Assignment 5

Now you're going to design a Lending System to manage the information of the repayers in a bank. In Q1, you are asked to simply calculate the month span about mortgage. In Q2, you should design a structure to represents repayer. In Q3, you need to build a statistic system to manage data of repayers for the bank. The bank would appreciate it if you could finish the designment as efficiency and concise as possible.

> Designer: ZHU Yueming
>
> Document: HE Qijun
>
> Junit Code: LI Qilong, QIN Yao
>
> Tester: YU Fenghua

Q1 MortgageDate (10%)

Design a class named **MortgageDate**

Submit:

MortgageDate.java

Methods:

1. getDifference

```
public static int getDifference(int date1, int date2)
```

Return the month difference of `date1` and `date2`

It is guaranteed that the `date` is composed of 6 numbers, of which the first four numbers represent the years, and the last two represent months. **All the Dates in this Assignment are in this format!**

It is guaranteed that `date2` comes after `date1` in all test cases.

Testcase：

```
for (int i = 2023; i < 2025; i++) {
            for (int j = 1; j <= 3; j++) {
                        System.out.println(getDifference(202301, i * 100 + j));
            }
}
```

Result:

```
0
1
2
12
13
14
```

2. getMonthsPassedDate

```
public static int getMonthsPassedDate(int startDate, int months)
```

Return a date represents that `months` passed from `startDate`

Return value is the same format of `startDate`.

It is guaranteed that `int months` is bigger than 0.

Testcase:

```
for (int i = 0; i < 20; i++) {
            System.out.println(getMonthsPassedDate(202310, i));
        }
```

Result:

```
202310
202311
202312
202401
202402
202403
202404
202405
202406
202407
202408
202409
202410
202411
202412
202501
202502
202503
202504
202505
```

Q2 Repayer (50%)

Submit:

MortgageDate.java

Repayer.java

EqualPaymentRepayer.java

EqualPrincipalRepayer.java

Profession.java

or other class you think is necessary

Class Profession

An enumeration class, representing the profession.

```
public enum Profession {
    PUBLIC_INSTITUTIONS,ENTERPRISE,AGRICULTURE,MILITARY_POLICE,OTHER
}
```

You can add other attributes and methods in this class.

Class Repayer

Abstract class representing a repayer.

Fields:

1. id

```
private int id;
```

The testcases of `id` is distinct among repayers.

2. gender

```
private char gender;//M or F
```

Represent gender values only 'M' (male) or 'F' (female).

3. profession

```
private Profession profession;
```

Represent the profession of the repayer, using the *enum class*.

4. totalLoan;

```
protected int totalLoan;
```

Total loan amount of a repayer.

5. mortgageStartDate

```
protected int mortgageStartDate;
```

The date where mortgage starts. (a six-numbers integer)

6. mortgageTotalMonths

```
protected int mortgageTotalMonths;
```

The total mortgage months. (repayers keep repaying the loan)

7. rate

```
protected double rate;//year rate
```

The year rate of mortgage. We **wouldn't** change year rate in test cases.

Methods:

1. Constructor

```
public Repayer(String infos)
```

`infos` includes these parameters below, forming a space-separated `String`. It is guaranteed that all the testcases are normal.

```
repayerID gender ProfessionId totalLoan date month rate
An Example:
Repayer("188 F 3 2000 202311 12 0.005")
```

In this format:

`ProfessionId` represents the **indices** of `enum Profession` .

For example, `ProfessionId=3` represents `AGRICULTURE`

`date` represents the `mortgageStartDate`

`month` represents the `mortgageTotalMonths`.

2. toString

```
public String toString()
```

The return value should be in the following format.

```
[id] gender profession
```

For example :

```
[188] F AGRICULTURE
```

3. getMonthNumber

```
public int getMonthNumber(int date)
```

Return the number of mortgage month about the month `date` :

Begin from **1**, if the date isn't in the mortgage months, then return **0**.

For example :

mortgageStartDate	mortgageTotalMonths	Date	return value
202301	10	202301	1
202301	10	202302	2
202301	10	202212	0
202301	10	202401	0

4. getPayment

```
public abstract double getPayment(int date);
```

return the mortgage payment in the month `date`. If the date isn't in the mortgage months, then return **0**.

5. getInterest

```
public abstract double getInterest(int date);
```

return the interest in the total mortgage in the month `date` . If the date isn't in the mortgage months, then return **0**.

6. getPrincipal

```
public abstract double getPrincipal(int date);
```

return the principal in the total mortgage in the month `date`. If the date isn't in the mortgage months, then return **0**.

7. getCurrentLiability

```
public abstract double getCurrentLiability(int date);
```

Return the total loan owed to the bank before **mortgage starts** in current month `date`. If the date isn't in the mortgage months, then return **0**.

> For example:
>
> When the mortgage month is 1, the month that have been mortgaged is 0, so that if the return value of the getMonthNumber(date) is 1, the return value of the getCurrentLiability(date) is the totalLoan.

Class EqualPrincipalRepayer

People who repay in a way called "equal principal". This class is subclass of `Repayer`.

Fields:

1. principal

```
private final double principal;
```

It is the principal to repay for each month, and the principal in each month is a certain and equal value.

Methods:

Implement four abstract methods in the superclass (Method 4-7).

