

EEE 102 C++ Programming and Software Engineering II

Assessment 3 SDP Report

**Student: Kai-Yu Lu
ID: 1614649
Date: 2018.5.9**

Catalogue:

Exercise1.....

1. Problem statement.....	3
2. Analysis.....	4
3. Design.....	5
3.1 Flowchart.....	5
3.2 Design Step.....	6
4. Implementation.....	7
5. Testing.....	7

1. Problem statement:

This assignment requires students to use C++ codes to build a game called Monopoly. As for this program, it also requires student to design at least three classes. In this game, there are two players, which are the user and computer. Two paragraphs below will describe the rules of this game.

At the beginning, 10000 would be added to two players' accounts respectively. It is therefore students need to build a data base to record every player's information such as name, gender and account balance. In addition, the program should have the ability to calculate and track the changes of balance during the process of this game. There is a map shown in Figure 1, which consists of 80 uniform squares. The price of each square is a random number whose range is from 10 to 500. When the player crosses the left bottom corner which represents start point, 200 will be added in the account. The user and the computer will take turns to roll a dice (the num is from 1 to 6), which decides how far player could go. Player can decide whether to buy or the invest the square that just arrived. If the player arrived the square or the adjacent squares that are bought by opponents, this play will be punished. The punishment rule is:

1. The fine will be increased to 10% if one of the adjacent squares is bought by the opponent.
2. The fine will be increased to 15% if one of the adjacent squares are bought by the opponent, for example, you are in the mid of two squares bought by your opponents, you will get 15% punishment the price of the current square.
3. The fine will be increased to 20% if more than three adjacent squares are

bought by the opponent. (20% is the top fine rate.)

Player could invest the square which has been bought, of which the price is the half of the current price. In addition, the fine to opponent would be increased to 5% when the player choose to invest.

When one of the account balances of the players is less than 0, that player need to declare bankruptcy. Finally, the game will end.

In terms of the bonus part, students could realize the function of saving and loading the game. In other words, user can save current game and quit. Next time, the user could start the saved game.

Model Answer – Software Development Process

2. Analysis

Inputs:

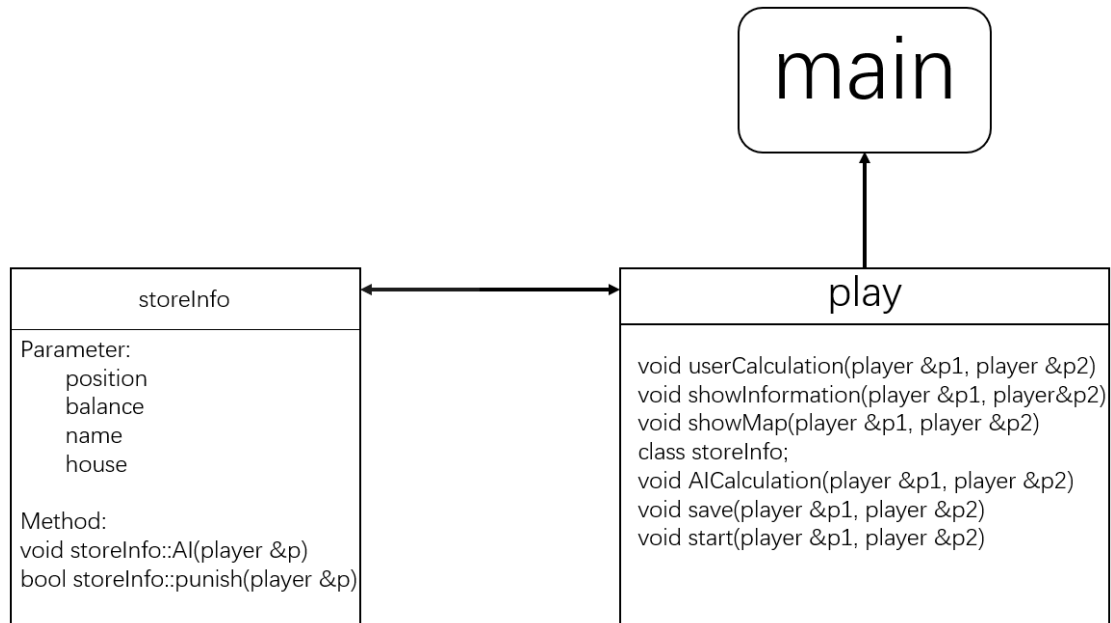
- At the beginning, the game will ask the user whether to start a new game. Input "Y" will start the saved game. Input "N" will start a new game. Other inputs are all invalid.
- If the user chooses the new game, the user needs to input his name. If the user chooses the old game, it will load the saved game and resume the game.
- Then, system will ask the user if want to buy the square that user is approaching. Input "Y" to buy the square. Input "N" to choose to stay and do nothing.
- If the player go to the square has been bought, it can be chosen to decide whether to invest. Input "Y" to invest. Input "N" to not invest.
- In addition, the user can save the current data of two player by inputting "S".

