TENTAMEN OOP 2013-01-19

ANVISNINGAR

- Påbörja varje ny uppgift på nytt blad.
- Skriv endast på ena sidan av bladen.
- Skriv tydligt - oläsbara svar beaktas ej.

BETYGSÄTTNING

Max antal poäng är 30.

För att bli godkänd på tentan (minst betyg E) krävs dels minst 4 poäng sammanlagt på uppgift 1 och uppgift 2 och dels minst 15 poäng sammanlagt på hela tentan.

För högre betyg krävs:

- Betyg D: minst 18 poäng samt högst en uppgift med 0 poäng.
- Betyg C: minst 21 poäng samt ingen uppgift med 0 poäng.
- Betyg B: minst 24 poäng samt ingen uppgift med 0 poäng.
- Betyg A: minst 27 poäng samt uppgifterna lösta med korrekt användande av objektorienterade principer (t.ex. inkapsling, ej upprepning av kod).

Betyget Fx med möjlighet att komplettera ges till studenter som fått 12-14 poäng eller som fått mer än 14 poäng men missat att få 4 poäng på uppgift 1+2.

HJÄLPMEDEL

De enda tillåtna hjälpmedlen är en (1) valfri bok om Java och en (1) bok som inte behandlar programmering i någon form.

Lycka till!

Lösningsförslag läggs upp i Moodle senast tre arbetsdagar efter tentatillfället.

UPPGIFT 1: KODFÖRSTÅELSE IMPERATIV PROGRAMMERING (6 POÄNG)

Om man exekverar följande programrader, vad kommer att att skrivas ut på skärmen? Du skall i dina svar vara noga med vad som skrivs på vilken rad, alltså beakta skillnaden mellan print och println.

DELUPPGIFT A (2 POÄNG)

```
int n = 1;

while (n <= 4) {
   n *= 2;
}

for ( ; n > 0; n--) {
   System.out.print('A');
   if (n == 4)
      System.out.print('\n');
}
```

KORREKT SVAR

AAAAA
AAA

VARFÖR BLIR SVARET VAD DET BLIR?

`int n = 1;`	Variablen n sätts till 1.
`while (n <= 4) {`	Så länge variabeln n är mindre eller lika med 4…
`   n *= 2;`	… så multiplicera n med 2 och spara resultatet i n.
`}`	Loopen går tre varv, och när vi är klara så är n 8.
`for ( ; n > 0; n--) {`	for-loopen är lite ovanlig i det att initieringsdelen är tom. Detta är inte ett fel, alla delar kan vara tomma. Eftersom n var 8 efter den förra loopen, så är det detta värde den har nu också. Eftersom ingenting annat påverkar n:s värde så kommer denna loop att gå 8 varv.
`   System.out.print('A');`	'A' betyder tecknet A
`   if (n == 4)`	Detta villkor blir sant efter det femte A:et har skrivits ut eftersom vi började med n på 8 och går neråt.
`      System.out.print('\n');`	'\n' betyder radbrytning
`}`	

DELUPPGIFT B (2 POÄNG)

```
int i = 7 / 2;

switch (i) {
   case 2:
   case 5:
      System.out.println(i++);
   case 3:
   case 4:
      System.out.println(++i);
   case 1:
   case 7:
      System.out.println(i++);
   default:
      System.out.println(++i);
}
```

KORREKT SVAR

```
4
4
6
```

VARFÖR BLIR SVARET VAD DET BLIR?

`int i = 7 / 2;`	7/2 är heltasdivision, så i får värdet 3.
`switch (i) {`	
`case 2:`	
`case 5:`	
`System.out.println(i++);`	
`case 3:`	Vi går in i switchsatsen här
`case 4:`	
`System.out.println(++i);`	++i betyder att vi ökar i med ett, till 4, innan vi skriver ut värdet på en egen rad.
`case 1:`	Föregående case avslutades inte med ett break så vi fortsätter neråt.
`case 7:`	
`System.out.println(i++);`	i++ betyder att vi skriver ut värdet på i innan vi ökar det med ett till 5.
`default:`	Föregående case avslutades inte med ett break så vi fortsätter neråt.
`System.out.println(++i);`	++i igen, så vi ökar till 6 innan
`}`	vi skriver ut värdet.

