Göteborg, Magnus Myréen, 2014-10-15

Detta dokument är ett exempel, cirka hälften av en tentamen för
TDA545 Objektorienterad programvaruutveckling

Fulltentamen vitsord: 3=28p, 4=38p, 5= 48p, max 60p.
Max 30p i denna "halvtentamen"

Hjälpmedel: ett utdrag ur Javas API för `String`, `Integer` och `Character` klasserna.

Exempel svar finns med i blå färg.

Uppgift 1: [8 poäng totalt]

1. Förklara *med exempel* hur man anropar en instanmetod. [2 poäng]

   Om en klass `A` har en instansmetod `f`, och `a` är en instans av klassen `A`, då kan man anropa `f` med `a.f(`*`parametrar här`*`)`. Konkret exempel: `c.getRadius()`.

2. Förklara *med exempel* hur man anropar en klassmetod. [2 poäng]

   Om en klass `A` har klassmetod `g`, då kan man anropa g med `A.g(`*`parametrar här`*`)`. Konkret exempel: `Circle.getNumberOfCircles()`.

3. Förklara *med exempel* skillnaden mellan `private` och `public`. [2 poäng]

   Om en klass `A` har en variabel (eller metod) som är `private`, så går det inte att använda variabel (eller anropa metoden) utifrån klassen `A`. Ifall den istället är public, då går det att använda variabel (eller anropa metoden) varifrån som helst.

4. Förklara *med exempel* hur `for` satsen fungerar. [2 poäng]

   `for`-satser har följande form:

   ```
   for (init; guard; inc) { body }
   ```

   Och körs på samma sätt som:

   ```
   init; while (guard) { body; inc }
   ```

   Ett konkret exempel:

   ```
   for (int i=0;i<5;i++) {
     System.out.println("i="+i);
   }
   ```

   Denna for-loop skriver ut följande:

   ```
   i=0
   i=1
   i=2
   i=3
   i=4
   ```

Uppgift 2: [10 poäng totalt]

1. Skriv ett program `CheckPersonNumber` som kollar om ett *finskt* personnummer har korrekt format. Kolla alltså att kontrolltecknet är korrekt och också att det inte finns andra fel (t.ex. att det inte är förkort, har fel slags tecken osv). [10 poäng]

   Här är en beskrivning av hur *finska* personnummer måste se ut:

   > *Personnummer har elva tecken (DDMMÅÅSNNNK). De första sex siffrorna är födelsedatum i ordningen dag-månad-år. Sedan har man en symbol för århundrade, "+" för 1800-talet, "-" för 1900-talet och "A" för 2000-talet. Efter det följer tre siffror (den sista av dem är udda för män och jämn för kvinnor) samt ett kontrolltecken som kan vara en siffra eller en bokstav.*
   >
   > *Det avslutande kontrolltecknet beräknas på följande sätt: De 9 första siffrorna betraktas som ett heltal vilket divideras med 31. Resten vid denna division motsvarar ett tecken i följande lista: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,H,J,K,L,M,N,P,R,S,T,U,V,W,X,Y. Om resten är 10 blir kontrolltecknet A, om resten är 30 blir kontrolltecknet Y.*

