LoanCalc.java

```
    public static double bruteForceSolver(double loan,
            double rate, int n, double epsilon) {

        iterationCounter = 0;

        double startPayment = loan / n;

        while (endBalance(loan, rate, n,
                startPayment) > 0) {
            startPayment += epsilon;
            iterationCounter++;
        }

        return startPayment;
    }

    public static double bisectionSolver(double loan,
            double rate, int n, double epsilon) {

        iterationCounter = 0;

        double highPayment = loan;
        double lowPayment = loan / n;
        double g = (highPayment + lowPayment) / 2.0;

        while (highPayment - lowPayment > epsilon) {

            if (endBalance(loan, rate, n, g) * endBalance(
                    loan, rate, n, lowPayment) > 0) {
                lowPayment = g;
            } else {
                highPayment = g;
            }
            g = (highPayment + lowPayment) / 2.0;
            iterationCounter++;
        }

        return g;
    }
```

```
private static double endBalance(double loan,
        double rate, int n, double payment) {

    double result = loan;

    for (int i = 0; i < n; ++i) {
        result -= payment;
        result *= (1 + rate / 100);
    }

    return result;
}
```

LowerCase.java

```
    public static String lowerCase(String s) {
        // return s.toLowerCase();
        String lowercase = "";

        for (int i = 0; i < s.length(); ++i) {
            if (s.charAt(i) >= 'A' && s.charAt(i) <= 'Z') {
                lowercase += (char) ((int) s.charAt(i)
                        + ((int) 'a' - (int) 'A'));
            } else {
                lowercase += s.charAt(i);
            }
        }
        return lowercase;
    }
```

UniqueChars.java

```
    public static String uniqueChars(String s) {

        String newstring = "";

        for (int i = 0; i < s.length(); ++i) {
            if (newstring.indexOf(s.charAt(i)) == -1
                    || s.charAt(i) == ' ') {
                newstring += s.charAt(i);
            }
        }
        return newstring;
    }
```

Calendar.java

```
public class Calendar {
    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 (Calendar.year < calendarYear + 1) {
            if (Calendar.year == calendarYear) {
                System.out.println(Calendar.dayOfMonth + "/"
                        + Calendar.month + "/"
                        + Calendar.year
                        + (Calendar.dayOfWeek == 1 ? " Sunday"
                                : ""));
            }
            advance();
        }
    }

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

            if (Calendar.month > 12) {
                Calendar.month = 1;
                Calendar.year++;
            }
        }
    }

    private static boolean isLeapYear(int year) {
        return (year % 400 == 0
                || (year % 100 > 0 && year % 4 == 0));
    }
```

```
    private static int nDaysInMonth(int month, int year) {
        switch (month) {
            case 2: // february
                return isLeapYear(year) ? 29 : 28;
            case 4: // april
            case 6: // june
            case 9: // september
            case 11: // november
                return 30;
            default:
                return 31;
        }
    }
}
```