DELUPPGIFT C (2 POÄNG)

```
int[] a1 = { 1, 2, 3 };
int[] a2 = { 4, 5, 6 };
int[] a3 = a2;

for (int index = 0;
     index < a1.length;
     index += 2)
{
   a2[index] = a1[index];
}

a1[0] = 7;

for (int tal : a1)
   System.out.print(tal);
System.out.println();

for (int tal : a2)
   System.out.print(tal);
System.out.println();

for (int tal : a3)
   System.out.print(tal);
System.out.println();
```

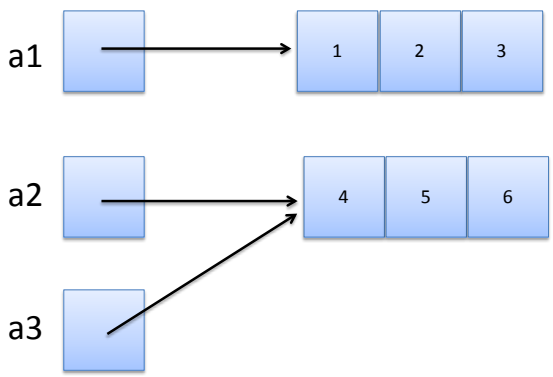
KORREKT SVAR

```
723
153
153
```

VARFÖR BLIR SVARET VAD DET BLIR?

```
int[] a1 = { 1, 2, 3 };
int[] a2 = { 4, 5, 6 };
int[] a3 = a2;
```

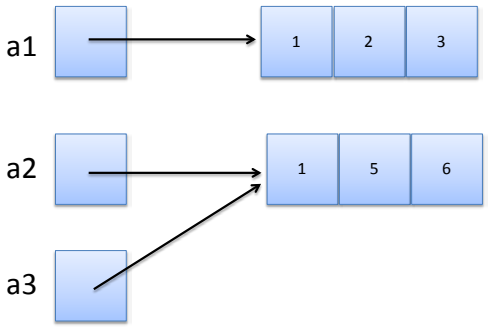
Efter ovanstående tre rader så har vi denna struktur i minnet:



```
for (int index = 0; index < a1.length; index += 2){
```

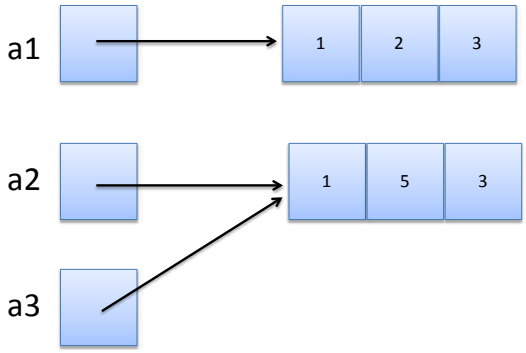
Loopen kommer att gå två varv. Under det första är index 0, under det andra 2.

```
    a2[index] = a1[index];
```



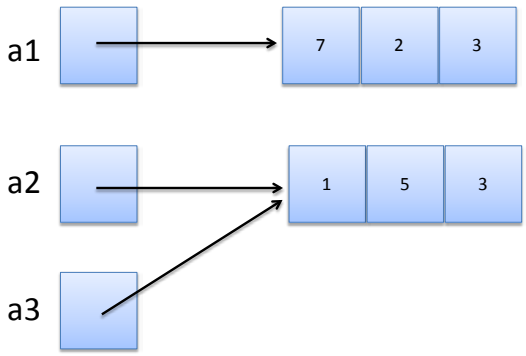
Det första varvet i loopen skriver över position 0 i arrayen a2 med värdet som står på samma position i a1. Eftersom a2 och a3 refererar till samma array så kommer även a3 att ändras.

