Assignment 4

1301058

Zhang Junming

Exercise 1

Question

EXERCISE 1 (7.5 POINTS OUT OF 15)
Derive a sub-class iFraction from the base class Fraction, which was designed by you in Exercise 2 of Assessment 2. The iFraction class represents *the mixed fractions* like $1\frac{2}{3}$
Part 1: Design the sub-class **iFraction**: 1. It should have a constructor for initialisation; 2. It should contain a method to display the mixed fraction on screen;
Part 2: Design an external function **convertF** (not function member of class Fraction and iFraction) to convert the improper fractions to mixed fractions. Hint: convertF can be the friend function of those two classes.

Model Answer

Software Development Process

1. Problem statement

In part 1: Design a sub-class iFraction of class Fraction in Assignment 2 which could represents the mixed fraction, at the same time, write the constrictor and display function for iFraction.

In part 2: Write an external function to convert improper fractions to mixed fractions.

2. Analysis

Inputs:

1) Fraction: Construct part (two numbers, one represents numerator, another represents denominator)
2) iFraction: Construct part(one number to represent integer part)
3) In part2: Convert function, ask user to create an improper fraction

Outputs:

1) Display the improper fraction
2) Display the mixed fraction
3) Display the improper fraction which user input and then display the mixed fraction which convert from the improper fraction

Additional requirements or constraint

The convert function should be an external function, it means that convert function is not the function member of class

3. Design

Fraction.h

1. Add header file Fraction.h

   <1> use #ifndef and #define to prevent repetitive include

   <2> include library "iostream"

   <3> using of the ste namespace

   <4> define a class which called Fraction

Private

int top – represent the numerator

int bottom – represent denominator

Public

Fracion() – default constructor

<1> let top equal to 0 and bottom equal to 1

Fraction(int N,int D) – normal constructor

<1> let top equal to N and bottom equal to D

void out () – display the fraction on screen

<1> display numerator first, then display "/" and the denominator

<5> add class iFraction and iFraction convertF to friend

<6> use #endif

2. Add header file iFraction.h

<1> use #ifndef and #define to prevent repetitive include
<2> include library "iostream" and "Fraction.h"
<3> using of the ste namespace
<4> define a private type sub-class which called Fraction and it inherit from class Fraction

Private

int number – represent the integer part

Public

iFracion() – default constructor

<1> let number equal to 1

iFraction(int Number,Fraction Term) – normal constructor

<1> let number equal to Number, top equal to top in Term and bottom equal to bottom in Term

void display () – display the fraction on screen

<1> display number first, then display the fraction part

<5> add class Fraction and iFraction convertF to friend

<6> use #endif

3. Write external function int max(int x,int y) to find the greatest common divisor in fraction
   <1> int i – to store the remainder
   <2> there are four conditions:
   i numerator and denominator are positive
   ii numerator is negative and denominator is positive
   iii numerator is positive and denominator is negative
   iiii numerator and denominator are negative
   setting up a loop let i equal to numerator / denominator and then let numerator equal to denominator and let denominator equal to i until i equal to 0, at this time, the value of numerator is the greatest common divisor

   for i and iiii, return x

   for ii and iii, return -x

4. Write convert function convert: iFraction convert (Fraction term)
   <1> int k – represent the greatest common divisor
   <2> int i – represent the remainder of fraction
   <3> iFraction result – to store mixed fraction
   <4> using external function max to obtain the greatest common divisor

<5> numerator and denominator divide their greatest common divisor at the same time

<6> let i equal to numerator divide denominator

<7> there are four conditions

i numerator and denominator are positive

ii numerator is negative and denominator is positive

iii numerator is positive and denominator is negative

iiii numerator and denominator are negative

for i and iiii

if their remainder small than 1

keep the original numerator and denominator

but if the numerator and denominator are negative, let them become positive and let integer part * -1

give the related value to result

if their remainder big than 1

let integer part equal to reminder

let numerator equal to original numerator – original denominator*i

if the numerator and denominator are negative, let them become positive and let integer part * -1

give the related value to result

for i and iii

it similar as i and iiii but change the negative value to positive first and adjust the positive and negative after convert

5. Write the main function

<1> tell user the function of this program

<2> ask user to input an improper fraction

<3> convert it to mixed fraction

<4> display the improper fraction and mixed fraction on screen

4. test:

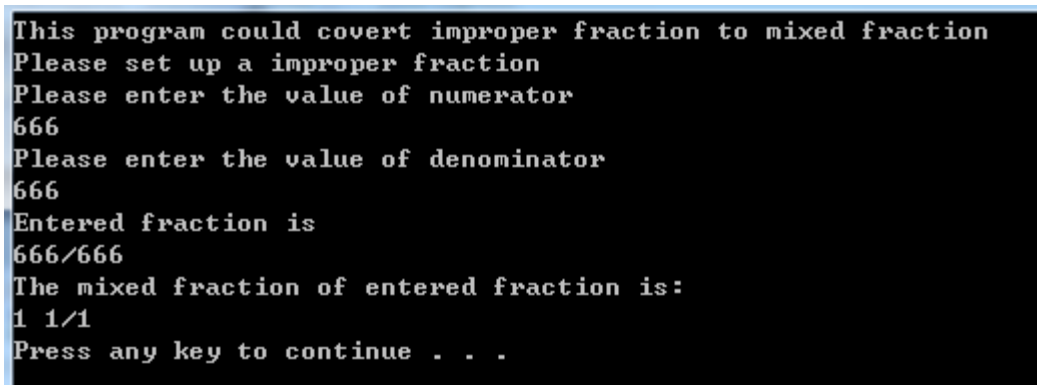
```
This program could covert improper fraction to mixed fraction

for part 1: constructor test
2 8/9
```

