Report of CS5001-Practical 2 -OO Implementation

Contents

Introduction

This report will describe the design and implementation of CS5001-Practical 2 – OO Implementation. This practical aims to establish a tower defence game in Java. The overall structure is provided by *CS5001-O2.pdf*. Thus, this report will focus on part that excludes given information.

Design

This java program has 10 java files in total. Figure 1 shows the flowchart of the process of the game. Class Game is the main class for running the game. Class Tower and Class Enemy are two super classes. Extended from Class Tower, three subclasses are generated for instantiation. Extended from Class Enemy, four subclasses are for instantiation.

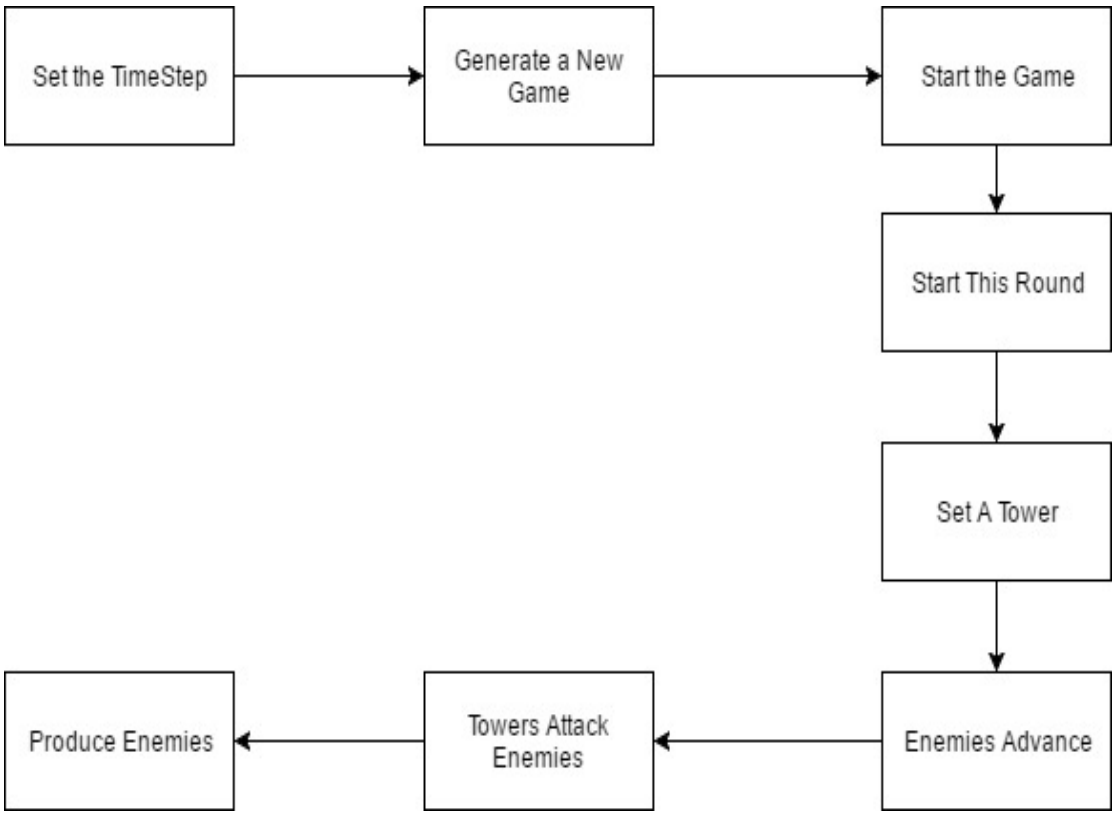

Figure 1 The Flowchart of method deployment

Overall, in this game, the player needs to set length of the corridor. The length should be an integer between 10 and 100. The player should protect himself by preventing enemies going through the corridor. The round of one game is equal the length of the corridor. Once the player survives till the round ends, he can win the game. If any live enemy goes through the corridor successfully, he will lose the game. Every round, the player should set one tower, then enemies start to do some actions (e.g. be produced, advancing). After that, all existing towers attack enemies who are in the front of the corridor one by one. Next, the system will report results and start next round. A losing case with three rounds of the game is shown in Appendix A.

In terms of towers, they will be placed from the right side of the corridor one by one. One position can only have one tower. Once one tower is set, it will attack one enemy instantly. Table 1 shows the information of different towers. Tower MoneyTower is an extension of the practical. This tower can earn two points into budgets every round once it is set. However, the damage of this kind of tower is low. The player should balance the profits and damage when setting it.