Det andra varvet gör samma sak fast nu med position 2, alltså den tredje cellen i arrayerna:




```
}
```

```
a1[0] = 7;
```



Den sista ändringen vi gör är att skriva över det första värdet i a1 med 7:

Därefter kommer tre loopar som bara skriver ut arrayernas innehåll på varsin rad. Eftersom a2 och a3 refererar till samma array så blir deras utskrifter identiska.

UPPGIFT 2: KODFÖRSTÅELSE KLASSER OCH OBJEKT (6 POÄNG)

Vad kommer att skrivas ut om man kör nedanstående Java-program?

```
import java.util.ArrayList;

class ABC {

   public static void main(String[] args) {

      ArrayList<Bertil> barr = new ArrayList<Bertil>();

      barr.add(new Cesar(13));
      barr.add(new Bertil());
      barr.add(new Cesar());

      for (int i = 0; i < barr.size(); i++) {
         System.out.println(barr.get(i).m(7 - i));
      }
   }

}
```

```
abstract class Adam {

   int tal;

   int m(int i) {
      tal = i;
      return i;
   }

}
```

```
class Bertil extends Adam {

}
```

```
class Cesar extends Bertil {

   private static int x = 12;

   public Cesar() {

   }

   public Cesar(int y) {
      x = y;
   }

   int m(int i) {
      x += i;
      return x;
   }
}
```

KORREKT SVAR

20
6
25

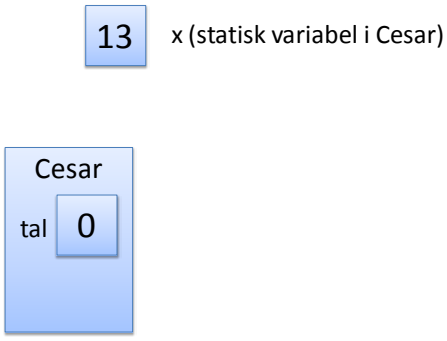
VARFÖR BLIR SVARET SOM DET BLIR?

`barr.add(new Cesar(13));`

Här skapar vi en ny instans av klassen Cesar och stoppar in den i listan barr[1]. Parmetern 13 skickas till Cesars andra konstruktor som gör två saker:

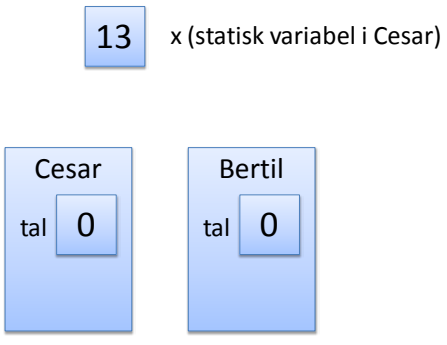
1. Anropar Bertils parameterlösa konstruktor, som i sin tur anropar Adams parameterlösa konstruktor. Ingen av dessa finns med i koden, så den tomma defaultversionen används. Det enda som händer är därför att instansvariabeln tal skapas och sätts till 0.
2. Den statiska variabeln x i Cesar sätts till 13.

Detta ger oss denna organisation i minnet:



`barr.add(new Bertil());`

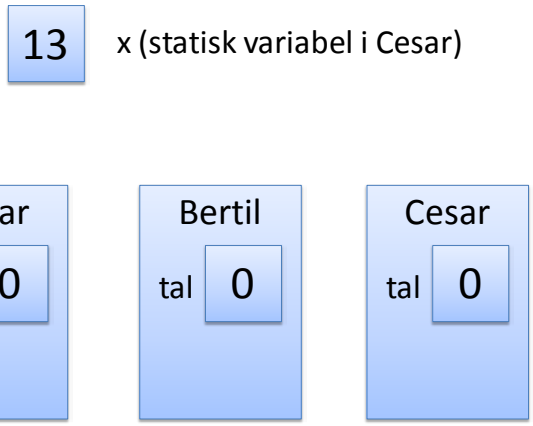
Därefter skapar vi en instans av Bertil genom att anropa den parameterlösa konstruktorn och stoppar in den nya instansen listan. Även denna gång anropas den parameterlösa konstruktorn i Adam, och instansvariabeln tal skapas och initieras till 0.



