Hw3.amitmanoshevitz

1.LoanCalc

```
public class LoanCalc {

   static double epsilon = 0.001;
   static int iterationCounter;
   static int iterationCounter1;

   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: " + iterationCounter1);


      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;
         iterationCounter1++;
```

```
    }
    return g;
  }



  public static double bisectionSolver(double loan, double rate, int n, double
epsilon) {
       iterationCounter = 0;
    double h = loan;
    double l = 0;
    double g = (h + l) / 2;

    while ((h - l) > epsilon) {
      if ((endBalance(loan, rate, n, l) * endBalance(loan, rate, n, g)) > 0) {
        l = g;
      } else {
        h = g;
      }

      g = (l + h) / 2;
      iterationCounter++;
    }

    return g;
  }


private static double endBalance(double loan, double rate, int n, double
payment) {
  double balance = 0;

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

  return balance;
}
}
```

2.Unique

```
public class UniqueChars {
   public static void main(String[] args) {
      String word = args[0];
      System.out.println(UniqueChars(word)); }
      //// what the method does
      /// checking if the char at the first index is uqule to the second
      /// if it does it'll not be adding them to the new word
      /// if it unique it'll add the char to the new word.
      /// aftere im done it'll move to check if the char at the second index is = to
the first

   public static String UniqueChars(String word) {
      char first = word.charAt(0);
      String newword = first + "";
      int i = 1;
      int len = word.length();

      while(i<len){
         char start = word.charAt(i);
         int j = 0;

         while (j<len){
         char comp = word.charAt(j); //finding the char in the j index to compare
               if (comp == start){
                  break;}
               j++;}

               if (i==j|| start==' '){
               newword = newword + start;}
               i++;} /// adding the unique char to the word
                     return newword;

         }
```

```
}
```

3.LowerCase

```
public class LowerCase {
  public static void main(String[] args) {
    String word = args[0];
  System.out.println(lowerCase(word));

  }

 public static String lowerCase(String word) {
  String newword = "";
      int len = word.length();

      /////// now Im changing from char to ASCII numbers
      ////// when there is a capital letter it is in the range of 65 to 90
      ////// I changeD it to lowercase letters by bringing them 32 tabs in the ASCII
tabel
      ////// the I change it back to char when the number represent a lower class
letter

      for (int i=0;i<len ;i++) {
      char c = word.charAt(i);
      int d = (int)c;
      if (d >= 65 && d <= 90) {
        d = d + 32;
        char e = (char)d;
        newword = newword + e;}

      else{ newword = newword + word.charAt(i);}

      }



   return newword;
```

```
    }
}
```

4. calendar0

```
public class Calendar0 {

    public static void main(String args[]) {
        int year = Integer.parseInt(args[0]);
        isLeapYearTest(year);
        nDaysInMonthTest(year);
    }

    private static void isLeapYearTest(int year) {
        String commonOrLeap = "common";
        if (isLeapYear(year)) {
            commonOrLeap = "leap";
        }
        System.out.println(year + " is a " + commonOrLeap + " year");
    }


    private static void nDaysInMonthTest(int year) {
        for (int month = 1; month <= 12; month++) {
            int daysInMonth = nDaysInMonth(month, year);
            System.out.println("Month " + month + " has " + daysInMonth + " days");
        }
    }


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

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


}
```

calendar1

```
public class Calendar1 {
   // 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;

      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;
   int sundays = 0;

   while (true) {
      System.out.print(dayOfMonth + "/" + month + "/" + year);

      if (dayOfWeek == 1 && dayOfMonth == 1) {
         sundays++;
      }
      if(dayOfWeek == 1){
          System.out.print(" Sunday");
      }

      System.out.println();
      advance();

      debugDaysCounter++;

      if (dayOfMonth == 1 && month == 1 && year == 2000) {
         break;
      }

      }System.out.println("During the 20th century, " + sundays + " Sundays fell on
the first day of the month");
}
   // 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() {
   if ((month == 1 || month == 3 || month == 5 || month == 7 || month == 8 ||
month == 10) && dayOfMonth == 31) {
      month++;
      dayOfMonth = 1;
   } else if ((month == 4 || month == 6 || month == 9 || month == 11) &&
dayOfMonth == 30) {
      month++;
```

```
      dayOfMonth = 1;
    } else if (month == 12 && dayOfMonth == 31) {
      month = 1;
      dayOfMonth = 1;
      year++;
    } else if (!isLeapYear(year) && month == 2 && dayOfMonth == 28) {
      month++;
      dayOfMonth = 1;
    } else if (isLeapYear(year) && month == 2 && dayOfMonth == 29) {
      month++;
      dayOfMonth = 1;
    } else {
      dayOfMonth++;}

      dayOfWeek = (dayOfWeek % 7) + 1;

}


    // Returns true if the given year is a leap year, false otherwise.
    private static boolean isLeapYear(int year) {
      return ((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0);
    }

    // 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.
    // Returns the number of days in the given month and year.
    public static int nDaysInMonth(int month, int year) {
      switch (month) {
        case 1: case 3: case 5: case 7: case 8: case 10: case 12:
          return 31;
        case 4: case 6: case 9: case 11:
          return 30;
        case 2:
          if (isLeapYear(year)) {
          return 29;
          } else {
          return 28;
```

```
            }
          default:
            return -1;
       }
    }
}
```

calendar:

```
public class Calendar {
   static int dayOfMonth = 1;
   static int month = 1;
   static int year = 1900;
   static int dayOfWeek = 2;
   static int nDaysInMonth = 31;

      public static void main(String args[]) {
     int newyear = Integer.parseInt(args[0]);
   int debugDaysCounter = 0;
   int sundays = 0;

    while (true) {
          advance();

          debugDaysCounter++;

          if (dayOfMonth == 1 && month == 1 && year == newyear) {
             break;
          }
       }
       while (true) {
          System.out.print(dayOfMonth + "/" + month + "/" + year);

          if (dayOfWeek == 1 && dayOfMonth == 1) {
             sundays++;
          }
```

```
        if(dayOfWeek == 1){
           System.out.print(" Sunday");
        }

        System.out.println();
        advance();

        debugDaysCounter++;

        if (dayOfMonth == 1 && month == 1 && year == (newyear + 1)) {
           break;
        }
      }
   }


   private static void advance() {
    if ((month == 1 || month == 3 || month == 5 || month == 7 || month == 8 ||
month == 10) && dayOfMonth == 31) {
       month++;
       dayOfMonth = 1;
    } else if ((month == 4 || month == 6 || month == 9 || month == 11) &&
dayOfMonth == 30) {
       month++;
       dayOfMonth = 1;
    } else if (month == 12 && dayOfMonth == 31) {
       month = 1;
       dayOfMonth = 1;
       year++;
    } else if ((month == 2) && (dayOfMonth == 28) && (isLeapYear(year) == false)) {
       month++;
       dayOfMonth = 1;
    } else if (isLeapYear(year) && month == 2 && dayOfMonth == 29) {
       month++;
       dayOfMonth = 1;
    } else {
       dayOfMonth++;}

       dayOfWeek = (dayOfWeek % 7) + 1;
```

```
}


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


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