Overview documentation of 261200 Object Oriented Programming Project

Tools

- Git
- Intellij IDEA
- VitualVM
- JUnit
- Spring Framework

Design

Data Structures: LinkedList

```
Class Region{
        protected long i; //Collect index of row in the map.
        protected long j; //Collect index of column in the map.
        protected LinkedList<Region> connectReg; //List of Regions that connected with this
                                                    Region.
        protected boolean citycenter; //Collect this Region is City Center?
        protected Player president; //Collect Player that occupy this Region.
        protected double budget; //Budget that keep in this Region.
        protected double interest; //Collect interest rate of this Region.
        public static void computeInterest(Region r); //Calculate this budget when changing
                                                        turns.
        public double getBudget(); //Get value of budget in this Region.
        public void setBudget(double budget); //Set value of budget in this Region.
        public bool checkCitycentre(); //Checking this Region is City Center?
        public double getInterest();//Get value of interest rate in this Region.

}

Class Map{
        protected long row; //Collect number of rows in this Map
        protected long column; //Collect number of column  in this Map
        protected Region [ ][ ] Map; //Array that collect every region in this Map
        public static void computeConnect(Region r); //Calculate that region Which region is
                                                        it close to?
}
```

```
Interface Action {
        void done(); //
        void relocate(); // Relocates the city center to the current player region
        void move(long i, long j); // Moves the city crew to specified direction
        void invest(Region reg, long deposit); // Adds more deposits to the current region
        void collect(Region reg); // Retrieves deposits from the current region
        void shoot(Region reg); // Attack a region in the specified direction
}
```

Data Structures: LinkedList

```
Class Player implements Action{
        long  budget; //Player's budget
        Crew MyCrew;//Player's crew
        Region citycenter;//Player's citycenter
        LinkedList<Region> Myregion;//Regions are belonged by Player
}

Class Crew {
        Region  current;// Current position of city crew
        long opponent();// the location of the closest region belonging to an opponent
        long nearby();// looks for the opponent's region closest to the city crew
}

Interface Parser {
        Expr Plan() throws Exception;
}
Class ExprParse implements Parser{
        Parser Plan();
        Parser Statement();
        Parser Command();
        Parser AssignmentStatement();
        Parser ActionCommand();
        Parser MoveCommand();
        Parser RegionCommand();
        Parser AttackCommand();
        Parser Direction();
        Parser BlockStatement();
        Parser IfStatement();
        Parser WhileStatement();
        Parser Expression();
        Parser Term();
        Parser Factor();
        Parser Power();
        Parser InfoExpression();
}
```

Testing plan

Date	Activity
24 - 31 Jan 2024	• write tests before code
31 Jan - 7 Feb 2024	• test Region class • test Map class
7 - 21 Feb 2024	• test parser • test game function class
21 Feb - 6 Mar 2024	• test UI/UX interface • test server client

- Will your test cases have enough coverage of the input space?
  We will partition input space into subdomains
- Will your test cases have enough coverage of your code?
  every line of code should be run
  every path through the code should be taken
  every possible object type should be constructed

Work plan

Date	Activity
24 Jan - 7 Feb 2024	implement Map class | by Patiphan implement Region class | by Patiphan implement Player class | by Natan, Bannawat implement ExprParse class
7 - 21 Feb 2024	design and implement UX/UI
21 Feb - 6 Mar 2024	manage network system and server client and check overview of all program

- Identify any key dependencies between different implementation steps, where one team member cannot start an implementation task until another team member has completed their assigned task?

  For those who completed their own part first went in to help with the parts that weren't finished yet.

Team member

Natan Bunkerd	650610777
Bannawat Vongpooton	650610778
Patiphan Klinhom	650610781