/\*\*

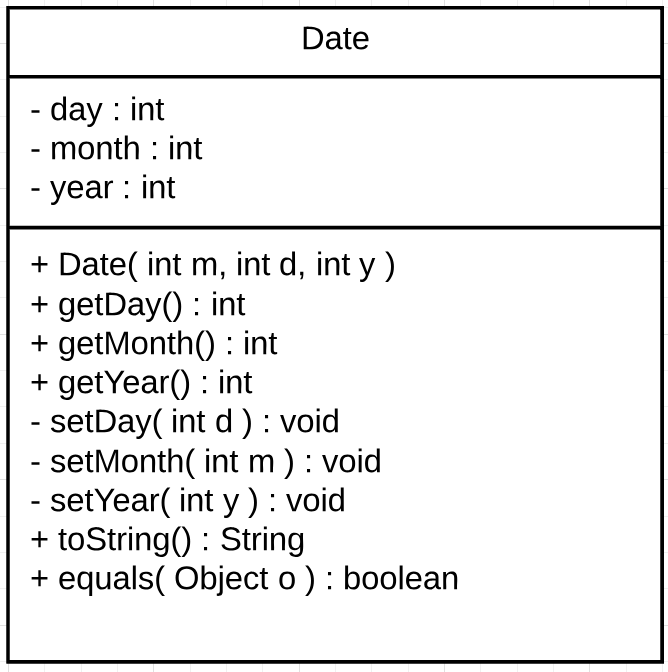
\* @author zhenhua.yang.1

\*/

import java.util.Calendar;

import java.util.GregorianCalendar;

public class Date {

 // Create instant variables

private int day;

private int month;

private int year;

// overloaded Date constructor

public Date( int m, int d, int y ){

setYear(y);

setMonth(m);

setDay(d);

}

// Access methods

public int getDay(){

return day;

}

public int getMonth(){

return month;

}

public int getYear(){

return year;

}

/\*\*

\* mutator methods

\* @param d

\*/

// set the day

private void setDay( int d ){

// create GregorianCalendar object

GregorianCalendar cal = (GregorianCalendar) Calendar.getInstance();

// if statements that validates users' inputs

if( month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12 ){

// check if the month is 1, 3, 5, 7, 8, 10, 12

if( d <=1 || d >= 31 ) // if d<1 or d>31, throw error message.

throw new IllegalArgumentException("Day value must be from 1 to 31 for month " + month);

} else if( month == 4 || month == 6 || month == 9 || month == 11 ){

// check if the month is 4, 6, 9 or 11

if( d <1 || d > 30 ) // if d<1 or d>30, throw error message.

throw new IllegalArgumentException("Day value must be from 1 to 30 for month " + month);

} else if( cal.isLeapYear(year) && month == 2 ){

// check if the year is leap year and the month=2.

if( d <1 || d > 29 ) // if d<1 or d>29, throw error message.

throw new IllegalArgumentException("At" + year + ", the day in February must be from 1 to 29");

} else if( !cal.isLeapYear(year) && month == 2 ){

// if the year is not leap year and the month=2.

if( d <1 || d > 28 ) // if d<1 or d>28, throw error message.

throw new IllegalArgumentException("At" + year + ", the day in February must be from 1 to 28");

}

day = d; // assign d to day.

}

// set month

private void setMonth( int m ){

if( m < 1 || m > 12 ) // if m<1 or m>12, throw error message.

throw new IllegalArgumentException("Month must be greater than 0 or smaller than 13" );

month = m;

}

// set month

private void setYear( int y ){

if( y < 1582 ) // if y<1582, throw error message.

throw new IllegalArgumentException( "Year must be greater than or equal to 1582");

year = y;

}

// toString method

@Override

public String toString(){

return ( month + "/" + day + "/" + year);

}

// equals method

@Override

public boolean equals( Object o ){

if( ! ( o instanceof Date ))

return false;

else{

Date objDate = ( Date ) o;

return day == objDate.day && month == objDate.month && year == objDate.year;

