LoanCalc

```
public class LoanCalc {

   static double epsilon = 0.001; // The computation tolerance (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) {
      double g = loan / n;
      iterationCounter = 0;
      while (endBalance(loan, rate, n, g) > 0) {
         g += epsilon;
         iterationCounter++;
      }

      return g;
   }

   public static double bisectionSolver(double loan, double rate, int n, double epsilon) {
      double L = loan / n;
      double H = loan;
```

```
        double g = (L + H) / 2;
        iterationCounter = 0;

        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) {
        for (int i = 1; i <= n; i++) {
            loan = (loan - payment) * (1.00 + (0.01 * rate));
        }
        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) {
        String LowerCase = "";
        for (int i = 0; i < s.length(); i++) {
            char ch = s.charAt(i);
            int ascii = ch;

            if (65 <= ascii && ascii <= 90) {
                int asciiLower = ascii + 32;
                LowerCase = LowerCase + ((char) (asciiLower));
            } else {
                LowerCase = LowerCase + ch;

            }
        }

        return LowerCase;
    }
}
```

UnquieChars

```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 s) {
        String uniqueStr = "";
        for (int i = 0; i < s.length(); i++) {
            if (uniqueStr.indexOf(s.charAt(i)) == -1) {
                uniqueStr = uniqueStr + s.charAt(i);
            } else if (s.charAt(i) == ' ') {
                uniqueStr = uniqueStr + " ";
            }
        }
        return uniqueStr;
    }
}
```

Calendar

```
public class Calendar {
   static int dayOfMonth = 1;
   static int month = 1;
   static int dayOfWeek = 2; // 1.1.1900 was a Monday
   static int nDaysInMonth = 31; // Number of days in January
   static int SundaysCount = 0;
   static boolean isSunday;

   public static void main(String args[]) {
      int year = Integer.parseInt(args[0]);
      advance(year);
   }

   private static void advance(int year) {

      for (int m = month; m <= 12; m++) {
         for (int d = dayOfMonth; d <= nDaysInMonth(m, year); d++) {

            if (dayOfWeek == 1) {
               System.out.println(d + "/" + m + "/" + year + " Sunday");
            } else {
               System.out.println(d + "/" + m + "/" + year);
            }

            dayOfWeek++;
            if (dayOfWeek == 8) {
               dayOfWeek = 1;
            }
         }

      }

   }

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

```
}


private static int nDaysInMonth(int month, int year) {
    int daysNum = 31;

    switch (month) {
        case 1:
            daysNum = 31;
            break;
        case 2:
            if (isLeapYear(year)) {
                daysNum = 29;
            } else {
                daysNum = 28;
            }
            break;
        case 3:
            daysNum = 31;
            break;

        case 4:
            daysNum = 30;
            break;
        case 5:
            daysNum = 31;
            break;

        case 6:
            daysNum = 30;
            break;
        case 7:
            daysNum = 31;
            break;
        case 8:
            daysNum = 31;
            break;
        case 9:
            daysNum = 30;
            break;
        case 10:
            daysNum = 31;
            break;
        case 11:
```

```
            daysNum = 30;
            break;
        case 12:
            daysNum = 31;
            break;
    }

    return daysNum;
  }
}
```