Home work 3 : Ohad Swissa

LoanCalc

```
/**
* 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 iterationCounter1;
        static int iterationCounter2;  // Monitors the efficiency of the calculation

   /**
    * 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: " + iterationCounter1);

                // 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: " + iterationCounter2);
        }

        /**
        * 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 g = loan / n ;
        double balance = endBalance(loan, rate, n, g);
        boolean endpay = false;
        while ( endpay == false )
    {
        balance = endBalance(loan, rate, n, g);
        if (balance > 0)
        {
            g = g + epsilon;
            iterationCounter1++;

        }
        else endpay = true;

    }
    return g;
}

/**
    * 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 g = (l+h) / 2;
    boolean endpay = false;
    while (h - l > epsilon)
    {
        if (endBalance(loan, rate, n, g) * (endBalance(loan, rate, n, l)) > 0)
        {
            l = g;
        }
        else
        {
            h = g;

        }
        iterationCounter2++;
        g = (l+h) / 2;

    }
    return g;

}

    /**
    * 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=1; i <= n ;i++)
    {
        loan = loan - payment;
        loan = loan * (1+(rate/100));

    }
    return loan;

    }
}
```

LowerCase

```
/** 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 s) {
      // Replace the following statement with your code
      String bigger = "";
      int i = 0;
      while (i < s.length())
      {
        char ch = s.charAt(i);
        if ((ch >= 65) && (ch <= 90))
      {
        bigger += (char)(ch + 32);
      }
      else
      {
         bigger = bigger + ch;
      }

      i++;

      }
      return bigger;
   }
}
```

UniqueChars

```
/** 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 s) {
      // Replace the following statement with your code
       String r = "" + (s.charAt(0));
       int i = 1;
      while (i < s.length())
      {
        char ch = s.charAt(i);
        if ((r.indexOf(ch) == -1) && (ch != ' '))
      {
        r += ch;
      }
      if (ch == ' ')
       {
          r += ch;
       }

        i++;
      }
      return r;
   }
}
```

Calendar

```
 //uses a couple of functions for pritning the given year calendar and all the sundays
public class Calendar {
        static int dayOfMonth = 1;
        static int month = 1;
        static int year = 1900;;
        static int dayOfWeek = 2;
        static int nDaysInMonth = 31;
        //function that gets an unt argument and a year argument and gives back the number of the days in
this month
        public static int nDaysInMonth(int month, int year) {

                boolean leap = isLeapYear(year);
                        if (month == 1) return 31;
                        if (month ==2)
                        {
                                if (leap == true) return 29;
                                else return 28;
                        }
                        if (month == 3) return 31;
                        if (month == 4) return 30;
                        if (month == 5) return 31;
                        if (month == 6) return 30;
                        if (month == 7) return 31;
                        if (month == 8) return 31;
                        if (month == 9) return 30;
                        if (month == 10) return 31;
                        if (month == 11) return 30;
                        if (month == 12) return 31;
                        return 0;
                        }
//function that prints a date in the right order
        private static void printDate() {
                if (dayOfWeek == 7)
       System.out.println(dayOfMonth + "/" + month + "/" + year+ " Sunday");
     else   System.out.println(dayOfMonth + "/" + month + "/" + year);
  }

//gets back if a year is a loop year
        public static boolean isLeapYear(int year) {
                boolean isleap;
         isleap =  ((year % 400) == 0);
         isleap = isleap || (((year % 4 ==0) && ((year % 100)!=0)));
         return isleap;
        }
//advane parameters at each end of month
        private static void advance() {
```

```
     dayOfMonth = 1;
  if (month == 12)
  {
    nDaysInMonth = 31;
    month = 1;
    year ++;
  }
 else
 {
    nDaysInMonth = nDaysInMonth(month+1,year);
    month ++;
 }

  return;
    }
public static void main(String args []) {

    int y = Integer.parseInt(args[0]);
    while (year < y+1)
{
    if (nDaysInMonth(month,year) == 31)
    {
        if (year == y)
        {
         printDate();
        }

        if (dayOfWeek == 7)
         {
            dayOfWeek = 1;
         }
         else dayOfWeek ++;
         dayOfMonth ++;

        if (dayOfMonth == 32)
         {
            advance();
         }
    }
    if (nDaysInMonth(month,year) == 30)
    {
        if (year == y)
        {
         printDate();
        }
        if (dayOfWeek == 7)
         {
            dayOfWeek = 1;
```

```
             }
            else dayOfWeek ++;

            dayOfMonth ++;

            if (dayOfMonth == 31)
            {
                advance();
            }
        }

        if (month == 2)
        {
            if (year == y)
            {
             printDate();
            }
            if (dayOfWeek == 7)
              {
                dayOfWeek = 1;
              }
             else dayOfWeek ++;

            dayOfMonth ++;
            if ((isLeapYear(year) == false) && (dayOfMonth == 29))
            {
                advance();
            }
            if ((isLeapYear(year) == true) && (dayOfMonth == 30))
            {
                advance();
            }


    }
}
}
}
```