(M = totalLoan ; m = loan_have_been_repayed ; n = mortgageTotalMonths ; r = month rate)

How to calculate principal :

$$principal = M \div n \tag{1}$$

How to calculate interest :

$$interest = (M - m) \times r \tag{2}$$

How to calculate payment :

$$\text{payment} = (M \div n) + (M - m) \times r \tag{3}$$

How to calculate CurrentLiability :

Before repaying this month, the currentLiability is:

$$currentLiability = M - m \tag{4}$$

$$nextMonthLiability = currentLiability - principal \tag{5}$$

Example：

```
totalLoan: 50000,
mortgageStartDate: 202310,
mortgageTotalMonths: 10,
rate: 5%
```

date	payment	Principal	Interest	Liability
202310	5208.3	5000	208.3	5.00 w
202311	5187.5	5000	187.5	4.50 w
202312	5166.7	5000	166.7	4.00 w
202401	5145.8	5000	145.8	3.50 w
202402	5125.0	5000	125.0	3.00 w
202403	5104.2	5000	104.2	2.50 w
202404	5083.3	5000	83.3	2.00 w
202405	5062.5	5000	62.5	1.50 w
202406	5041.7	5000	41.7	1.00 w
202407	5020.8	5000	20.8	0.50 w
202408	0	0	0	0 w

Class EqualPaymentRepayer

People who repay in a way called "equal payment", which means they repay certain amount (not principal) each month. This class is subclass of Repayer.

Fields:

1. payment

```
private final double payment;
```

Total repayments every month is certain.

2. insterests[]

```
private final double[] interests;
```

Record the interests to repay every month.

Methods

Implement four abstract methods in the superclass (Method 4-7).

(M = totalLoan ; m = loan_have_been_repayed ; n = mortgageTotalMonths ; r = month rate)

How to calculate payment :

$$\text{payment} = [M \times r \times (1+r)^n] \div [(1+r)^n - 1] \tag{6}$$

How to calculate interest :

$$interest = (M - m) \times r \tag{7}$$

How to calculate principal :

$$principal = payment - interest \tag{8}$$

How to calculate CurrentLiability :

Before repaying this month, the currentLiability is:

$$currentLiability = M - m \tag{9}$$

$$nextMonthLiability = currentLiability - principal \tag{10}$$

Example：

```
totalLoan: 50000,
mortgageStartDate: 202310,
mortgageTotalMonths: 10,
rate: 5%
```

date	payment	Principal	Interest	Liability
202310	5115.3	4907.0	208.3	5.00 w
202311	5115.3	4927.4	187.9	4.51 w
202312	5115.3	4947.9	167.4	4.02 w
202401	5115.3	4968.6	146.7	3.52 w
202402	5115.3	4989.3	126.0	3.02 w
202403	5115.3	5010.0	105.2	2.53 w
202404	5115.3	5030.9	84.4	2.02 w
202405	5115.3	5051.9	63.4	1.52 w
202406	5115.3	5072.9	42.4	1.02 w
202407	5115.3	5094.1	21.2	0.51 w
202408	0	0	0	0 w

For all classes in this question, you can add other attributes, methods or classes that you think is necessary.

Q3 CommercialLoan (40%)

Submit:

MortgageDate.java

Repayer.java

EqualPaymentRepayer.java

EqualPrincipalRepayer.java

Profession.java

CommercialLoan.java

ConcreteCommercialLoan.java

or other class you think is necessary

Interface CommercialLoan

An **Interface** to manage all the repayers' information, and to query their credit situation.

```
public interface CommercialLoan {
}
```

Methods:

1.1 loan

```
public void loan(String loanInfo);
```

The method is to input one repayer's information with a format below:

```
1|101 F 1 3000000 202301 240 0.04
2|102 M 2 2500000 202210 180 0.045
1|103 F 1 800000 202101 60 0.057
```

- The first character represents the loan type:
  - 1 for `Equal Payment`
  - 2 for `Equal Principal`
- The second charater is a separater `|`
- Then, the following characters have the same format of the parameter `info` in constructer of class `Repayer`.

1.2 loan

```
public void loan(List<String> loanInfos);
```

This method is used to input multiple repayers, and each String value in `loadInfos` represents the parameter `loanInfo` in the method `public void loan(String loanInfo);` above.

2. getRepayerCountsByProfession

```
public int getRepayerCountsByProfession(Profession profession);
```

Return the counts of one profession among all repayers.

3. getRepayerById

```
public Repayer getRepayerById(int id);
```

Return a repayer object by his/her own ID.

4. getAllRepayersById

```
public List<Repayer> getAllRepayersById();
```

Return a List of repayers, **sorted by their ID in ascending**.

5. getAllIncomeByDate

```
public double getAllIncomeByDate(int date);
```

Return payments from all repayers in a certain month (`date`)

6. getAllInterestByDate

```
public double getAllInterestByDate(int date);
```

Return interests from all repayers in a certain month (`date`)

7. getAllRepayersByCurrentLiabilityAndId

```
public List<Repayer> getAllRepayersByCurrentLiabilityAndId(int date);
```

Sorted all repayers in a List sorted by their current liability in the month `date` in ascending order. If more than one repayer has the same current liability, then sort by their own ID in ascending order.

Class ConcreteCommercialLoan

Fields:

1. repayers

```
 private List<Repayer> repayers;
```

Methods:

1. Constructor

```
public ConcreteCommercialLoan();
```

The Constructor in definition of repayers.

2. other methods

Then you should implement all the methods in `Interface CommercialLoan`.

> For all classes in this question, you can add other attributes, methods or classes that you think is necessary.