Tower Name	Costs	Damage	Attacking Frequency
Catapult	7	5	Every three rounds
Slingshot	2	1	Every round
MoneyTower	5	2	Every round

Table 1 Towers in the game

In terms of enemies, for one enemy, producing it will cost one round. Namely, in the next round of the production of this enemy, it can proceed. Table 2 presents information about enemies. Enemy Boss and Enemy Theft are extensions. Enemy Boss has high HP and Kill Bonus. Enemy Theft has low HP while every time it advances, one point of the players' budgets will be reduced (e.g. if there are five Enemy Thefts alive, in this round they advance one position, 5 points of the budgets will be reduced).

Enemy Name	Kill Bonus	Speed	Healthy Point (HP)	Time of Birth
Elephant	10	1 position/2 rounds	10	Randomly
Rat	1	2 positions/1 round	1	Every two rounds
Boss	30	1 position/1 round	15	A certain round[1]
Theft	8	3 positions/1 round	1	Every four rounds

Table 2 Enemies in the game

[1] The formula is Time of Birth = The Length of Corridor / 3 * 2.

Implementation

To implement this design, the order of running methods and manipulation ArrayList by play a vital role. Figure 2 presents the overall structure of this program. It is also the deployment of methods. ArrayList < Enemy > is used to store live enemies and delete dead enemies. ArrayList < Tower > is responsible to store set towers. Object Enemy and Object Tower are used to manipulate individual objects.

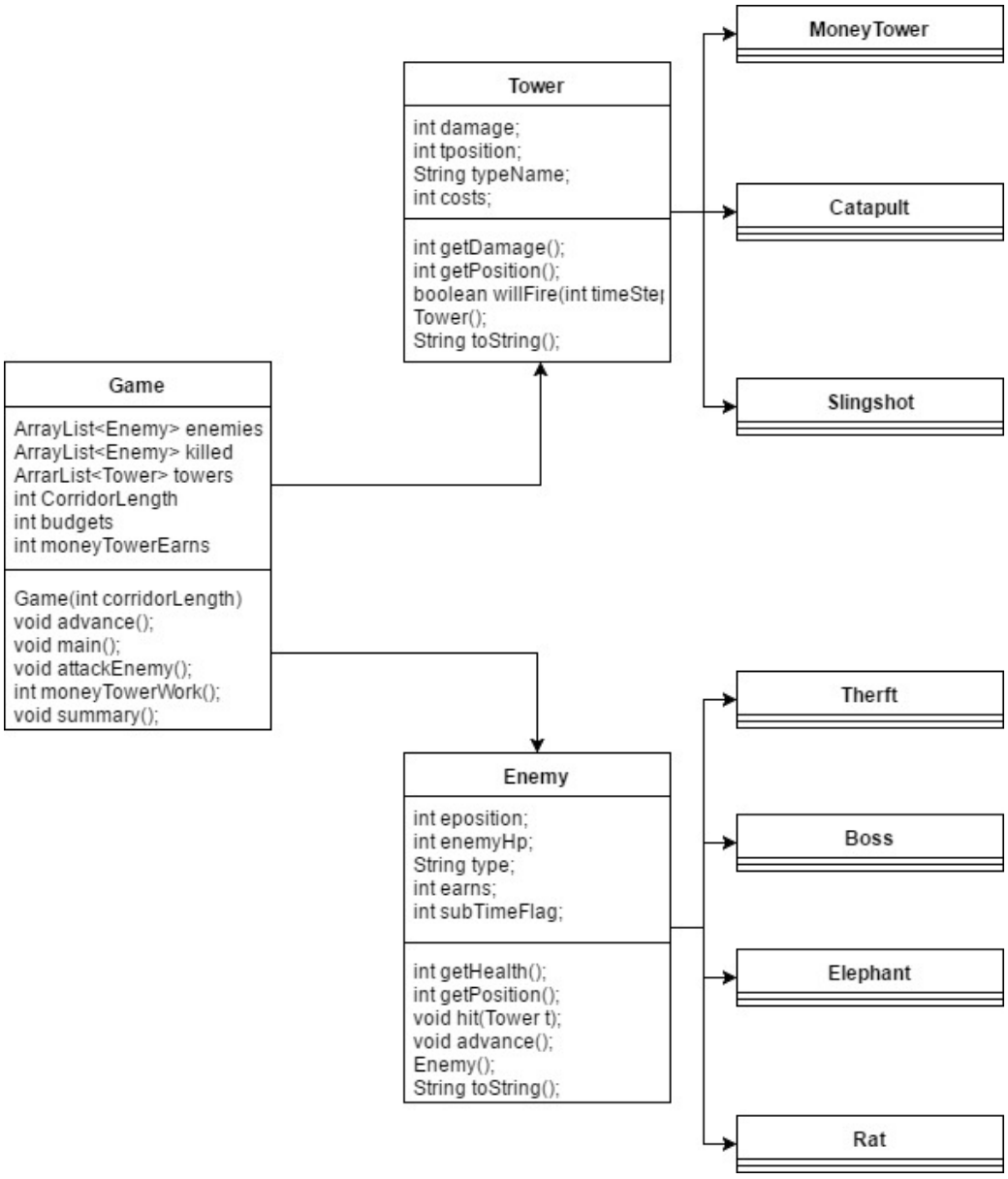

Figure 2 The UML of the Java Program[2]

[2] int earns is the kill bonus of an enemy. int subTimeFlag is the flag to make an enemy stay in position 0 when it is produced.

