Algorithms 3

Project – Strategy Game

Team 2: Nour Eldin CHARIFH, Tiago Goncalves Da Silva, Yinghan LING

Town Center (Tiago Goncalves Da Silva):

Ignore TownCenterPart but should be similar

Food resource:

- Food is an integer variable
- We have a method for gathering food (for workers) which is private
- We have a method build Workers which checks if we have 25 food unit, to create one worker

Build Workers:

- We have an Array list of objects with [int: **ID**, Boolean: **availably**, String: **task**], referring to each element (worker) uniquely
- Garnering food method can be triggered from the Town Center part
- Gathering wood method can be triggered from the workshop part

Unit limit:

- A Boolean method to control the maximum number of units that can exist (the condition of upgrade)
- Another method to check the number of units taken by any players.

Upgrade Town Center:

- We have upgrade method which will turn a Boolean global variable from false to true.
- Town center can run upgrade method, but should get access into wood and gold variables in the other parts.

Status:

We have status method to display info about unit limit, current food, available workers, active workers.

Workshop (Yinghan LING):

Wood resource:

- Wood is a class with an array of integers that stores the number of chestnut, birch, and pine respectively.

Gather wood:

- We have the gather wood method which takes woodDemand value as a parameter.
- Workers is retrieved from Town Center.

Build houses:

- We have a method build house which reduce the wood value by 40, and then for each house we build, our limit of units (which is in Town Center) increases by 5.

- Once the Town Center request the number of the Houses, it will send the String transferred from the integer of counter of Houses.

Status:

By inputting 3, the player can see the program printing the status, including the current woods resources, active workers gathering wood and the number of houses.

Barracks (Nour Eldin CHARIFH):

Build Archers:

- Barracks is a class which has a method to build archers.
- We're going to have a variable called archers (initially 0) which accumulate all archers from all barracks.
- We have the build archer method which will request access into the other parts (Town Center & Workshop) in order to use variables and arrays (food, wood, workers) as parameters.
- We have a method which returns what we retrieved if the other resources are not available. We use it after if condition to check if all resources are available.
- NOTE: build archer method will consume (decrease) food and wood values but workers value will be back after building the archer.

Gold resource:

- Gold should be a variable of the barracks class which increases (when invoking "plunder" method) by (5,25, or 50) depending to its type

Plunder treasures:

- We have "plunder" method which runs automatically (randomly every 5 to 8 seconds) and takes archers as a parameter.

- In the body of plunder method, we increase Monuments and/or gold values,applying probability of (25% 75%) respectively.
- Monuments is a variable which will be increased probably after running the plundermethod.

NOTE: an archer can't find monuments before one half upgrade.

- Within the method body we apply probability of losing the archer (50% beforeupgrade, 25% after upgrade)
- A method to inform the Town Center about the unit limit after each plunder methodrun.

Upgrade archers:

- A method in barracks class to upgrade its archers which takes an archer as a parameter and reduce the risk of dying from 50% to 25% (can be applied once foreach archer)
- This method will require access into (Town Center and Workshop) to retrieve data(100 units of wood and 50 units of food)

Status:

- We have status method to display info about Gold, active archers, Monuments.

==

Concurrency:

Town Center: In the town center we have the workers which we are going to use Collections<Workers> ArrayList and the class Workers consist of "public record Worker (Stringtask, …)"

Workshop && Barracks: In the workshop and barracks we have the wood/Gold which we are going to use Collections<Wood>/Collection<Gold> ArrayList and the class Wood/Gold consist of "public record Gold (String kindOfGold, int amount)"/

"Public record Wood (String kindOfWood, int amount)"

Communication:

The three of us decided to use java, so we are also going to use javelin to connect our codeto one chosen port. We will use the HTTP operation (GET, POST, PUT, DELETE) for the methods that request any concurrency of the sub-parts.

Project – Strategy Game–Revised

Concurrency:

For the concurrency of the three parts, we used Array Lists to store each resource. Each resources had a record class.

Communication:

For the communication from the three sub-parts, we only used GET methods because of some difficulties understanding the other methods. In order to make the RESTful API get request synch we did not use the ex() method in client request. Our GET methods used a path for example /food/>ClientValue<. The ClientValue indicated the amount of food that the client wanted to use, meaning that the town center must subtract the amount of the “food” variable. If the food variable has enough food, it returns true as a response and subtracts the food, otherwise it must return false as a response. This was also used for the other resources.
This method of getting resources was also used to return the asked value by a different path but with the same logic. We used clientBuilder to communicate and we specify the target from the server’s path, we request the data to be JSON.

In general:

The three sub-parts have some differences and do not really follow some aspects of the description above because we considered that we did not have to create the methods and variables that were listed on the description.