity of the commodity
At the beginning of the game	
R-10	A blackhole explodes and a galaxy is created randomly
R-11	A list of commodities with arbitrary names and properties generated.
R-12	A list of spaceships is crated randomly by a spaceship factory
R-13	Each player is placed at a random planet in the galaxy

R-14	Each player is asked to choose and buy a spacehip from an initial list of spacehips which decreases the current money by the buy price of the chosen spaceship
R-15	At each planet a random market containing random supply of the commodities are generated by a market generator

At each turn player can do the following actions

R-16	Buy new market items from the market of the currentplanet as much as the capacity of the spaceship and the supply of the market allows. The buy operation causes the current money drop with the amount calculated by unit buy price of the market item in the market and the amount
R-17	Sell any cargo in the spaceship. The sell operation causes increase in the current money with amount calculated by the cargo amount and unit sell price of the commodity in the market.
R-18	Buy fuel as much as the fuel capacity of the spaceship allows. It causes the current money drop with the amount calculated by the unit fuel price at the current planet
R-19	Plan journey to another planet. If this is done in one turn the player will be at the target planet in the next turn if the spaceship has sufficient amount of fuel which is calculated by the fuel usage of the spaceship and the distance between the planets. Otherwise the player stays at the same planet causing a drop in the current money by the parking price of the current planet.

The game performs the following at each turn

R-20	At each turn player actions are validated by the game and if they are valid necessary updates are performed accordingly which includes: • the current money of each player • current planet of each player
R-21	The current cargo of each player are decayed. (the amount of each commodity in the cargo are reduced by decay ratio of that commodity) Amount = Amount*(1-Decay Ratio)
R-22	At each market of each planet the following changes randomly • unit buy price and unit sell price of each market item (with relatively small amount) • current supply of each market item (with relatively small amount

Various game features

R-23	The players have read-only access to the following information during the game at each turn • Their current money • Their current spaceship (all attributes) • The distances between each planet • The market information of each planet (Market items etc.)
R-24	Each player starts the game with an initial amount of money and no spaceship

See the attached project.zip . Extract **project** folder and copy it into the src folder of the Java project. You are supposed to implement your code starting from the initial set of classes and interfaces provided to you as **gameengine** package Your code should be compatible with the interfaces and classes in the **gameengine** package. You are not allowed to change the game engine package. Implement your code under the package "**planettrade**". Your Planet Trade Game class must be implementing **Game** interface so that it can be run by **TurnBasedGameEngine**.

You are expected to apply the principles and mechanisms that we have learned in the lecture throughout the semester. Try to achieve this as much as you can.

Submit the following:

- The source code as a package (folder) named as planettrade.
- A project report that contains at least a table that includes the following columns
    - Object ID: The id of the principle (or topic) covered in the lecture(Item-XX) or the id of the requirement item listed above(R-XX)
    - Source Code Location: The name of the source file/class/interface/method that matches the principle or the requirement item.

    You will not get any credit for anything that is not listed in the table. So even if you submit a perfect implementation, without a project report with the corresponding table, it will get 0 grade, since it is not possible to evaluate your implementation.
    .

In the table there should be at least **15** principles and **20** requirement items listed to be considered as successful. The submissions with less number of coverage are graded partially.

There may be a short presentation of your work towards the end of the semester. In that case, the exact date and the expected content of the presentation will be announced later.

Optional Bonus Items:

1. A user-friendly **GameRenderer** instance that displays the game information during the game (It can be a fancy, colored console display or a Window Frame using Swing , JavaFX etc.)
   Depending on the level of the renderer you can get up **50** pts as bonus.

2. An intelligent **Player** instance that plays the game with some smart heuristics.
   Depending on the level of the player you can get up to **50** pts. as bonus.

3. A cool game feature not mentioned above.
   Up to **30** points as bonus

The implemented bonus items (if any) should also be listed on your project report with their related class, interface or method information.