Firstly, void main() starts the program. Object Game calls void advance() to produce a new game. In method void advance() , several Object Tower are created and stored into ArrayList < Tower >. Object Enemy is manipulated with ArrayList < Enemy >. For code reuse, void attackEnemy() is written to call void hit(Tower t) in Class Enemy in proper conditions (e.g. If one enemy has beyond one tower, the tower cannot attack this enemy anymore. This condition is written in void attackEnemy()).

Because Tower MoneyTower is a special case which will manipulate private int budgets, int moneyTowerWork() is not in Class Tower which is mainly used for inheritance. int moneyTowerWork() is written in Class Game to make itself easy to modify private int budgets.

To interact with a player in a simple way, String toString() is overridden in Class Enemy and Class Tower. Similarly, void summary() is called to inform results of one game to a player.

```
@Override
public String toString() {
  return ("Type: " + this.typeName + " Damage: " + this.damage + " Position: " + this.tposition + " ** ");
}
```

Results:

[Type: Catapult Damage: 5 Position: 5 ** , Type: Catapult Damage: 5 Position: 4 **]

```
@Override
public String toString() {
  return ("Type: " + this.type + " HP: " + enemyHp + " Position: " + eposition + " ** ");
}
```

Results:

[Type: Rat HP: 1 Position: 2 ** , Type: Theft HP: 1 Position: 0 ** , Type: Boss HP: 15 Position: 0 **]

Results and Discussion

Without a GUI, the game can interact with players in a relatively simple and clear text way. Roughly, there is no obvious logic or programming bugs as a simple game.

However, the balance of the game is not very reasonable. For corridors which are longer than 20, wins and loses become balanced. That is, the shorter the corridor is, the less possibility the player can win (e.g. If the length of a corridor is 3, as showed in Appendix A, it is not possible for a player to win the game. Thus, I restrict the length must be longer than 10). Meanwhile, if one corridor is longer than 70, it is almost impossible for a player to lose this game. The reason is that the feature of Tower MoneyTower is not well-balanced. The more rounds a game has, the easier the player can earn money to set Tower Catapult, which has the highest damage. Thus, the chance to win the game becomes impractical large.

Conclusion and Future Work

In conclusion, the game is an available but very simple frame. It can work but still need to be improved a lot, especially on the gaming balance. If possible, GUIs should be added in the future to make a good human-computer interaction.

Appendix A

A losing case with three rounds

There are three kinds of Towers: MoneyTower, Catapult, and Slingshot

There are four kinds of Enemies: Boss, Theft, Elephant, and Rat

Your Initial Budget is 15

The method of setting a tower: <TowerType>

Ready? Go!

****
****
****
****

Round: 1

Current budgets: 15

Tower:

[]

Set Your Tower: 0 Not Set 1 Catapult 2 Slingshot 3 Money Tower

1

Before the Attack, Current Enemies: (Position 0 means the Enemy has not entered the CorridorLength)

[Type: Rat HP: 1 Position: 0 **]

Killed Enemies in this round: []

After the Attack, Current Enemies (Position 0 means the Enemy has not entered the CorridorLength):

[Type: Rat HP: 1 Position: 0 **]

****
****

Round: 2

Current budgets: 8

Tower:

[Type: Catapult Damage: 5 Position: 3 **]

Set Your Tower: 0 Not Set 1 Catapult 2 Slingshot 3 Money Tower

1

BOSS IS APPROACHING!!!

Before the Attack, Current Enemies: (Position 0 means the Enemy has not entered the CorridorLength)

[Type: Rat HP: 1 Position: 2 ** , Type: Theft HP: 1 Position: 0 ** , Type: Boss HP: 15 Position: 0 **]

Killed Enemies in this round: []

After the Attack, Current Enemies (Position 0 means the Enemy has not entered the CorridorLength):

[Type: Rat HP: 1 Position: 2 ** , Type: Theft HP: 1 Position: 0 ** , Type: Boss HP: 15 Position: 0 **]

****

Round: 3

Current budgets: 0

Tower:

[Type: Catapult Damage: 5 Position: 3 ** , Type: Catapult Damage: 5 Position: 2 **]

Set Your Tower: 0 Not Set 1 Catapult 2 Slingshot 3 Money Tower

1

Sorry, you can not set a tower with budgets 0

Sorry! You Lose! Try Again?

Summary:

Your Money: 0

Kill Enemies: 0

Set Towers: 2

Earns from MoneyTower: 0