}

}

}

public class DateTest {

public static void main( String [] args ){

// create two Date objects

Date myDate = new Date( 2, 29, 2016 );

Date newDate = new Date( 2, 28, 2016);

// apply toString method

System.out.println( "The first date is " + myDate.toString() );

System.out.println( "The second date is " + newDate.toString() );

// apply equals method

if ( myDate.equals(newDate))

System.out.println( "Two dates are equal.");

else

System.out.println( "Two dates are not equal.");

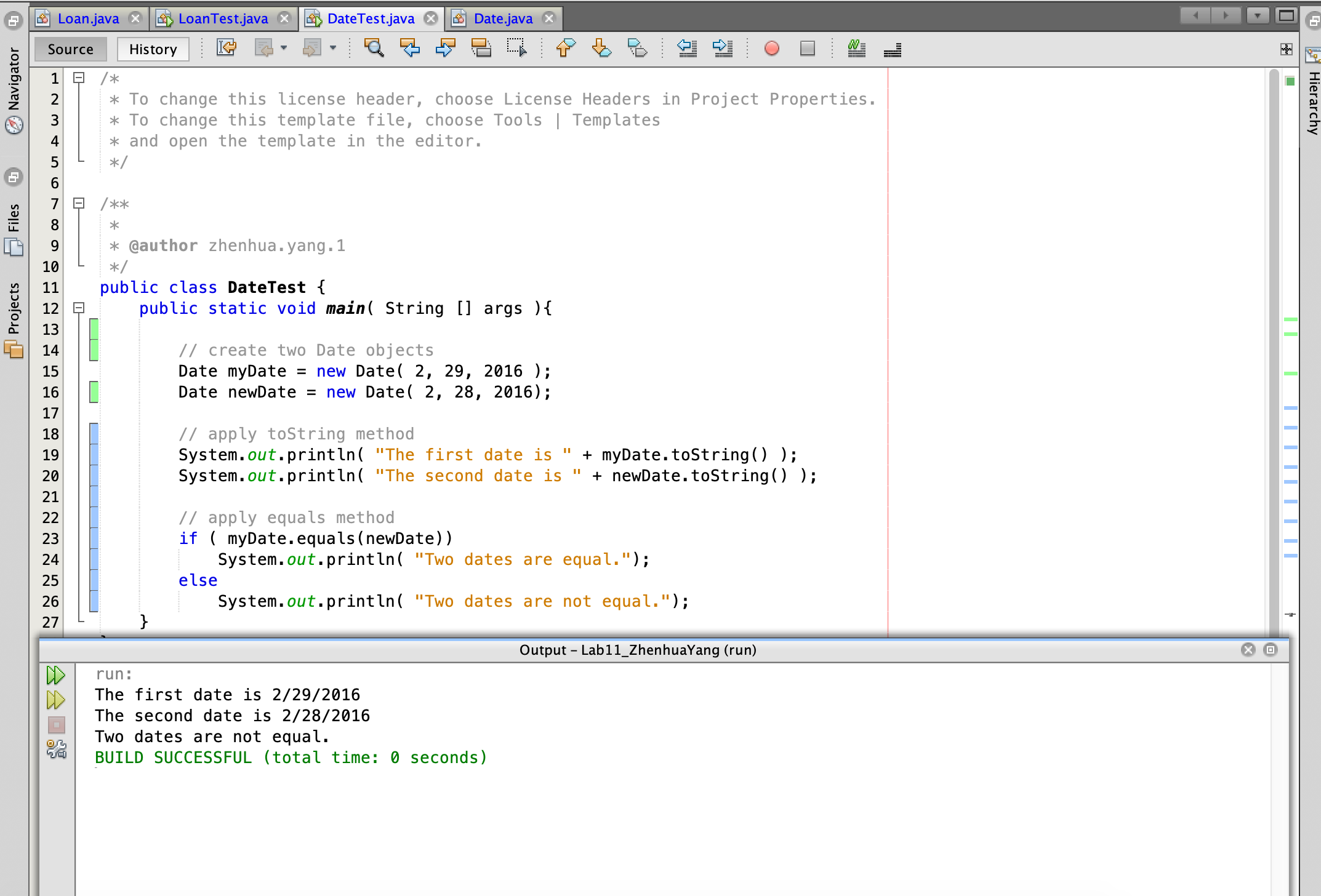
}

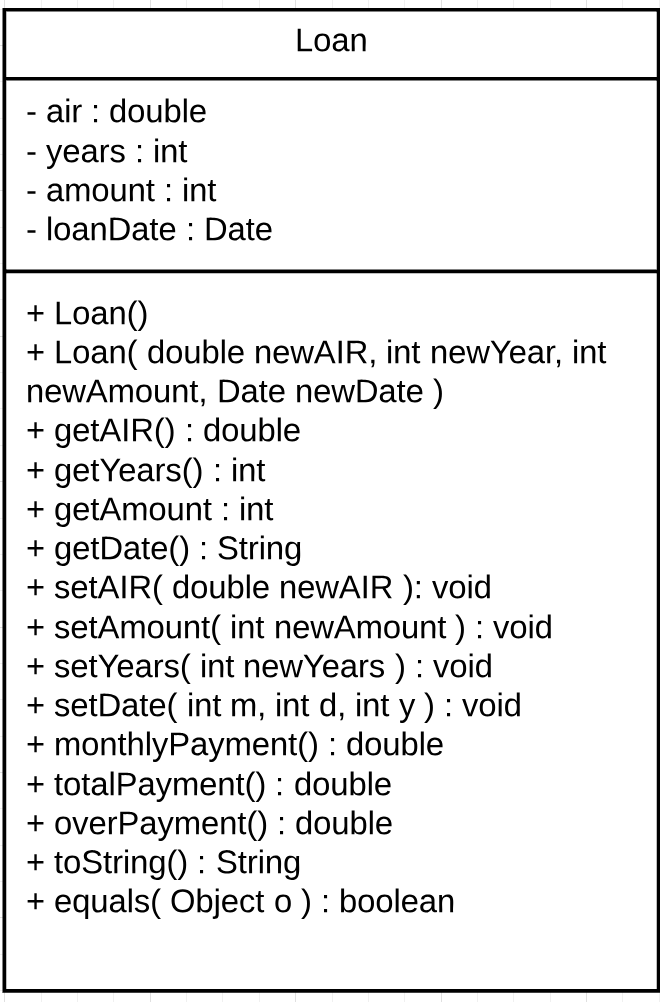
}

Explain in the word document why is it a good idea to declare the set methods to private instead of public.

**Declaring the set methods would give developer more control over the classes. The private methods can only be accessed by the methods in the same class. Clients outside the class could not use it directly. In this class, the setDay(), setMonth() and setYear() methods are referenced in the overloaded constructor and will be set in this constructor, which is in the same class. We allow user could only set day, month and year together through the constructor other than separately. Private methods provide the possibility of that**.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Inputs** | **Expected Output** | **Actual Output** |
| 1 | Year = 2012 Month = 2  Day = 28 | **2/28/2012** | 2/28/2012 |
| 2 | Year = 2013  Month = 3  Day = 32 | IllegalArgumentException generated with the message – “**Day value cannot be greater than 32 for month 3**” | IllegalArgumentException: Day value must be from 1 to 31 for month 3 |
| 4 | Year = 1325  Month = 3  Day = 5 | IllegalArgumentException generated with the message – “**Year must be greater than or equal to 1582**” | IllegalArgumentException: Year must be greater than or equal to 1582 |
| 5 | Year = 2013  Month = 2  Day = 29 | IllegalArgumentException generated with the message – **"At 2013, the day in February must be from 1 to 28"** | IllegalArgumentException: At2013, the day in February must be from 1 to 28 |
| 6 | Year = 2016  Month = 2  Day = 29 | **2/29/2016** | 2/29/2016 |



import java.text.DecimalFormat;

public class Loan {

// instance variables

private double air;

private int years;

private int amount;

private Date loanDate; // create Date object

// default constructor

public Loan(){

air = 0.0;

years = 0;

amount = 0;

loanDate = null;