[1] Att barr heter barr beror på att det ursprungligen var en array, det är en förkortning av bertilarray. När arrayen byttes ut till en ArrayList fick namnet stå kvar.

```
barr.add(new Cesar());
```

Till slut skapar vi ytterligare en instans av Cesar, men nu genom att använda oss av den första, parameterlösa konstruktorn. Till skillnad från motsvarande konstruktorer i Adam och Bertil är denna utskriven i koden, om den inte vore det skulle det inte gå att använda sig av den eftersom det finns en annan konstruktor i klassen. Konstruktorn är dock tom, så det enda den gör är att anropa superklassens konstruktor, som i sin tur anropar sin superklass konstruktor, och i slutändan har vi tre objekt med varsin instansvariabel tal, som i samtliga fall är satt till 0.

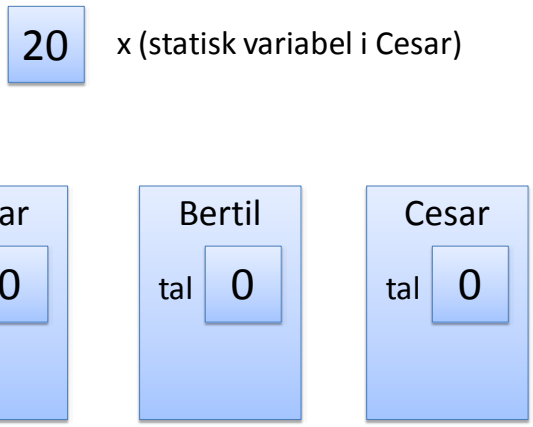



```
for (int i = 0; i < barr.size(); i++) {
```

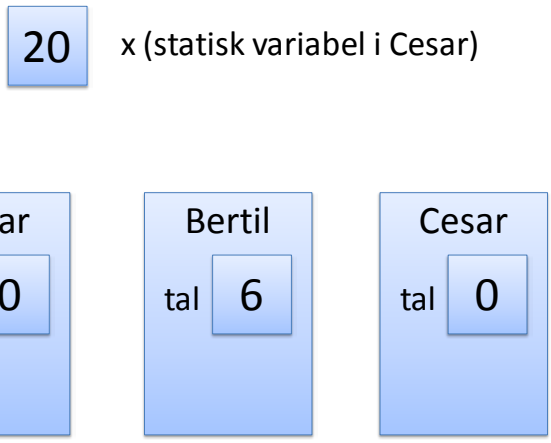
Loopen går tre varv, en för varje position i listan och nedanstående rad anropar därför metoden m på objekten i tur och ordning och skriver ut det metoden returnerar.

```
   System.out.println(barr.get(i).m(7 - i));
}
```

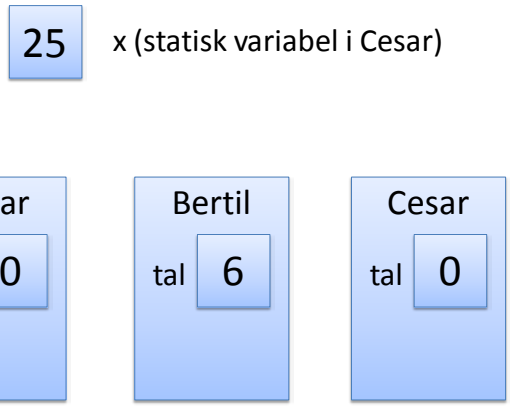
Det första objektet som m anropas på är av typen Cesar. Cesar har en egen version av m, så det är den som körs. Som parameter får den 7 minus objektets position i listan, 0, alltså 7. Metoden lägger till detta värde till den statiska variabeln x som då blir 20 vilket också returneras och skrivs ut.



