Loan calculations

```
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);

                iterationCounter  = 0;
                // 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;
        double balance = endBalance(loan,rate,n,g);
   while(balance > epsilon){
        iterationCounter++;
        g = g + epsilon;
        balance = endBalance(loan,rate,n,g);
   }
        return g;
   }
public static double bisectionSolver(double loan, double rate, int n, double epsilon) {
   double H = loan;
   double L = 1.0;
   double g = (H + L) / 2.0;
        while (Math.abs(H-L) > epsilon) {
                iterationCounter++;
                if (endBalance(loan,rate,n,g) < 0) {
                        H = g;
                } else {
                        L = g;
                }
                g = (L + H)/ 2.0;
        }
        return g;
   }


        private static double endBalance(double loan, double rate, int n, double payment)
{
                double balance = 0;
                double addRate = (rate + 100)/100;
                for(int i = 0; i < n; i++){
                        balance = (loan - payment) * addRate;
                        loan = balance;
                }
        return balance;
        }
}
```

Lower case

```
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 str = "";
         int n = s.length();

        for(int i = 0; i < n ; i++){
             if(s.charAt(i)>=65 && s.charAt(i)<=90){
                char low = (char)(s.charAt(i) + 32);
                str = str + low;

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

Unique characters

```
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 removed = "";
        int n = s.length();
        for(int i = 0; i < n ; i++){
            char check = s.charAt(i);
            if (check == ' ' || removed.indexOf(check) == -1){
                removed = removed + check;

            }
        }
        return removed;
    }
}
```

Calendar

```
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

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

                }
        }

         private static void advance() {
                dayOfWeek++;
                if(dayOfWeek > 7){
                        dayOfWeek = 1;
                }
                dayOfMonth++;
                if (dayOfMonth > nDaysInMonth) {
                        month++;
                        if(month > 12){
                                year++;
                                month = 1;
                        }
                        dayOfMonth = 1;
                        nDaysInMonth = nDaysInMonth(month,year);
                }
         }

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

```
                return true;
            }
            return false;
        }

        // 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) {
            switch (month) {
                case 4:
                case 6:
                case 9:
                case 11:
                    return 30;
                case 2:
                    if (isLeapYear(year)){
                        return  29;
                    } else  {
                        return 28;
                    }
                default:
                    return 31;
            }
        }
    }
```