}

// overloaded constructor

public Loan( double newAIR, int newYear, int newAmount, Date newDate){

air = newAIR;

years = newYear;

amount = newAmount;

loanDate = newDate;

}

// accessor methods

public double getAIR(){

return air;

}

public int getYears(){

return years;

}

public int getAmount(){

return amount;

}

public String getDate(){

return loanDate.toString();

}

// mutator methods

public void setAIR( double newAIR ){

air = newAIR;

}

public void setAmount( int newAmount ){

amount = newAmount;

}

public void setYears( int newYears ){

years = newYears;

}

public void setDate( int m, int d, int y ){

loanDate = new Date( m, d, y);

}

// calculate the monthly payment

public int monthlyPayment(){

double mir = air / 12;

return (int) ((mir \* amount) / ( 1 - ( Math.pow ( 1 / (1 + mir), 12 \* years))));

}

// calculate the total payment

public double totalPayment(){

// calculate total payment

double total = amount \* Math.pow( ( 1 + air ), years);

return total;

}

// calculate the over payment

public double overPayment(){

return this.totalPayment() - amount;

}

// toString method

@Override

public String toString(){

DecimalFormat percent = new DecimalFormat( "0.00%" );

DecimalFormat dollar = new DecimalFormat( "$0.00" );

return "===================Loan Information===================="

+ "\nAnnual interest rate: " + percent.format(air)

+ "\nMortgage amount: " + dollar.format(amount)

+ "\nMonthly payment: " + dollar.format(this.monthlyPayment())

+ "\nTotal payment over the years: " + dollar.format(this.totalPayment())

+ "\nOverpayment: " + dollar.format(this.overPayment())

+ "\noverpayment as a percentage: " + percent.format(this.overPayment() / this.getAmount());

}

// equals method

public boolean equals( Object o ){

if ( ! ( o instanceof Loan ) )

return false;

else{

Loan objLoan = (Loan) o;

return air == objLoan.air && amount == objLoan.amount

&& years == objLoan.years && loanDate == objLoan.loanDate;

}

}

}

import java.util.Scanner;

public class LoanTest {

public static void main( String [] args ){

// create instant variables

String tryAgain = "";

int month;

int day;

int year;

// create Loan object

Loan newLoan = new Loan();

// create Scanner object

Scanner input = new Scanner( System.in );

// prompt user input and print the result in a do while loop

do{

// ask users to input interest rate

System.out.print( "Annual interest rate> ");

newLoan.setAIR(input.nextDouble()); // set the interest rate to the newLoan

// ask users to input number of years

System.out.print( "Number of years the loan will be held > ");

newLoan.setYears(input.nextInt()); // set the year to the newLoan

­­ // ask users to input the loan amount

System.out.print( "Loan amount > ");

newLoan.setAmount(input.nextInt()); // set the loan amount to the newLoan

// ask users to input start date

System.out.print( "Start month, day and year followed by an enter for each> \n");

// set month, day and year to the newLoan

month = input.nextInt();

day = input.nextInt();

year = input.nextInt();

newLoan.setDate(month, day, year);

// print the information of newLoan

System.out.println( newLoan.toString());

// ask users to input YES to try again or any key to exit

System.out.print("\nPlease enter YES to try again or any key to exit> ");

tryAgain = input.next();

} while ( tryAgain.equals("YES") ); // end do while loop

System.out.println("\n\n========Thank you for trusting our loan company========\n");

}

}

