**Project: The Account Class**

Problem Description:

(*The Account class*) Design a class named Account that contains:

* A private int data field named id for the account (default 0).
* A private double data field named balance for the account (default 0).
* A private double data field named annualInterestRate that stores the current interest rate (default 0). Assume all accounts have the same interest rate.
* A private Date data field named dateCreated that stores the date when the account was created.
* A no-arg constructor that creates a default account.
* A constructor that creates an account with the specified id and initial balance.
* The accessor and mutator methods for id, balance, and annualInterestRate.
* The accessor method for dateCreated.
* A method named getMonthlyInterestRate() that returns the monthly interest rate.
* A method named withdraw that withdraws a specified amount from the account.
* A method named deposit that deposits a specified amount to the account.

Draw the UML diagram for the class. Implement the class. Write a test program that creates an Account object with an account ID of 1122, a balance of $20,000, and an annual interest rate of 4.5%. Use the withdraw method to withdraw $2,500, use the deposit method to deposit $3,000, and print the balance, the monthly interest, and the date when this account was created.

**Project: The MyInteger Class**

Problem Description:

Design a class named MyInteger. The class contains:

* An int data field named value that stores the int value represented by this object.
* A constructor that creates a MyInteger object for the specified int value.
* A get method that returns the int value.
* Methods isEven(偶), isOdd(奇), and isPrime(素) that return true if the value is even, odd, or prime, respectively.
* Static methods isEven(int), isOdd(int), and isPrime(int) that return true if the specified value is even, odd, or prime, respectively.
* Static methods isEven(MyInteger), isOdd(MyInteger), and isPrime(MyInteger) that return true if the specified value is even, odd, or prime, respectively.
* Methods equals(int) and equals(MyInteger) that return true if the value in the object is equal to the specified value.
* A static method parseInt(**char**[]) that converts an array of numeric characters to an int value.
* A static method parseInt(String) that converts a string into an int value.

Implement the class. Write a client program that tests all methods in the class.

**实验问题描述**

**Problem Statement**

* Problem statement:

*Write a loan calculator program that computes both monthly and total payments for a given loan amount, annual interest rate, and loan period.*

Overall Plan

* Tasks:
  + Get three input values: **loanAmount**, **interestRate**, and **loanPeriod**.
  + Compute the monthly and total payments.
  + Output the results.

Required Classes

LoanCalculator

Loan

Development Steps

* We will develop this program in five steps:
  1. Start with the main class LoanCalculator. Define a temporary placeholder Loan class.
  2. Implement the input routine to accept three input values.
  3. Implement the output routine to display the results.
  4. Implement the computation routine to compute the monthly and total payments.
  5. Finalize the program.

Step 1 Design

* The methods of the LoanCalculator class

|  |  |  |
| --- | --- | --- |
| Method | Visibility | Purpose |
| start | public | Starts the loan calcution. Calls other methods |
| computePayment | private | Give three parameters, compute the monthly and total payments |
| describeProgram | private | Displays a short description of a program |
| displayOutput | private | Displays the output |
| getInput | private | Gets three input values |

Step 1 Test

* In the testing phase, we run the program multiple times and verify that we get the following output
* inside describeProgram
* inside getInput
* inside computePayment
* inside displayOutput

Step 2 Design

* Design the input routines
  + LoanCalculator will handle the user interaction of prompting and getting three input values
  + LoanCalculator calls the setAmount, setRate and setPeriod of a Loan object.

Step 2 Test

* We run the program numerous times with different input values
* Check the correctness of input values by echo printing

System.out.println("Loan Amount: $"

+ loan.getAmount());

System.out.println("Annual Interest Rate:"

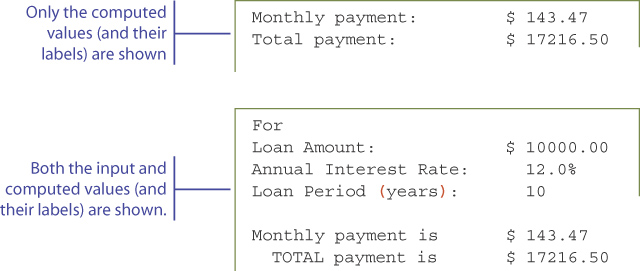
+ loan.getRate() + "%");

System.out.println("Loan Period (years):"

+ loan.getPeriod());

Step 3 Design

* We will implement the displayOutput method.
* We will reuse the same design we adopted in Chapter 3 sample development.



Step 3 Test

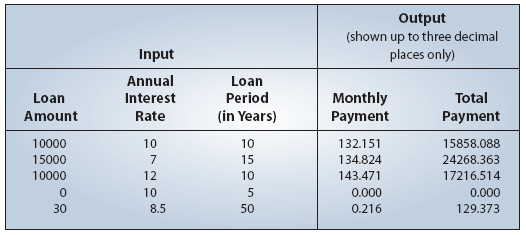
* We run the program numerous times with different input values and check the output display format.
* Adjust the formatting as appropriate

Step 4 Design

* Two methods getMonthlyPayment and getTotalPayment are defined for the Loan class
* We will implement them so that they work independent of each other.
* It is considered a poor design if the clients must call getMonthlyPayment before calling getTotalPayment.

Step 4 Test

* We run the program numerous times with different types of input values and check the results.



Step 5: Finalize

* We will implement the describeProgram method
* We will format the monthly and total payments to two decimal places using DecimalFormat.