Part 1: constructor

```
This program could covert improper fraction to mixed fraction
Please set up a improper fraction
Please enter the value of numerator
12
Please enter the value of denominator
45
Entered fraction is
12/45
The mixed fraction of entered fraction is:
1 4/15
Press any key to continue . . .
```

```
This program could covert improper fraction to mixed fraction
Please set up a improper fraction
Please enter the value of numerator
12
Please enter the value of denominator
45
Entered fraction is
12/45
The mixed fraction of entered fraction is:
1 4/15
Press any key to continue . . .
```


This program could covert improper fraction to mixed fraction
Please set up a improper fraction
Please enter the value of numerator
666
Please enter the value of denominator
666
Entered fraction is
666/666
The mixed fraction of entered fraction is:
1 1/1
Press any key to continue . . .

Part 2: test

Exercise 2

Question

EXERCISE 2 (7.5 POINTS OUT OF 15)
Take the code provided for the container class (**container.h**), player class (**player.h**) and the swordsman class (**swordsman.h**), and the main function (**main.cpp**).
Part 1: 1. Read the source codes, understand the meaning and fill the blank (shown as ?????????) to make it work; 2. Generate the CRC cards of the classes: clarify the responsibilities of each class, illustrate the collaboration with others and label all the members' as public, protected and private. Then draw their hierarchy chart;
Part 2: 3. Design another two classes, archer and mage. Generate their CRC card and put them in the hierarchy chart drawn above. Write the code for these two classes and make them work (cooperate with main function); 4. Modify main function, to have enemies with all three professions (could be randomly chosen).
Part 3 (Optional): 5. Can you involve another attribute – luck, in this game?
Notes: The factors are imperfect, make the fight too hard or too easy. Change whatever you want, but leave hints for reviewers to know what you have done.

Model Answer

Software Development Process

1. Problem statement

In part 1: Add codes to complete a game, totally 7 blanks. Let this game run successful and then generate a CRC cards to explain the relationship between each class

In part 2: Add two new professions to this game and let this game run successful, add them to the CRC in the end

2. Analysis

Inputs:

Orders to control your role to fight with one enemy

Outputs:

The order result and weather win this game

3. Design

Blank 1:

#ifndef _CONTAINER :

#ifndef means if not define , it works with #define, they could prevent repeated include to program

Blank 2:

numOfHeal--;

numOfHeal means the number of Heal Water, it could add blood to character, bool container::useHeal() this function means use Heal Water, therefore, the number of Heal Water should minus 1 when use this function

Blank 3:

bag.set(bag.nOfHeal()+p.bag.nOfHeal(),bag.nOfMW()+p.bag.nOfMW());

it means get the Heal and Magic water from died enemy, this function is: void player::transfer(player &p)and this function will run when enemy died, the number of people's Heal and Magic water will plus enemy's

Blank 4:

void showinfo(player &p1, player &p2)

This function name found in player.h header files and is a friend function to class player

Blank 5:

class swordsman:public player

Base on the note after this code: `subclass swordsman publicly inherited from base player` it could infer that class swordsman is a sub-class of class player and the type of inherit is public

Blank 6:

Delete human;

Note is: `player is dead, program is getting to its end, what should we do here?`

It could infer player is died, therefore delete the data of player

Blank 7:

Nothing

Note is: `You win, program is getting to its end, what should we do here?`

When player win all enemy, the following code will display congratulations, therefore I think there does not need other code, maybe it could create a file to store the winner's name.

Another two classes:

Archer:

Similar to swordsman, create a header file which called archer and a cpp file which called archer

<1> change the enumerate type of job to ar

<2> reduce the original HP and the development of it

promote the original AD to improve attack ability

promote the speed to improve the rate of evade

archer should be a high attack and low HP people, therefore I change the normal attack calculate formula to improve archer's attack ability

Magical man:

Similar to swordsman, create a header file which called migicman and a cpp file which called migicman

<1> change the enumerate type of job to mg

<2> promote the original MP and the development of it

promote the original AD to improve special attack ability

reduce the speed to decrease the rate of evade

magic man should has a high special attack and low HP, therefore I change the special attack calculate formula to improve mage's spcial attack ability

Enemies randomly choose their professions

<1> add "time.h" library

<2> set the seeds of time for generate different random number

<3> generate random number between 1~3 and store in int choose

<4> judge the value of choose, if choose equal to 1, set enemy to swordsman; if choose equal to 2, set enemy to archer; if choose equal to 3, set enemy to magic man

CRC cards:

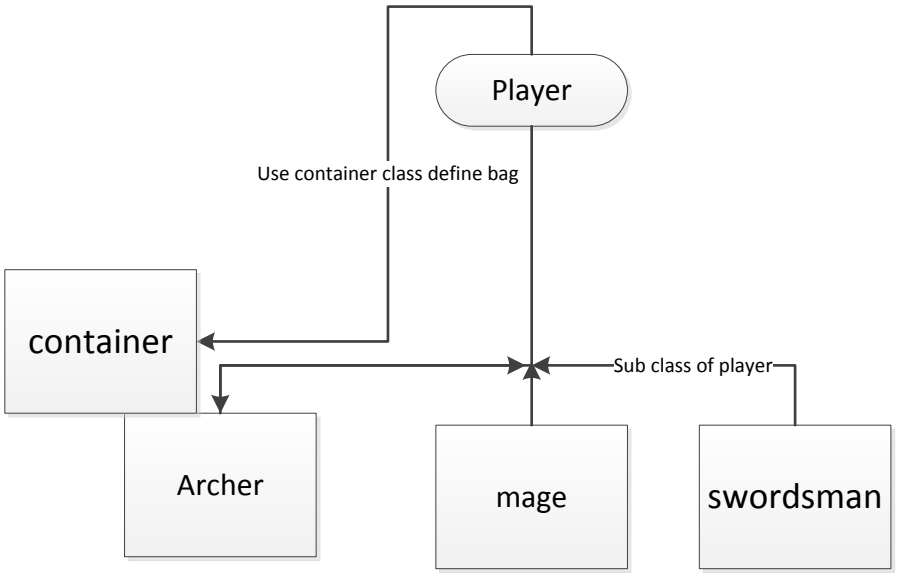
CLASS CONTAINER
Protected: 1. int numOfHeal to represents the number of Heal Water 2. int numOfMW to represents the number of Magic Water
Public: container(): Default constructor void set(int heal_n,int mw_n); give player Heal and Magic water int nOfHeal() obtain the number of Heal Water int nOfMW() obtain the number of Magic water void display() display the related information of player bool useHeal() judge weather use the Heal Water bool useMW() judge weather use the Magic Water

CLASS PLAYER
Let function showinfo and classed swordsman, archer and magicman be the friend to class player
Protected: declare the property of player such like HP,HPmax….. string name to store the name of player job role – represents the three different job declare bag which is a class of container to store the Heal and Magic water of player

bool attack: judge weather use normal attack. bool specialatt: judge weather use special attack. isLevelUp: judge weather level up reFill(): reset the HP and MP when start a new fight bool death: judge weather player dead isDead(): check whether player dead bool useHeal(): judge weather use Heal Water bool useMW(): judge weather use Magic Water transfer(): let the Heal and Magic water to player when win the enemy shwoRole(): display the job of player
Private: bool playerdeath: judge weather user death and weather need to inherit

CLASS OF THREE JOB
Public: archer(): normal constructor isLevelUp(): player is level up bool attack: weather use normal attack bool specialatt: weather use special attack AI(): control the action of enemy, use which type attack, weather use Heal or Magic water

Hierarchy Chart



4. Test:

```
Please input player's name: junming
Please choose a job: 1 Swordsman, 2 Archer, 3 Mage
3
```

Create player

```
STAGE1
Your opponent, a Level 1 Swordsman.
Press any key to continue . . .
```

Enemy

```
######################################################################
# Player   junming   LV.   1  # Opponent   Warrior   LV.   1 #
# HP 140/140 ¦ MP 150/150     # HP 150/150 ¦ MP  75/ 75      #
# AP  40 ¦ DP  35 ¦ speed  25 # AP  25 ¦ DP  25 ¦ speed  25  #
# EXP     75 Job:    Mage   # EXP     75 Job: Swordsman    #
----------------------------------------------------------------------
Your bag contains:
Heal(HP+100): 1
Magic Water (MP+80): 1
######################################################################
Please give command:
1 Attack; 2 Special Attack; 3 Use Heal; 4 Use Magic Water; 0 Exit Game
```

Interface

```
Please give command:
1 Attack; 2 Special Attack; 3 Use Heal; 4 Use Magic Water; 0 Exit Game
1
junming uses wand, Warrior's HP decreases 14
junming obtained 16 experience.
```

Use normal attack

```
junming Level UP!
HP improved 7 points to 147
MP improved 5 points to 155
Speed improved 1 points to 26
AP improved 5 points to 48
DP improved 2 points to 38
```

Level up

```
Critical attack: Warrior uses bash, junming's HP decreases 24
Warrior obtained 28 experience.
```

Enemy action

```
Please give command:
1 Attack; 2 Special Attack; 3 Use Heal; 4 Use Magic Water; 0 Exit Game
2
junming uses final spark attack, Warrior's HP decreases 58
junming obtained 87 experience.
Press any key to continue . . .
```

Use special attack

```
Please give command:
1 Attack; 2 Special Attack; 3 Use Heal; 4 Use Magic Water; 0 Exit Game
4
junming used Magic Water, MP increased by 100.
```

Use magic water

Enemy died

Win

When player died

Random job of enemy