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) {
                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);

                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);

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

        public static double bisectionSolver(double loan, double rate, int n, double epsilon) {
        iterationCounter = 0;
        double low = loan / n;
        double high = loan;
        double x = (high + low) / 2;
        while ( high - low >= epsilon ) {
            if ( endBalance(loan,rate,n,x) > 0 ) {
                    low = x;
            } else {
                    high = x;
            }
          x = (high + low) / 2;
          iterationCounter++;
         }
            return x;
         }
```

```
	public static double endBalance(double loan, double rate, int n, double payment) {
	double paymentLeft = loan;
	double sum = 0;
	for (int i = 0 ; i < n ; i++ ) {
		sum = (paymentLeft - payment) * ( 1 + rate / 100) ;
		paymentLeft = sum;
	}
		return paymentLeft;
	}
}
```

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 ans = "";
        for(int i = 0 ; i < s.length() ; i++ ) {
              char charI = s.charAt(i);
              if ( charI >= 'A' && charI <= 'Z') {
                   charI = (char) (charI + 32);
              }
              ans += charI;
       }
      return ans;
    }
}
```

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

Calendar

```
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 yearCal = Integer.parseInt(args[0]);
		int debugDaysCounter = 0;
		int countSunday = 0;
		while (year < yearCal ) {
		advance();
		}
		while (year == yearCal) {
			if (dayOfWeek == 1) {
			    System.out.println(dayOfMonth + "/" + month + "/" + year + " Sunday");
			} else {
			    System.out.println(dayOfMonth + "/" + month + "/" + year);
			}
			advance();
			debugDaysCounter++;
			if (debugDaysCounter == 400) {
			    break;
			}
		}
	 }
	 private static void advance() {
		if (dayOfMonth < nDaysInMonth(month,year)) {
		    dayOfMonth ++;
		} else {
			if (month !=12 ) {
			    month ++;
			    dayOfMonth = 1;
			} else {
			    year ++;
			    month = 1;
			    dayOfMonth = 1;
			}
		}
		if (dayOfWeek < 7) {
		    dayOfWeek++;
		} else {
		    dayOfWeek = 1;
		}
	}
```

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

	private static int nDaysInMonth(int month, int year) {
		int days;
		switch (month) {
		case 1: days = 31;
			break;
		case 2: if (isLeapYear(year)) {
			  days = 29;
			  } else {
			  days = 28;
			  }
			break;
		case 3: days = 31;
			break;
		case 4: days = 30;
			break;
		case 5: days = 31;
			break;
		case 6: days = 30;
			break;
		case 7: days = 31;
			break;
		case 8: days = 31;
			break;
		case 9: days = 30;
			break;
		case 10: days = 31;
			break;
		case 11: days = 30;
			break;
		case 12: days = 31;
			break;
		default: days = 0;
			break;
		}
		return days;
	}
}
```