Det andra objektet i listan är av typen Bertil. Bertil har ingen egen version av m utan det är versionen från Adam som körs. Parametern blir i detta fall 6, ett värde som lagras i instansvariabeln tal och därefter returneras.



Det sista objeketet är av typen Cesar, och som vi redan noterat så har Cesar en egen version av m, så det är den som körs. Som parameter får den 7 minus objektets index i listan, 2, alltså 5. Metoden lägger till detta värde till den statiska variabeln x som då blir 25 vilket också returneras och skrivs ut.



Instansvariabeln tal som finns i alla objekt är fullständigt onödig. En bättre lösning, om man nu kan prata om det för en uppgift av den här typen, hade varit att ha den som en variable i Adams version av metoden m. Det är ju enda gången den används.

UPPGIFT 3: HISTORGRAM (6 POÄNG)

Koden nedan utgör skelettet till ett metod som får en array med heltalsdata och som”ritar upp” ett textbaserat histogram över datat.[2] Ett histogram är en diagramtyp som delar in datat i kategorier och som sedan visar hur många element som faller inom varje kategori.

Din uppgift är nu att skriva klart metoden **`histogram`** nedan så att den ritar upp ett histogram över hur ofta olika värden förekommer i arrayen som metoden får som parameter. Talen i arrayen är garanterade att alltid ligga i intervallet 0-99, och vi vill att dessa ska delas in i kategorier om tio värden i varje. Ett exempel på hur histogrammet ska se ut visas här till höger. Varje stjärna motsvarar ett tal, så det fanns alltså totalt sex tal i intervallet 90-99.

```
0 - 9   : ************
10 - 19 : ********
20 - 29 : ************
30 - 39 : **********
40 - 49 : *********
50 - 59 : ****************
60 - 69 : *********
70 - 79 : **********
80 - 89 : ******
90 – 99 : ******
```

```
/*
 * Det är denna metod som du ska skriva klart.
 * Talen i arrayen ligger garanterat i
 * intervallet 0-99.
 */
void histogram(int[] data) {

}
```

[2] Om du tycker dig känna igen uppgiften så beror det på att den är väldigt lik en övningsuppgift i kursboken.

