CS5001 Object Oriented Modelling Design & Programming

Practical 2: OO Implementation

University of
St Andrews

Student ID: 170009479

Contents

The implementation of a Tower Defence game

1 Introduction

Today, there are many various types of computer games. Amongst, Tower Defence games, as a type of classic game, are still very popular. In this practical, I develop a simple Tower Defence game based on the provided UML class diagram (Game, Tower, Enemy, Catapult, Slingshot, Rat and Elephant). This report illustrates the process of developing this game that is consist of the design of basic class, the implementation of user interface enhancement and the result of this system.

2 Design basic class

2.1 Game class

The Game class include two method and a constructor. The first is main method, the second is advance methods.

In the main method, it implements steps including invoking BuyTowers class, installing towers, adding enemies, invoking advance method and invoking DisplayGame class. In advance method, it manipulate shot enemies and update the state of game. Specific steps are as follows:

(1) In main method

a. Invoke BuyTowers class

To realise the purpose of making a player buy towers in the beginning of the game. This BuyingTowers class is called by main method. (The practical ideas how make a player buy towers will be illustrated in a later section.)

b. Install towers

After buying towers, a player is asked to install towers. The result of installing towers is to configure every tower in specific position following the player's requirements. (The practical ideas how put ever tower in specific position according to the player's requirements is illustrated in "Design enhancement" part later.)

c. Add enemies

The emergence of enemies is automatic but not random. More specifically, enemies introduce every 3 game steps and include 3 rats and 2 elephants each time. (The practical ideas how make enemies occur automatically is illustrated in "Design enhancement" part later.)

d. Invoke advance method

The purpose of invoking advance method is to realise the update of the game state. In other word, it is mainly to update the position and the numbers of enemies. (The practical ideas how advance the state of the game will be illustrated in a later section.)

e. Invoke DisplayGame class

After invoking DisplayGame class, a simple terminal-based user interface that will allow a user to play this game is achieved. (The practical ideas display the interface is illustrated in "Design enhancement" part later.)

(2) In advance method

This method is another method in Game class. It needs a timeStep parameter to update the state of the game every game step. Advance method include making towers shot enemies in proper time. To achieve it, my idea is as following:

a. Check every tower (realised by looking at every elements of array list of towers) to get the result if the tower have loaded (realise by looking at the result of executing willFire method in Tower class)

b. If the tower have loaded , the first enemy will be shot by the tower(realised by hit method in Enemy class).
c. Check if the health of the first enemy is smaller or equal to 0. If it is, the first enemy is removed, the second enemy will become the first enemy.

In the process of shooting, One of main points is to give the sequence of hitting enemies. Here, every game step, the first enemy is shot until it die. After the first enemy die, the second enemy becomes the fist enemy.

2.2 Tower class

The Tower class is the super class of Catapult class and Slingshot class and include five methods (getDamage, getPostion, willFire, getName and toString) and a constructor. Specific steps are as follows:

(1) The getDamage method is to return the damage of the current tower.
(2) The getPosition method is to return the position of the current tower.
(3) The willFire method is to give the result whether fire the first enemy or not, which is realised by if sentence.
    a. If the tower have loaded, the result is that the tower fire.
    b. If the tower have not loaded, the result is that the tower cannot fire.

The condition of checking if the tower have loaded is that the number of game step is integral multiples of load frequency (load frequency is the number of how many game steps the current tower can load).

(4) The getName function is to store the type of towers (slingshot or catapult), which will display in game interface to indicate the type of different towers.
(5) The toString method is to override the default toString method to show a desirable format.

2.3 Enemy class

The Enemy class is the super class of Elephant class and Rat class and include six methods (getHealth, getPostion, hit, advance, getName and toString) and a constructor. Specific steps are as follows:

(1) The getHealth method is to return the health of the current enemy.

(2) The getPosition method is to return the position of the current enemy.

(3) The hit method is to calculate the health of the current enemy after being hit. The condition is the position of the current enemy is before the position of the current tower.

(4) The advance method is to update the position after the current enemy moving forward. Note that, since an elephant move one position every two game step, it can transfer to move 0.5 position every game step.

