```
mport java.util.Set;

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) {
      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;

      while (endBalance(loan, rate, n, g) > 0 ) {

         g = 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;
}

    double x = 0;

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

        x = (loan - payment) * ((rate/100) + 1);

        loan = x;
    }

    return x;

    }
}
```

```java
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 newString = "";

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

            int strNew = str.charAt(i);

            if (65 <= strNew && strNew <= 90 ) {

                strNew = strNew + 32;

                newString = newString + (char) strNew;

            } else {

                newString = newString + str.charAt(i);
            }
        }

        return newString;
    }
}
```

```
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 newString = "";

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

            char char1 = str.charAt(i);

            if (char1 == ' ') {

                newString = newString + " ";

            } else if ((newString.indexOf(String.valueOf(char1)) == -1)) {

                  newString = newString + char1;

            }
        }

        return newString;
    }
}
```

```
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; // Number of days in January

   public static void main(String args[]) {

      int y = Integer.parseInt(args[0]);

      while (year < y) {

         advance();
      }

      while (year == y) {

         if (dayOfWeek == 1) {

            System.out.println(dayOfMonth + "/" + month + "/" + year + " Sunday");

         } else {

            System.out.println(dayOfMonth + "/" + month + "/" + year);
            }
            advance();
         }
      }

    private static void advance() {

      dayOfWeek = (dayOfWeek % 7) + 1;

         dayOfMonth ++;

         if (dayOfMonth > nDaysInMonth(month,year)) {

            dayOfMonth = 1;

            month++;

            if (month > 12) {
                month = 1;

                year++;
            }
        }
    }

private static boolean isLeapYear(int year) {

    if ((year % 4 == 0) || ((year % 100 == 0) && (year % 400 != 0))){

        return true;

    } else {

        return false;
    }
 }

private static int nDaysInMonth(int month, int year) {

    int numbeOfDays;

    if (month == 4 || month == 6 || month == 9 || month == 11) {

        numbeOfDays = 30;

    } else if (month == 2) {

        if (isLeapYear(year)) {

            numbeOfDays = 29;
        } else {

                numbeOfDays = 28;
            }
        } else {

        numbeOfDays = 31;
    }

    return numbeOfDays;

}
}
```