Danny Nguyen, 6334502, dn17hg

COSC 3P91 Assignment 2

Documentation on the program.

How to play/run the program:

- Run the program
- Player will be in the non-fighting mode first. Player has 1 gold mine, iron mine, lumber mill, 100 gold, iron and lumber at the start.
- Player then can build buildings and army and such by typing in from 1 to 9. They can also upgrade buildings and army/defense system.
- Player can collect resources but they must first employ a worker corresponding to each mine/lumber mill (gold miner for gold mine, iron miner for iron mine, …)
- If player decide to fight someone (only bots in this state), he/she can press 1 to change to fight mode. They can change back to non-fight mode by press 1 again.
- When in fight mode, player can choose to see, attack, filter or generates new opponents by type in from 1-4.
- The maximum level a building or army can get is 10.
- Each village can only have 1 village hall and farm and map. Both village hall and map can be upgraded but map is not.
- Press 10 to exit.
- When perform an action, player has to wait for 1-3 seconds (can be changes in the future). During this wait time, player cannot perform any other actions. The time is arbitrary and short for the sake of debugging.

How the program is written:

- Most of the program derived from the suggested uml diagram of assignment 1, with some changes.
- Nothing in the GUI package since there isn't any gui needed for this state.
- Real time is only used when perform an action or when generating resources for mines and lumber mills. Mines and lumber mills will generate a certain amount of resources according to the time passed. So wait longer means more resources.
- A village has 2 score, health and damage of the army and health and damage of the defense system. In this single player phase, army will be used the most since there isn't any other player to attack the player. So the health/damage is the total health/damage of every units in the army. Same with army, the health/damage of defense system is summation of health/damage of all the buildings in the defense system.
- Success of an attack is calculated by reducing the health by opponent damage and who reach 0 or below first lose. Winner takes all the loots and their army take damages while loser takes half the loot and lose all of the army (has to rebuild).
- Loot can be calculated differently when multiplayer is introduced.
- Bots are generated to some what similar with the player. Which means the stronger player becomes, they have to face stronger bots.

Generic:

- In Print abstract class in Utility package.
- Used to print numbers such as resources or health/damage of a player so let it be generic so that we can reuse the code.

Local and Anonymous class:

- Local class: Time (line 76) in Player class in Game package. Since time is used only in when we need to generate resources, it is appropriate to let it be a local class.
- Anonymous class: printAll (line 137) method in CollectResource class in Game Engine package. Implement the Print abstract class. Since the print abstract class can be used to print other thins as well, we will implement the method according to what we need.

Lambda Expression and Method References:

- Lambda Expression: Can be found in many methods in totalArmy, totalDefense and CollectedResource class all in the GameEngine package. Methods such as getTotalHealth, getTotalDamage. Can also be found in other classes as well.
- Method References: Can be found in lots of classes. One example is in Bots class in GameEngine package. Method sortBots (line 113).

Exceptions:

- Can be found in Player class and other classes in GameEngine package. In Player class, exceptions can be found in methods such as FightingPhase (line161) where the exception is used to mainly check the user input. We require user input under the form as integer so exception can be used to ask the user over and over again if they give wrong type of input. Can also be found in other methods in Player class.

Java-Standard Utility classes (Collection):

- Collection: Can be found in Bots classed in GameEngine package. Appears in sortBots (line 113) method where we use the sort method of Collection to sort out the bots once they are made.

I/O(Uses of stream):

- Stream: can be founded in Bots classed in GameEngine package. Appears in filterBots (line 143) method, stream is used to filter all of the bots with certain health thresh hold and return the chosen bots as the result.