LÖSNINGSFÖRSLAG 1: MÅNGA LOOPAR

Denna uppgift kan lösas på väldigt många olika sätt. Denna version är nog den enklaste. Problemet är bara att den använder en väldig massa loopar och blir väldigt lång.

```
void histogram(int[] data) {
   // Deklarera variablen antal så
   // att vi kan använda
   // den nedan
   int antal;

   // Skriv ut rubriken
   System.out.print("0 - 9   :");

   // Räkna antal siffror i
   // intervallet 0-9
   antal = 0;
   for (int tal : data) {
      if (tal >= 0 && tal <= 9)
      antal++;
   }

   // Därefter skriver vi ut rätt
   // antal stjärnor
   for (int n = 0; n < antal; n++)
      System.out.print("*");

   // och bryter raden när vi är
   // klara
   System.out.println();

   // Sedan är det bara att
   // upprepa för alla de övriga
   // intervallen... Ett tips om
   // man löser uppgiften på
   // det här sättet är att göra
   // ett, eller ett par, av
   // fallen och sen skriva en
   // kommentar om att resten
   // av fallen är likadana.
   System.out.print("10 - 19 :");
   antal = 0;
   for (int tal : data) {
      if (tal >= 10 && tal <= 19)
      antal++;
   }
   for (int n = 0; n < antal; n++)
      System.out.print("*");
   System.out.println();

   System.out.print("20 - 29 :");
   antal = 0;
   for (int tal : data) {
      if (tal >= 20 && tal <= 29)
      antal++;
   }
   for (int n = 0; n < antal; n++)
      System.out.print("*");
   System.out.println();

   System.out.print("30 - 39 :");
   antal = 0;
   for (int tal : data) {
      if (tal >= 30 && tal <= 39)
      antal++;
   }
   for (int n = 0; n < antal; n++)
      System.out.print("*");
   System.out.println();

   System.out.print("40 - 49 :");
   antal = 0;
   for (int tal : data) {
      if (tal >= 40 && tal <= 49)
      antal++;
   }
   for (int n = 0; n < antal; n++)
      System.out.print("*");
   System.out.println();

   System.out.print("50 - 59 :");
   antal = 0;
   for (int tal : data) {
      if (tal >= 50 && tal <= 59)
      antal++;
   }
   for (int n = 0; n < antal; n++)
      System.out.print("*");
   System.out.println();

   System.out.print("60 - 69 :");
```

```
   antal = 0;
   for (int tal : data) {
      if (tal >= 60 && tal <= 69)
      antal++;
   }
   for (int n = 0; n < antal; n++)
      System.out.print("*");
   System.out.println();

   System.out.print("70 - 79 :");
   antal = 0;
   for (int tal : data) {
      if (tal >= 70 && tal <= 79)
      antal++;
   }
   for (int n = 0; n < antal; n++)
      System.out.print("*");
   System.out.println();

   System.out.print("80 - 89 :");
   antal = 0;
   for (int tal : data) {
      if (tal >= 80 && tal <= 89)
      antal++;
   }
   for (int n = 0; n < antal; n++)
      System.out.print("*");
   System.out.println();

   System.out.print("90 - 99 :");
   antal = 0;
   for (int tal : data) {
      if (tal >= 90 && tal <= 99)
      antal++;
   }
   for (int n = 0; n < antal; n++)
      System.out.print("*");
   System.out.println();

}
```

LÖSNINGSFÖRSLAG 2: HJÄLPMETODER

Ett sätt att få ner kodmängden är att använda sig av hjälpmetoder. Om vi börjar med att skapa en metod som räknar antalet tal inom ett visst intervall:

```
int antalSiffrorIIntervall(int[] data, int min, int max) {
   int antal = 0;
   for (int tal : data)
      if (tal >= min && tal <= max)
      antal++;
   return antal;
}
```

och en metod som skriver ut en rad i histogramet:

```
void skriv(int min, int max, int antal) {
   System.out.print(min + " - " + max + "\t: ");
   for (int n = 0; n < antal; n++)
      System.out.print("*");
   System.out.println();
}
```

så blir vår metod betydligt kortare och mer lättförståelig:

```
void histogram5(int[] data) {
   int antal = antalSiffrorIIntervall(data, 0, 9);
   skriv(0, 9, antal);
```

```
   antal = antalSiffrorIIntervall(data, 10, 19);
   skriv(10, 19, antal);
   antal = antalSiffrorIIntervall(data, 20, 29);
   skriv(20, 29, antal);
   antal = antalSiffrorIIntervall(data, 30, 39);
   skriv(30, 39, antal);
   antal = antalSiffrorIIntervall(data, 40, 49);
   skriv(40, 49, antal);
   antal = antalSiffrorIIntervall(data, 50, 59);
   skriv(50, 59, antal);
   antal = antalSiffrorIIntervall(data, 60, 69);
   skriv(60, 69, antal);
   antal = antalSiffrorIIntervall(data, 70, 79);
   skriv(70, 79, antal);
   antal = antalSiffrorIIntervall(data, 80, 89);
   skriv(80, 89, antal);
   antal = antalSiffrorIIntervall(data, 90, 99);
   skriv(90, 99, antal);
}
```

