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)
{
	iterationCounter = 0;
	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  = (H + L)/2;
	  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 balance = 0;
            for(int i = 1; i <= n; i++) {
                balance = (loan - payment)*(1 + rate/100);
                loan = balance;


            }
        return balance;
        }
}
```

LowerCase

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

      return newStr;
   }
}
```

UniqueChars

```
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 newStr = "";

      for (int i = 0; i < s.length(); i++) {
         char c = s.charAt(i);

         if (newStr.indexOf(c) == -1)  {
            newStr += c;
         } else if (c == ' ') {
            newStr += c;
         }
      }

      return newStr;
   }
}
```

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 debugDaysCounter = 0;
           int givenYear = Integer.parseInt(args[0]);
                while (year < givenYear) {
                        advance();
                }
                while(year == givenYear) {
                        System.out.print(dayOfMonth + "/" + month + "/" + givenYear);
                        if(dayOfWeek == 1) {
                                System.out.print(" Sunday");
                        }
                        System.out.println();
                        advance();

                }
         }
         private static void advance() {
                if(dayOfMonth < nDaysInMonth(month,year)) {
                        dayOfMonth++;
                } else {
                        month++;
                        dayOfMonth = 1;
                }
                if(month > 12) {
                        month = 1;
                        year++;
                }
                if(dayOfWeek % 7 != 0) {
                                dayOfWeek++;
                } else {
                                dayOfWeek = 1;
                }

        }
```

```
    private static boolean isLeapYear(int year) {
       if (year % 4 == 0 && year % 100 != 0) {
            return true;
       } else if (year % 400 == 0) {
            return true;
       } else {
            return false;
       }
    }

    private static int nDaysInMonth(int month, int year) {
            if(month == 4 || month == 6 || month == 9 || month == 11) {
                    return 30;
            } else if(month == 2) {
                    if(isLeapYear(year)) {
                            return 29;
                    } else {
                            return 28;
                    }
            }
             else {
                    return 31;
            }
    }
}
```