Hw 3 – idan nir

LoanCalc

```
public class LoanCalc {

    static double epsilon = 0.001;
    static int iterationCounter;
    public static void main(String[] args) {

        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);

        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);


        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) {
      double g = loan/n;
      iterationCounter=0;
      while ( endBalance ( loan,  rate,  n,  g) >= epsilon) {
         g += epsilon;
         iterationCounter++;
      }
        return g;
    }


    public static double bisectionSolver(double loan, double rate, int
n, double epsilon) {
        iterationCounter = 0;
        double L = loan / n;
        double H = loan;
        double g = (L + H) / 2;
        while (H - L >= epsilon)
        {
             if (endBalance(loan, rate, n, g) * endBalance(loan, rate,
n, L) > 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) {
        double x = loan;
        for(int i = 0; i<n; i++)
        {
            x -= payment;
            x *= ((100 + rate)/100);
        }
        return x;
    }
}
```

LowerCase

```
public class LowerCase {
    public static void main(String[] args) {
        String str = args[0];
        System.out.println(lowerCase(str));
    }


    public static String lowerCase(String str) {
        String str1 = "";
        for (int i =0;i<str.length(); i++)
        {
            if ((char) str.charAt(i) > (64) && (char) str.charAt(i)
<(91)){
            str1 += (char) (str.charAt(i) + 32);
            }
            else
             str1 += (char) str.charAt(i);
        }
        return str1;
    }
}
```

UniqueChars

```java
public class UniqueChars {
    public static void main(String[] args) {
        String str = args[0];
        System.out.println(UniqueChars(str));
    }


     public static String UniqueChars(String s) {
        String str1 = "" ;
        int length = s.length();

        for (int i =0;i<length; i++)
        {

                if ( (s.charAt(i)) == 32 )
                {
                     str1 += (char)(s.charAt(i));
                }

                else if(str1.indexOf(s.charAt(i)) == -1)
                {
                str1 += (char)(s.charAt(i));
                }

        }
        return str1;
    }
}
```

Calendar

```java
public class Calendar {
    // Starting the calendar on 1/1/1900
    static int dayOfMonth = 1;
    static int month = 1;
    static int year = 1;
    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[]) {
        // Advances the date and the day-of-the-week from 1/1/1900 till
31/12/1999, inclusive.
        // Prints each date dd/mm/yyyy in a separate line. If the day
is a Sunday, prints "Sunday".
        // The following variable, used for debugging purposes, counts
how many days were advanced so far.
        int debugDaysCounter = 0;
        int newyear = Integer.parseInt(args[0]);
        //// Write the necessary initialization code, and replace the
condition
        //// of the while loop with the necessary condition
        while (year!= newyear) {

            advance();
            debugDaysCounter++;
            //// If you want to stop the loop after n days, replace the
condition of the
            //// if statement with the condition (debugDaysCounter ==
n)
            //if (debugDaysCounter == 365) {
            //  break;
            }
             while (year!= newyear + 1) {

                if (dayOfWeek == 1)
                {
                System.out.println(dayOfMonth + "/" + month + "/" +
year + " Sunday");
                }
                else {
                System.out.println(dayOfMonth + "/" + month + "/" +
year);
```

```
                }
                advance();
                 debugDaysCounter++;
                 //// If you want to stop the loop after n days,
replace the condition of the
                 //// if statement with the condition (debugDaysCounter
== n)
                 if (debugDaysCounter ==365) {
                     break;
                 }
        }
        //System.out.println("During the 20th century, " + firstsundays
+ " Sundays fell on the first day of the month");
     }

     // Advances the date (day, month, year) and the day-of-the-week.
     // If the month changes, sets the number of days in this month.
     // Side effects: changes the static variables dayOfMonth, month,
year, dayOfWeek, nDaysInMonth.
     private static void advance() {
        if (dayOfWeek == 7)
        {
            dayOfWeek = 1;
        }
        else
        {
        dayOfWeek++;
        }
        if (month==12 && (nDaysInMonth(month, year) == dayOfMonth))
        {

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

     }

    // Returns true if the given year is a leap year, false otherwise.
    private static boolean isLeapYear(int year) {
```

```java
        boolean isLeapYear;

        isLeapYear = ((year % 400) == 0);

        isLeapYear = isLeapYear || (((year % 4) == 0) && ((year % 100)
!= 0));
        return isLeapYear;
    }

    // Returns the number of days in the given month and year.
    // April, June, September, and November have 30 days each.
    // February has 28 days in a common year, and 29 days in a leap
year.
    // All the other months have 31 days.
    private static int nDaysInMonth(int month, int year) {
        int DaysInMonth = 0;
        switch (month) {
        case 1: DaysInMonth = 31;
        break;
        case 2:  if (isLeapYear(year))
        DaysInMonth = 29;
        else
         {
        DaysInMonth = 28;
         }
        break;
        case 3: DaysInMonth = 31;
        break;
        case 4: DaysInMonth = 30;
        break;
        case 5: DaysInMonth = 31;
        break;
        case 6: DaysInMonth = 30;
        break;
        case 7: DaysInMonth = 31;
        break;
        case 8: DaysInMonth = 31;
        break;
        case 9: DaysInMonth = 30;
        break;
        case 10: DaysInMonth = 31;
        break;
        case 11: DaysInMonth = 30;
        break;
        case 12: DaysInMonth = 31;
        break;
        }
        return DaysInMonth;
    }
}
```