(5) The getName function is to store the type of enemies (elephant or rat), which will display in game interface to indicate the type of different enemies.

(6) The toString method is to override the default toString method to show a desirable format.

2.4 BuyTowers class

As it mentioned in the first section of the second part, the method of BuyTower is invoked in the main method of Game class. The practical ideas of reaching this purpose are as follows:

(1) Ask a player if need buy towers.

(2) If a player choose no, the process of buying tower finishes.

(3) If a player choose yes, a player is asked to choose a type of desired towers.

(4) Match the type of towers and enter the right loop. A player is asked to enter the number of towers he wants.

(5) After buy the tower, the number of coin is updated and showed in game interface.

(6) A player is asked if want to buy other towers.

(7) If a player choose no, the process of buying tower finishes.

(8) If a plyer choose yes, from (3) to (6) will ask again until a player choose no to express that they do not want to buy towers.

Note that, when a player input invalid answers, the corresponding question is asked again until a play input valid answer. The conditions of giving invalid answer include that the answer is out of options (for example giving answer" C" that is not included in options" A" and" B") and the number of coins is not enough.

2.5 Catapult class and Slingshot class

Both of these two classes are the subclasses of Enemy class. They inherent the methods of enemy class and have same parameters with enemy class. Both of these two classes are to initialize the health, position, movement and name respectively. Amongst them, movement means how many positions they move forward each game step.

2.6 Rat class and Elephant class

Both of these two classes are the subclasses of Tower class. They inherent the methods of Tower class and have same parameters with Tower class. Both of these two classes are to initialize the damage, position, load frequency and name respectively.

3 Design enhancement

3.1 Install towers in the main method of Game class

As it mentioned in the first section of the second part, the function of install towers should be concluded in the main method of Game class. The practical ideas of reaching this purpose are as follows:

(1) Manipulate two types of towers (slingshot or catapult) in separate for loop.

(2) In every loop, every tower position is asked. A player gives the number of position to every tower.

(3) Put towers to positions that a player requires.

3.2 Add enemies automatically in the main method of Game class

As it mentioned in the first section of the second part, the function of add enemies automatically should be concluded in the main method of Game class. The practical ideas of reaching this purpose are as follows:

(1) Manipulate two types of enemies (elephant or rat) in separate for loop.

(2) In adding rat loop, every rat is add until reaching the value of “numberOfRat”.

(3) In adding elephant loop, elephant enemy is add until reaching the value of “numberOfElephant”.

(4) Note that variable named numberOfRat is the number of rat needed to be added each game step; variable named numberOfElephant is the number of elephant needed to be added each game step.

3.3 DisplayGame class

As it mentioned in the first section of the second part, the method of DisplayGame is invoked in the main method of Game class. The practical ideas of reaching this purpose are as follows:

(1) Create 4 for loops in this class to achieve 4 different functions.

(2) The first loop is to store the position and the name of every tower.

(3) The second loop is to put every tower in desired position.

a. Firstly, estimate if the number of position of tower equals to the number of the current position in the corridor.

b. If it is no, add “#”(representing corridor). If it is yes, estimate the type of the current tower.

c. “s” is used to represent slingshot, and “c” is used to represent catapult.

d. Add the proper letter.

e. Continue the loop until reach the end of the corridor.

(4) The third loop is similar to (3), and the main difference is that this loop is to operate enemies.

(5) The fourth loop is to add “#” until reach the end of the corridor.

Note that:

(1) All towers are put in the first row.