   Ditt program bör läsa input från kommandoraden. Exempel:

```
$ java CheckPersonNumber 280634-040Y
correct!

$ java CheckPersonNumber 2806340040Y
wrong!

$ java CheckPersonNumber 280634-040J
wrong!
```

```
public class CheckPersonNumber {

    public static void main(String[] args) {
        // isOK kollar om indata från kommando raden är korrekt
        if (isOK(args[0])) {
            System.out.println("correct!");
        } else {
            System.out.println("wrong!");
        }
    }

    private static boolean isOK(String n) {
        // vi kollar om längden är korrekt
        if (n.length() != 11) {
            return false;
        }
        try {
            // vi läser in siffrona, ifall det inte går så
            // kastar dessa en NumberFormatExcpetion
            int d = Integer.parseInt(n.substring(0,2));
            int m = Integer.parseInt(n.substring(2,4));
            int y = Integer.parseInt(n.substring(4,6));
```

```
        int nnn = Integer.parseInt(n.substring(7,10));
        String sep = n.substring(6,7);
        // vi kollar att det inte fanns minus tecken bland talen.
        if (y < 0 || d < 0 || m < 0  || nnn < 0 ) { return false; }
        // vi sätter årtalet rätt
        if (sep.equals("+")) {
            y = y + 1800;
        } else if (sep.equals("-")) {
            y = y + 1900;
        } else if (sep.equals("A")) {
            y = y + 2000;
        } else {
            // här blev det fel mellantecken
            return false;
        }
        if (!isValidDate(y,m,d)) {
            // hit kommer vi om det var ett felaktigt datum
            return false;
        }
        String k = getControlSign(n);
        // till sist kollar vi om kontrolltecknet är korrekt
        return (k.equals(n.substring(10,11)));
    } catch (NumberFormatException e) {
        return false;
    }
}

private static String getControlSign(String n) {
    // vi läser in de första nio siffrorna som ett tal
    int i = Integer.parseInt(n.substring(0,6) + n.substring(7,10));
    // räknar resten från division med 31
    i = i % 31;
    // ... och så väljer vi det korrekta tecknet ur serien
    String str = "0123456789ABCDEFHJKLMNPRSTUVWXY";
    return str.substring(i,i+1);
}

private static boolean isValidDate(int y, int m, int d) {
    if (m < 0 || 12 < m) { return false; }
    if (d < 0 || daysInMonth(m,y) < d) { return false; }
    return true;
}

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

```
    private static int daysInMonth(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; }
        }
        throw new IllegalArgumentException("not valid month: " + month);
    }

}
```

Uppgift 3: [12 poäng totalt]

1. Skriv en klass `PointInTime` vars instanser representerar tidpunkter, t.ex. 2014-10-01 13:01:05. Klassen bör ha en konstruktor med signaturen:

```
public PointInTime(int year, int month, int day,
                   int hour, int min, int sec)
```

   [2 poäng]

```
public class PointInTime {

    private int y;
    private int m;
    private int d;
    private int h;
    private int i;
    private int s;

    public PointInTime(int year, int month, int day,
                       int hour, int min, int sec) {
        y = year;
        m = month;
        d = day;
        h = hour;
        i = min;
        s = sec;
    }

}
```

2. Skriv en `toString()` metod. [2 poäng]

```
public String toString() {
    return y + "-" + m + "-" + d + " " + h + ":" + i + ":" + s;
}
```

3. Skriv en metod `distanceInSeconds` som räknar hur många sekunder det finns mellan två tidpunkter. Exempel: koden som finns nedan bör skriva ut 55 och 55.

```
PointInTime t1 = new PointInTime(2014,10,1,13,1,5);
PointInTime t2 = new PointInTime(2014,10,1,13,2,0);
System.out.println(t1.distanceInSeconds(t2));
System.out.println(t2.distanceInSeconds(t1));
```

[8 poäng]

Obs! Du får anta att någon annan har skrivit en `isLeapYear` metod med följande signatur.

```
public static boolean isLeapYear(int year)
```

4. ***Obs!*** Du får inte använda `DateTime` eller liknande klasser från Javas API i denna uppgift.

```
private static int daysInMonth(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; }
    }
    throw new IllegalArgumentException("not valid month: " + month);
}

private final long SECS_IN_MIN = 60;
private final long SECS_IN_HOUR = 60 * SECS_IN_MIN;
private final long SECS_IN_DAY = 24 * SECS_IN_HOUR;

private static long secsInMonth(int month, int year) {
    return SECS_IN_DAY * daysInMonth(month, year);
}

private static long secsInYear(int year) {
    if (isLeapYear(year)) {
        return 357 * SECS_IN_DAY;
    } else {
        return 356 * SECS_IN_DAY;
    }
}

// Räknar hur många sekunder det är till datumet från
// datumet 0000-00-00 00:00:00.
private long fromZero() {
    long total = 0;
    for (int k=1; k < y; k++) {
        total = total + secsInYear(k);
    }
    for (int k=1; k < m; k++) {
        total = total + secsInMonth(k,y);
    }
    total = total + SECS_IN_DAY * (d-1);
    total = total + SECS_IN_HOUR * h;
    total = total + SECS_IN_MIN * i;
    total = total + s;
    return total;
}

public long distanceInSeconds(PointInTime p) {
    return Math.abs(p.fromZero() - this.fromZero());
}
```