Outputs:

- Let the user input the name without the space and indicate the game should be played in full screen condition.
- Then the system will show user's name and explain the information on the map.
- Next, it will output information of two players. In addition, it will show the current position of the user and ask if the user want to invest the square after the user rolling the dice. If the user input "S", it will exit directly with saving the current data.
- After choosing the options, it will show the number of the dice that computer rolled and show the information of the square.
- If the player arrived the square or the adjacent square that bought by opponent, it would indicate the sentence about the punishment. Then the map would be drawn on the screen.
- If the balance of the computer is less than 0, the user will win. If the balance of the user is less than 0, the user will lose.

3. Design:

3.1 Flow Chart



3.2 Design step

1. Create a class to store the information of two players. In this class, the Boolean for method `punish()` is to judge if the player occupy the adjacent square that bought by opponent.
2. `userCalculation()` and `AICalculation()` are used to realize the rules indicated previously. The balance of user and computer player will change due to the different situation.
3. `showInformation()` will show the instant information of two players: Name, balance, houses, current position.
4. `showMap()` is one of the most difficult stuffs in this program. Firstly, the shaped of the draft map should be drawn. In addition, the length of each characteristics should be considered. After drawing the basic map, it is required to mark every bought square.

Finally, user could observe the detailed information from the completed map. For example, user can know the price of each square and how many and which squares are occupied by user themselves or opponents.

5. `save()` is to save the current information of two players. The data will be saved in a file called "save.txt". `start()` is to load the saved game.

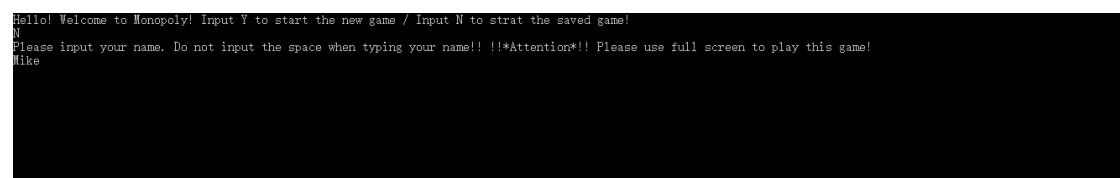
6. In `main()`, it will firstly ask user if want to start a new game or start the saved game. If the user choose to start the saved game, `start()` will be used immediately to load and apply the old data.

7. If one of the players declare bankruptcy, it will show the corresponding message.

Implementation:

See the code for assignment in the file: `storeInfo.h`, `storeInfo.cpp`, `player.h`, `player.cpp`, `main.cpp`.

Testing:



```
Hello! Welcome to Monopoly! Input Y to start the new game / Input N to strat the saved game!
N
Please input your name. Do not input the space when typing your name!! !!*Attention*!! Please use full screen to play this game!
Mike
```

Figure 2: At the beginning of the game.

```

Your name is: Mike
The number under the ordered square is the corresponding price of the square.
Your mark: **. PC's mark: ++
请按任意键继续. . .

```

Figure 3: Explain the marks to the user.

```

-----
. MIKE's Balance: 10000
. MIKE's Position: 0
. MIKE has square(s):
. Computer's Balance: 10000
. Computer's Position: 0
. Computer has square(s):
MIKE is rolling the dice! The number of the dice is: 4. MIKE has moved to position: 4
The price of this square now is 309. Input Y if you want to invest / Input N if you do not want to invest. Input S to save the game.
Attention!!: If you save the game, it will exit directly and your opponent will have one more chance to roll the dice and take action! So, be careful!!
?
You have bought the square: 4
请按任意键继续. . .
Computer is rolling the dice! The number of the dice is: 2. Computer has moved to position: 2
Computer spend 30 buying square: 2
请按任意键继续. . .
-----
. MIKE's Balance: 9691
. MIKE's Position: 4
. MIKE has square(s): 4
. Computer's Balance: 10000
. Computer's Position: 2
. Computer has square(s): 2

```

Figure 4: The outcome of the first time of the experiment.

Figure 4 shows:

- User can approach the position after rolling the dice.
- The square could be bought and system will record the information of the bought square.
- The balance will be decreased depends on the bought square.

	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	
	338	345	165	77	336	114	453	30	192	12	298	479	321	79	375	349	223	444	68	469	
61																					40
353																					13
62																					39
172																					357
63																					38
463																					261
64																					37
63																					284
65																					36
282																					431
66																					35
413																					449
67																					34
203																					194
68																					33
97																					300
69																					32
450																					25
70																					31
380																					77
71																					30
173																					189
72																					29
88																					486
73																					28
130																					493
74																					27
418																					150
75																					26
450																					391
76																					25
71																					197
77																					24
458																					462
78																					23
146																					265
79																					22
228																					373
80																					21
472																					489
Start	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
	432	38++	275	309**	245	421	166	293	80	42	507	459	97	430	159	499	37	360	305	221	

Figure 5: The map will record the bought squares (** for user, ++ for computer).

```

MIKE's is rolling the dice! The number of the dice is: 6, MIKE has moved to position: 10
The price of this square now is 42, Input Y if you want to invest / Input N if you do not want to invest. Input S to save the game.
Attention!!!: If you save the game, it will exit directly and your opponent will have one more chance to roll the dice and take action! So, be careful!!
Y
You have bought the square: 10
请按任意键继续.
Computer's is rolling the dice! The number of the dice is: 1, Computer has moved to position: 3
Computer Spend 275 buying square: 3
请按任意键继续.
.
.
.
. MIKE's Balance: 9649
. MIKE's Position: 10
. MIKE has square(s): 4, 10
. Computer's Balance: 9637
. Computer's Position: 3
. Computer has square(s): 2, 3

```

Figure 6: The second time of the experiment.