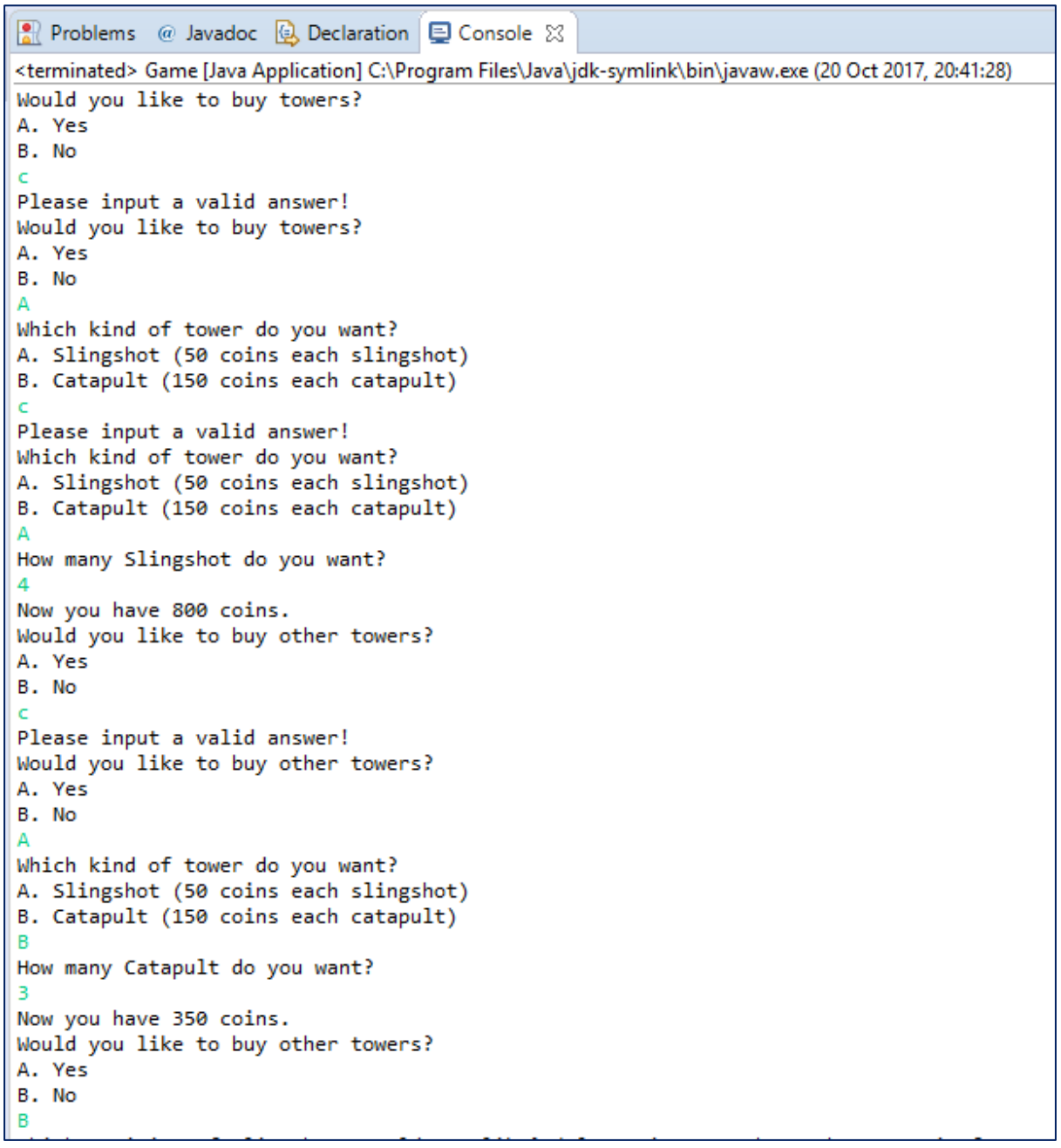
(2) When the positions of different tower are same, the letters (“s” or “c”) are overlaid, the game interface only show one letter.

(3) Even though there are some limitations as above, it can show written message to make the number of towers, the type of towers and the position of towers more clear.

4 Result

The result is shown as following by simulating the game:

(1) Ask a player to buy towers.