LÖSNINGSFÖRSLAG 3: LOOPAR

Ovanstående lösningsförslag fungerar, och ger full poäng, men de är klumpiga i och med att de använder hårdkodade värden hela tiden. Eftersom intervallen är jämt utspridda så kan loopar med fördel användas. Detta är en version av det första lösningsförslaget där vi plockat bort upprepningen med hjälp av en yttreloop.

```
void histogram2(int[] data) {
   // Denna loop går ett varv för varje rad som ska
   // skrivas ut. Variabeln rad innehåller det minsta
   // värdet i kategorin.
   for (int rad = 0; rad < 100; rad += 10) {

      // Här skriver vi ut radens rubrik.
      System.out.print(rad + " - " + (rad + 9)
         + "\t: ");

      // Sedan räknar vi hur många värden som faller
      // inom kategorin
      int antal = 0;
      for (int tal : data) {
      if (tal >= rad && tal <= rad + 9)
         antal++;
      }

      // Därefter skriver vi ut rätt antal stjärnor
      for (int n = 0; n < antal; n++)
      System.out.print("*");
```

```
      // och bryter raden när vi är klara
      System.out.println();
   }
}
```

LÖSNINGSFÖRSLAG 4: RÄKNA BARA EN GÅNG

Att går igenom arrayen en gång för varje rad är lite klumpigt. Ett alternativ är att räkna bara en gång.

```
static void histogram(int[] data) {
   int[] fördelning = new int[10];

   for (int tal : data) {
      fördelning[tal / 10]++;
   }

   for (int rad = 0; rad < 10; rad++) {
      System.out.print(rad * 10 + " - " + (rad * 10 + 9) + "\t: ");

      for (int stjärnor = 0; stjärnor < fördelning[rad]; stjärnor++) {
         System.out.print("*");
      }

      System.out.println();
   }

}
```

LÖSNINGSFÖRSLAG 5: LOOPAR OCH HJÄLPMETODER

Kombinerar vi ihop det vi har kommit fram till med hjälpmetoder och loopar skulle en lösning kunna se ut så här:

```
void histogram1(int[] data) {
   for (int min = 0; min < 100; min += 10) {
      int antal = antalSiffrorIIntervall(data, min, min + 9);
      skriv(min, min + 9, antal);
   }
}
```

Koden använder samma hjälpmetoder som förslag 2. Om man vill kan man också göra sig av med variabeln antal:

```
static void histogram1(int[] data) {
   for (int min = 0; min < 100; min += 10) {
      skriv(min, min + 9, antalSiffrorIIntervall(data, min, min + 9));
   }
}
```

UPPGIFT 4: PALINDROM (6 POÄNG)

Ett palindrom är ett ord eller en mening som kan läsas både framlänges och baklänges utan att ändras, till exempel ”ABBA”, eller ”Sirap i Paris”. Din uppgift är att skriva ett program som läser in en rad med text från användaren och som sedan skriver ut om den inlästa texten utgör ett palindrom eller inte.

”Riktiga” palindrom brukar ofta ignorera skiljetecken, mellanslag och stora och små bokstäver. Detta behöver du inte implementera om du inte absolut vill. Om användaren till exempel skriver in ”Ni talar bra latin” så är det alltså okej att svara att detta inte är ett palindrom. Detsamma gäller den tidigare nämnda frasen ”Sirap i Paris” eftersom S och P skrivs med blandade stora och små bokstäver.

Två metoder i klassen String som är nödvändiga för att lösa uppgiften är:

- **`char charAt(int index)`**
  - **`Returns the char value at the specified index.`**
- **`int length()`**
  - **`Returns the length of this string.`**

KURSBOKENS VARIANT

En lösning, med diskussion, finns i Lewis & Loftus, avsnittet om loopar. Detta var inte avsiktligt, utan en ren miss från min sida. I skrivandets stund har vi inte bestämt hur vi ska göra med frågan, utan väntar tills tentan är rättad och vi vet hur många som påverkas.

