HW03 – ROMI GOLAN

LoanCalc

```
/**
* Computes the periodical payment necessary to re-pay a given loan.
*/
public class LoanCalc {
      static double epsilon = 0.001;  // estimation error
      static int iterationCounter;    // Monitors the efficiency of the calculation

      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: " + iterationCounter);
      }
    public static double bruteForceSolver(double loan, double rate, int n, double
    epsilon) { //computes ending balance of an n-period loan in bruteForce
            double g = loan / n;
            iterationCounter = 0;
            while (endBalance(loan, rate, n, g) >= 0) {
                  iterationCounter++;
                  g += epsilon;
                  endBalance(loan, rate, n, g);
            }
            return g;
      }
    public static double bisectionSolver(double loan, double rate, int n, double epsilon)
    { //computes ending balance of an n-period loan in biSection
            double l = loan / n; //>0
            double h = loan + 1; //<0
            double g = (l+h)/2;
            iterationCounter = 0;
            while ((h - l) > epsilon) {
                  if (endBalance(loan, rate, n, g) >= 0) {
                        l = g;
                  } else {
                        h = g;
                  }
                  g = (l+h)/2;
                  iterationCounter++;
```

```
            }
            return g;
    }
      private static double endBalance(double loan, double rate, int n, double payment) {
            for (int i = 0; i < n; i++) {
                  loan = (loan - payment) * (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));
    }
    public static String lowerCase(String s) {
        String s1 = s;
        String s2 = "";
        for(int i = 0; i < s1.length(); i++) {
            char letter = s1.charAt(i);
            if((letter <= 'Z') && (letter >= 'A')) {
                s2 = s2 + (char)(s1.charAt(i) + 32);
            } else {
                s2 = s2 + letter;
            }
        }
        return s2;
    }
}
```

UniqueChars

```
/** String processing exercise 2. */
public class UniqueChars {
    public static void main(String[] args) {
        String str = args[0];
        System.out.println(uniqueChars(str));
    }
    public static String uniqueChars(String str) {
        String sNew = "";
        for(int i = 0; i < str.length(); i++) {
            char ch = str.charAt(i);
            if(sNew.indexOf(ch) == -1) {
                sNew += ch;
            } else if(ch == ' ') {
                    sNew += " ";
                }
        }
        return sNew;
    }
}
```

Calender

```
/**
 * Prints the calendars of selected year
 */
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;
      static int nDaysInMonth = 31;

      public static void main(String args[]) {
        int selectYear = Integer.parseInt(args[0]);
            while (year <= selectYear) {
            if(year == selectYear) {
                if(dayOfWeek == 1) {
                            System.out.println(dayOfMonth + "/" + month + "/" + year +
                            "Sunday");
                    } else {
                            System.out.println(dayOfMonth + "/" + month + "/" + year);
                    }
            }
        advance();
        }
      }

      // Advances the date (day, month, year) and the day-of-the-week.
      // Side effects: changes the static variables dayOfMonth, month, year, dayOfWeek,
      nDaysInMonth.
       private static void advance() {
            if(dayOfWeek < 7) {
            dayOfWeek++;
        } else {
            dayOfWeek = 1;
        }
        if(dayOfMonth < nDaysInMonth(month, year)) {
                  dayOfMonth++;
            } else {
                  month++;
                  dayOfMonth = 1;
            }
            if(month == 13) {
               month = 1;
                  year++;
        }
       }

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

```
            return isLeapYear;
      }

      // Returns the number of days in the given month and year.
      private static int nDaysInMonth(int month, int year) {
            int result;
            result = 28;
            if((isLeapYear(year)) && (month == 2)) {
            result = 29;
            } else {
            if((month == 4) || (month == 6) || (month == 9) || (month == 11)) {
                        result = 30;
                  } else if((month == 1) || (month == 3) || (month == 5) || (month == 7)
                  || (month == 8) || (month == 10) || (month == 12)) {
                              result = 31;
                    }
                }
            return result;
      }
}
```