(2) Ask a player to configure the towers.

```
Which position of Slingshot0 would you like? (Please input number, the range is from 0 to 50
34
Which position of Slingshot1 would you like? (Please input number, the range is from 0 to 50
35
Which position of Slingshot2 would you like? (Please input number, the range is from 0 to 50
36
Which position of Slingshot3 would you like? (Please input number, the range is from 0 to 50
37
Which position of Catapult0 would you like? (Please input number, the range is from 0 to 50
38
Which position of Catapult1 would you like? (Please input number, the range is from 0 to 50
39
Which position of Catapult2 would you like? (Please input number, the range is from 0 to 50
40
```

(3) Run the game.

```
Time Step: 0
Tower(Slingshot): position: 34, damage:1
Tower(Slingshot): position: 35, damage:1
Tower(Slingshot): position: 36, damage:1
Tower(Slingshot): position: 37, damage:1
Tower(Catapult): position: 38, damage:5
Tower(Catapult): position: 39, damage:5
Tower(Catapult): position: 40, damage:5
########################################SSSSCCC#########
R~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
R~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
R~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
E~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
E~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##################################################

Time Step: 1
Tower(Slingshot): position: 34, damage:1
Tower(Slingshot): position: 35, damage:1
Tower(Slingshot): position: 36, damage:1
Tower(Slingshot): position: 37, damage:1
Tower(Catapult): position: 38, damage:5
Tower(Catapult): position: 39, damage:5
Tower(Catapult): position: 40, damage:5
########################################SSSSCCC#########
E~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##################################################

Time Step: 2
Tower(Slingshot): position: 34, damage:1
Tower(Slingshot): position: 35, damage:1
Tower(Slingshot): position: 36, damage:1
Tower(Slingshot): position: 37, damage:1
Tower(Catapult): position: 38, damage:5
Tower(Catapult): position: 39, damage:5
Tower(Catapult): position: 40, damage:5
########################################SSSSCCC#########
~E~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##################################################
```

```
Time Step: 3
Tower(Slingshot): position: 34, damage:1
Tower(Slingshot): position: 35, damage:1
Tower(Slingshot): position: 36, damage:1
Tower(Slingshot): position: 37, damage:1
Tower(Catapult): position: 38, damage:5
Tower(Catapult): position: 39, damage:5
Tower(Catapult): position: 40, damage:5
#####################################SSSSCCC#########
R~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
R~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
R~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
E~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
E~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##################################################

Time Step: 4
Tower(Slingshot): position: 34, damage:1
Tower(Slingshot): position: 35, damage:1
Tower(Slingshot): position: 36, damage:1
Tower(Slingshot): position: 37, damage:1
Tower(Catapult): position: 38, damage:5
Tower(Catapult): position: 39, damage:5
Tower(Catapult): position: 40, damage:5
#####################################SSSSCCC#########
E~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##################################################

Time Step: 5
Tower(Slingshot): position: 34, damage:1
Tower(Slingshot): position: 35, damage:1
Tower(Slingshot): position: 36, damage:1
Tower(Slingshot): position: 37, damage:1
Tower(Catapult): position: 38, damage:5
Tower(Catapult): position: 39, damage:5
Tower(Catapult): position: 40, damage:5
#####################################SSSSCCC#########
~E~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##################################################
```

(4) The result of the game.

```
Time Step: 27
Tower(Slingshot): position: 34, damage:1
Tower(Slingshot): position: 35, damage:1
Tower(Slingshot): position: 36, damage:1
Tower(Slingshot): position: 37, damage:1
Tower(Catapult): position: 38, damage:5
Tower(Catapult): position: 39, damage:5
Tower(Catapult): position: 40, damage:5
#####################################SSSSCCC#########
R~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
R~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
R~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
E~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
E~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#####################################################

Time Step: 28
Tower(Slingshot): position: 34, damage:1
Tower(Slingshot): position: 35, damage:1
Tower(Slingshot): position: 36, damage:1
Tower(Slingshot): position: 37, damage:1
Tower(Catapult): position: 38, damage:5
Tower(Catapult): position: 39, damage:5
Tower(Catapult): position: 40, damage:5
#####################################SSSSCCC#########
E~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#####################################################

Time Step: 29
Tower(Slingshot): position: 34, damage:1
Tower(Slingshot): position: 35, damage:1
Tower(Slingshot): position: 36, damage:1
Tower(Slingshot): position: 37, damage:1
Tower(Catapult): position: 38, damage:5
Tower(Catapult): position: 39, damage:5
Tower(Catapult): position: 40, damage:5
#####################################SSSSCCC#########
~E~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#####################################################
```