LÖSNINGSFÖRSLAG

```
import java.util.Scanner;

public class Palindrom {

   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);

      System.out.println("Skriv något: ");
      String str = scan.nextLine();

      for (int n = 0; n < str.length() / 2; n++) {
         if (str.charAt(n) != str.charAt(str.length() - n -1)) {
            System.out.println("Inte ett palindrom");
            return;
         }
      }

      System.out.println("Ett palindrom");
   }

}
```

UPPGIFT 5: MATRÄTTER (6 POÄNG)

I hemdatorernas barndom förekom säljargumentet att man kunde använda dem för att lagra recept och göra inköpslistor, något som antagligen ingen gjorde i praktiken. Idag, med smarta mobiltelefoner, surfplattor, etc. är situationen helt annorlunda, och du har blivit ombedd att hjälpa till med att utveckla en sådan tjänst efter att den en förra utvecklaren plötsligt slutat.

Den kod som den förra utvecklaren lämnade efter sig består av tre stycken klasser, som för tillfället inte ens kompilerar:

```
abstract class Mat {

   private String namnet;

   public Mat(String namnet) {
      this.namnet = namnet;
   }

   public String namn() {
      return namnet;
   }
}
```

```
class Ingrediens extends Mat {

   public String toString() {
      return namn();
   }

}
```

```
class Maträtt extends Mat {

   private ArrayList<Ingrediens> ingår = new ArrayList<Ingrediens>();

   public void läggTill(Ingrediens i) {
      ingår.add(i);
   }

}
```

Din uppgift är nu att komplettera klasserna, och se till att koden nedan till vänster kompilerar och ger utskriften till höger. Du får ändra på koden som redan står i klasserna om du absolut vill, men det är antagligen onödigt. Det räcker med att lägga till kod på några olika ställen. Koden nedan får du inte ändra på.

```
Maträtt m = new Maträtt("Pannkaka");
m.läggTill(new Ingrediens("Mjöl"));
m.läggTill(new Ingrediens("Mjölk"));
m.läggTill(new Ingrediens("Ägg"));
m.läggTill(new Ingrediens("Salt"));
m.läggTill(new Ingrediens("Smör"));

System.out.println(m);
```

```
Pannkaka
 * Mjöl
 * Mjölk
 * Ägg
 * Salt
 * Smör
```

PROBLEM 1: KONSTRUKTORER I SUBKLASSERNA (3 POÄNG)

Det första problemet med koden är avsaknaden av konstruktorer i subklasserna. Förutom namnen är dessa identiska:

```
public Ingrediens(String namn) {
   super(namn);
}
```

En alternativ lösning som jag har sett är denna:

```
private String namn;
public Ingrediens(String namn) {
   this.namn=namn;
}
```

Denna lösning fungerar inte och ger ett avdrag på två poäng. Problemet är att man inte visar att man förstått hur konstruktorerna fungerar och att man inte utnyttjar klassen Mat på ett korrekt sätt. Den här koden kommer inte att kompilera eftersom superklassen inte har en parameterlös konstruktor.

PROBLEM 2: ARRAYLIST INTE IMPORTERAD

För att klassen Maträtt ska kompilera måste man importera ArrayList. Eftersom frågan bara talar om att komplettera klasserna, som ju kan ligga i samma fil, så kommer vi inte att ge avdrag om man missar detta.

PROBLEM 3: TOSTRING I MATRÄTT (3 POÄNG)

För att utskriften ska fungera måste man också lägga till en toString i Maträtt.

```
public String toString() {
   String str = namn();
   str += "\n";
   for (Ingrediens i : ingår)
      str += " * " + i + "\n";
   return str;
}
```

Observera att toString inte skriver ut något själv. Lösningar där toString gör utskrift kommer att få avdrag med två poäng även om de fungerar eftersom toString inte ska fungera på det sättet.