HW3
Question 1

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


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

```
        prevBalance=loan;
        finalBalance=0;

        for( int i = 1; i<=n; i++){
            finalBalance = (prevBalance - payment) * (1 + rate/100);
            prevBalance = finalBalance;
        }
        return finalBalance;
    }
}
```

Question 2

```
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 newString = "";
    for (int i = 0; i < s.length(); i++){
      if (s.charAt(i)<=90 && s.charAt(i)>=65){
        newString = newString + (char)(s.charAt(i) + 32);
      }
      else{
        newString = newString + s.charAt(i);
      }
    }
    return newString;
  }
}
```

Question 3

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

Question 4

```
public class Calendar {
/**
* Prints the calendars of all the years in the 20th century.
*/
   // 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
   static int sundayCounter = 0;

   /**
    * 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;
      //// Write the necessary initialization code, and replace the condition
      //// of the while loop with the necessary condition
      int givenYear = Integer.parseInt(args[0]);
```

```
        while (year != givenYear -1 || month != 12 || dayOfMonth != 31) {

            advance();
            debugDaysCounter++;
            //// If you want to stop the loop after n days, replace the condition of the
            //// if statement with the condition (debugDaysCounter == n)
        }

        advance();
        while(year == givenYear){
            if (dayOfWeek == 1) {
                System.out.println(dayOfMonth + "/" + month + "/" + year + " Sunday");

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

    }
    // 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() {
        int daysInMonth = nDaysInMonth(month, year);
        if (month == 12 && dayOfMonth == 31){
            dayOfMonth = 1;
            dayOfWeek++;
            month = 1;
            year++;
```

```
    } else{
        if (daysInMonth == dayOfMonth){
            dayOfMonth = 1;
            dayOfWeek++;
            month++;
        } else {
            dayOfMonth++;
            dayOfWeek++;
        }
    }
   if (dayOfWeek == 8){
        dayOfWeek = 1;
   }
 }

// Returns true if the given year is a leap year, false otherwise.
private static boolean isLeapYear(int year) {
    if (year%100 == 0 && year%400 !=0){
        return false;
    }
    else if (year % 4 == 0) {
        return true;
    } else {
        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) {
    if (isLeapYear(year) && month == 2) {
        return 29;
    } else {
        if (month == 4 || month == 6 || month == 9 || month == 11) {
            return 30;
        } else if (month == 2) {
            return 28;
        } else {
            return 31;


        }
    }
}
}
```