	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	
	338	345	165	77	336	114	453	30	192	12	298	479	321	79	375	349	223	444	68	469	
61																					40
353																					13
62																					39
172																					357
63																					38
463																					261
64																					37
63																					284
65																					36
282																					431
66																					35
413																					449
67																					34
203																					194
68																					33
97																					300
69																					32
450																					25
70																					31
380																					77
71																					30
173																					189
72																					29
88																					486
73																					28
130																					493
74																					27
418																					150
75																					26
450																					391
76																					25
71																					187
77																					24
458																					462
78																					23
145																					265
79																					22
225																					373
79																					22
225																					373
80																					21
472																					439
Start	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
	432	38++	275++	309**	245	421	166	293	80	42**	507	459	97	430	153	499	37	360	305	221	

Figure 7: The map for the second time of the experiment.

```

• MIKE's Balance: 9649
• MIKE's Position: 10
• MIKE has square(s): 4 10
• Computer's Balance: 9687
• Computer's Position: 3
• Computer has square(s): 2 3
MIKE's is rolling the dice! The number of the dice is: 3. MIKE has moved to position: 13
The price of this square now is 97. Input Y if you want to invest / Input N if you do not want to invest. Input S to save the game.
Attention!!: If you save the game, it will exit directly and your opponent will have one more chance to roll the dice and take action! So, be careful!!
  N
Invalid input!!! Please just input Y, N or S!!!
You choose not buy this square!
请按任意键继续. . .

```

Figure 8: When the input is invalid.

```

• MIKE's Balance: 3926
• MIKE's Position: 79
• MIKE has square(s): 4 10 16 20 22 24 30 36 39 43 44 48 52 54 59 65 68 72 73 74 76 79
• Computer's Balance: 5479
• Computer's Position: 78
• Computer has square(s): 2 3 8 9 11 25 28 29 33 37 38 40 42 45 51 57 58 62 67 78
Passing the start! MIKE's balance has added 200! Good luck!
MIKE's is rolling the dice! The number of the dice is: 3. MIKE has moved to position: 2
MIKE has pass Computer's bought square MIKE need to be punished by 4.5!
请按任意键继续. . .
Passing the start! Computer's balance has added 200! Good luck!
Computer's is rolling the dice! The number of the dice is: 6. Computer has moved to position: 4
Computer has passed MIKE's bought square. Computer need to be punished by 40.3!
请按任意键继续. . .
• MIKE's Balance: 4151
• MIKE's Position: 2
• MIKE has square(s): 4 10 16 20 22 24 30 36 39 43 44 48 52 54 59 65 68 72 73 74 76 79
• Computer's Balance: 5652
• Computer's Position: 4
• Computer has square(s): 2 3 8 9 11 25 28 29 33 37 38 40 42 45 51 57 58 62 67 78

```

Figure 9: Punishment.

Figure 9 shows if the player approach the adjacent square which has been bought by the opponent.

```

-----
• MIKE's Balance: 38
• MIKE's Position: 24
• MIKE has square(s): 1 3 9 15 16 19 20 21 23 28 30 32 38 39 44 46 47 48 51 52 53 56 57 59 61 64 69 70 71 75 79
• Computer's Balance: 1376
• Computer's Position: 16
• Computer has square(s): 4 6 8 10 11 12 14 17 18 22 24 25 27 29 33 35 36 37 42 43 45 54 58 60 62 63 66 68 73 76 -----
MIKE's is rolling the dice! The number of the dice is: 6. MIKE has moved to position: 30
You do not have enough money, wait for the next turn!
请按任意键继续. . .
Computer's is rolling the dice! The number of the dice is: 4. Computer has moved to position: 20
Computer has passed MIKE's bought square. Computer need to be punished by 43.65!
请按任意键继续. . .
-----

```

Figure 10: When the user do not have enough to buy.

```

Hello! Welcome to Monopoly! Input Y to start the new game / Input N to strat the saved game!
Y
Started from last time!
请按任意键继续. . .

```

Figure 11: When load the saved game.

```

***** You win this game! *****请按任意键继续. . .

```

Figure 12: When user win (computer declares bankruptcy).

```

C:\Users\mike\source\repos\main\Debug\main.exe
***** GAME OVER *****

```

Figure 13: When computer win (user declares bankruptcy).