HW3- Yoav Cohen Nehemia

LoanCalc.java

```
/**

* Computes the periodical payment necessary to re-pay a given
loan.

*/

public class LoanCalc

{



     static double epsilon = 0.001;  // The computation tolerance
(estimation error)

     static int iterationCounter;    // Monitors the efficiency
of the calculation

     static int iterationCounter1;



    /**

     * Gets the loan data and computes the periodical payment.

     * Expects to get three command-line arguments: sum of the
loan (double),

     * interest rate (double, as a percentage), and number of
payments (int).

     */

     public static void main(String[] args)

     {

          // Gets the loan data

          double loan = Double.parseDouble(args[0]);
```

```
            double rate = Double.parseDouble(args[1]);

            int n = Integer.parseInt(args[2]);

            System.out.println("Loan sum = " + loan + ", interest
rate = " + rate + "%, periods = " + n);



            // Computes the periodical payment using brute force
search

            System.out.print("Periodical payment, using brute
force: ");

            System.out.printf("%.2f", bruteForceSolver(loan, rate,
n, epsilon));

            System.out.println();

            System.out.println("number of iterations: " +
iterationCounter);



            // Computes the periodical payment using bisection
search

            System.out.print("Periodical payment, using bi-section
search: ");

            System.out.printf("%.2f", bisectionSolver(loan, rate,
n, epsilon));

            System.out.println();

            System.out.println("number of iterations: " +
iterationCounter1);

      }



      /**
```

```
     * Uses a sequential search method  ("brute force") to
compute an approximation

     * of the periodical payment that will bring the ending
balance of a loan close to 0.

     * Given: the sum of the loan, the periodical interest rate
(as a percentage),

     * the number of periods (n), and epsilon, a tolerance level.

     */

     // Side effect: modifies the class variable
iterationCounter.

    public static double bruteForceSolver(double loan, double
rate, int n, double epsilon)

    {

      double payment = loan / n;

      while(endBalance(loan, rate, n, payment) > 0)

      {

            payment += epsilon;

            iterationCounter++;

      }

      return payment;



    }



    /**

     * Uses bisection search to compute an approximation of the
periodical payment
```

```
     * that will bring the ending balance of a loan close to 0.

     * Given: the sum of theloan, the periodical interest rate
(as a percentage),

     * the number of periods (n), and epsilon, a tolerance level.

     */

     // Side effect: modifies the class variable
iterationCounter.

    public static double bisectionSolver(double loan, double
rate, int n, double epsilon)

    {

     double L = (loan / n);

     double H = loan;

     double payment = (L + H) / 2.0;

     while((H - L) > epsilon)

     {

          if(endBalance(loan, rate, n, payment) *
endBalance(loan, rate, n, L) > 0)

          {

               L = payment;

          }

          else

          {

               H = payment;

          }

          payment = (L + H) / 2.0;
```

```
            iterationCounter1++;

      }

      return payment;

    }



      /**

      * Computes the ending balance of a loan, given the sum of
the loan, the periodical

      * interest rate (as a percentage), the number of periods
(n), and the periodical payment.

      */

      private static double endBalance(double loan, double rate,
int n, double payment)

      {

            for(int i = 0; i < n; i++)

            {

                  loan = (loan - payment) * (1.0 + rate / 100);

            }

            return loan;

      }

}
```

LowerCase.java

```
/** String processing exercise 1. */

public class LowerCase

{

    public static void main(String[] args)

    {

        String str = args[0];

        System.out.println(lowerCase(str));

    }



   /**

    * Returns a string which is identical to the original string,

    * except that all the upper-case letters are converted to
lower-case letters.

    * Non-letter characters are left as is.

    */

    public static String lowerCase(String str)

    {

        String down = "";

        for(int i = 0; i < str.length(); i++)

        {

            if((str.charAt(i)) >= 65 && (str.charAt(i) <= 90))

            {

                down = down + ((char)(str.charAt(i) + 32));
```

```
            }
            else
            {
                down = down + str.charAt(i);
            }
        }
        return down;
    }
}
```

UniqueChars.java

```
/** String processing exercise 2. */

public class UniqueChars {

    public static void main(String[] args)

    {

        String str = args[0];

        System.out.println(uniqueChars(str));

    }



    /**

     * Returns a string which is identical to the original
string,

     * except that all the duplicate characters are removed,

     * unless they are space characters.

     */

    public static String uniqueChars(String str)

    {

        String uniq = "";

        boolean flag;

        for (int i = 0;i < str.length(); i++)

        {

            flag = true;

            for(int j = 0; j < uniq.length(); j++)

            {
```

```
                if(str.charAt(i) == uniq.charAt(j))
                {
                    flag = false;
                }
            }
            if(flag || str.charAt(i) == 32)//equal to space
            {
                uniq = uniq + str.charAt(i);
            }
        }
        return uniq;
    }
}
```

Calendar.java

```
//Prints the calendars of all the years in the 20th century.

public class Calendar

{

    // Starting the calendar on 1/1/1900

     static int dayOfMonth = 1;

     static int month = 1;

     static int year = 1900;

     static int dayOfWeek = 2;     // 1.1.1900 was a Monday

     static int nDaysInMonth = 31; // Number of days in January



      //Prints the calendars of all the years in the 20th
century. Also prints the

      //number of Sundays that occured on the first day of the
month during this period.

     public static void main(String args[])

     {

         //int debugDaysCounter = 0;

         //int numperfectsundays = 0;

         String str = "";

         int choseY = Integer.parseInt(args[0]);

          while (year < choseY + 1)

          {

                advance();
```

```
                while(year == choseY)
                {
                      str = "";
                      if(dayOfWeek == 1) str = " Sunday";
                      System.out.println(dayOfMonth + "/" + month
+ "/" + year + str);
                      //if((dayOfWeek == 1) && (dayOfMonth == 1))
numperfectsundays++;
                      advance();
                }

                //debugDaysCounter++;
        }
        //System.out.println("During the year " + choseY + ": " +
numperfectsundays + " Sundays fell on the first day of the
month");
        // System.out.println(debugDaysCounter);
      }
      private static void advance()
      {
          if(dayOfWeek == 7) dayOfWeek = 1;
          else dayOfWeek++;

          if(dayOfMonth == nDaysInMonth)
          {
```

```
                if(month == 12)
                {
                      month = 0;
                      year++;
                }
                dayOfMonth = 1;
                month++;
                nDaysInMonth = nDaysInMonth(month, year);
           }
           else dayOfMonth++;



       }
     private static boolean isLeapYear(int year) // Returns true
if the given year is a leap year, false otherwise.
     {
         if(year % 4 == 0 && (year % 100 != 0 || year % 400 ==
0)) return true;
           else return false;
     }


     private static int nDaysInMonth(int month, int
year)//Returns number of days in month.
     {
```

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

           {

                return 31;

           } else if((month == 4) || (month == 6) || (month == 9)
|| (month == 11)) return 30;

                     else if(isLeapYear(year)) return 29;

                               else return 28;

     }

}
```