TENTAMEN OOP 2013-04-27

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 i = 5;
int j = ++i;

if (true) {
   i++;
} else {
   j += 2;
}

System.out.println(i + " " + j);
```

KORREKT SVAR

7 6

VARFÖR BLIR SVARET SOM DET BLIR?

`int i = 5;`	**i=5**
`int j = ++i;`	**i=6 sedan sätts j=i, dvs j=6**
`if (true) {`	
`   i++;`	i=7
`} else {`	
`   j += 2;`	Kommer aldrig att köras eftersom villkoret alltid är sant
`}`	
`System.out.println(i + " " + j);`	Skriv ut i (7) följt av ett mellanslag och sist j (6)

DELUPPGIFT B (2 POÄNG)

```
System.out.println("X");
System.out.print(" X\n");
System.out.println("X X\n");
System.out.print("Slut");
```

KORREKT SVAR

X
X
X X

Slut

VARFÖR BLIR SVARET SOM DET BLIR?

\n bryter raden, dvs ett print-anrop som avslutas med \n fungerar ungefär som println, och ett println som avslutas med \n bryter raden två gånger, därav blankraden innan slut.

DELUPPGIFT C (2 POÄNG)

```
String[] årstider = { "vår", "sommar", "höst", "vinter" };

for (int månad = 1; månad <= 12; månad += 3) {
   System.out.println(årstider[månad % 4]);

}
```

KORREKT SVAR

```
sommar
vår
vinter
höst
```

VARFÖR BLIR SVARET SOM DET BLIR?

1%4 = 1 vilket ger sommar
4%4 = 0 vilket ger vår
7%4 = 3 vilket ger vinter
10%4 = 2 vilket ger höst
13 är större än 12 så loopen avslutas

UPPGIFT 2: KODFÖRSTÅELSE KLASSER OCH OBJEKT (6 POÄNG)

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

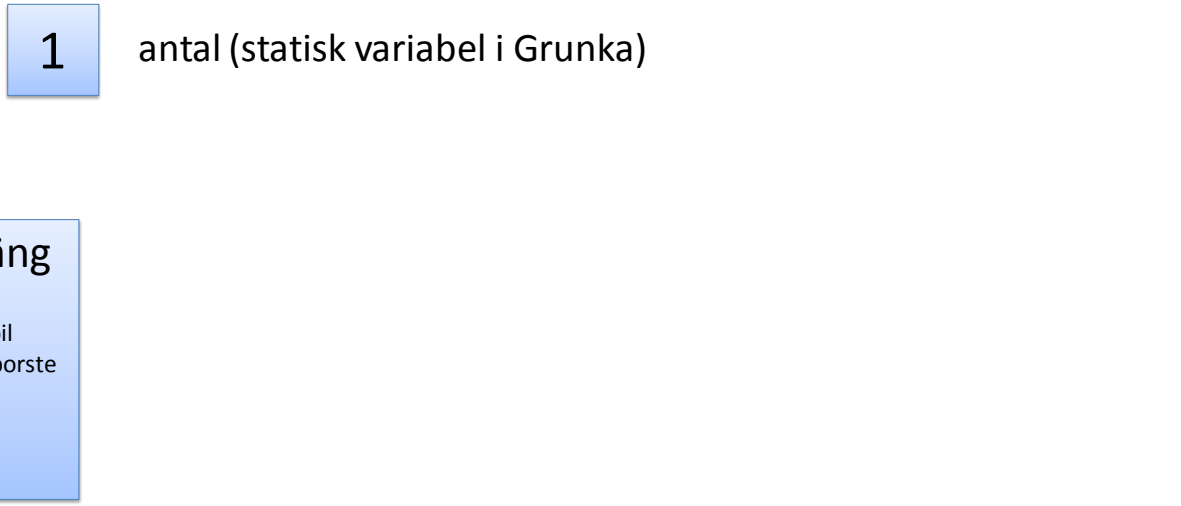
```
class Skrotnisse {
   public static void main(String[] args) {
      Grunka[] alla = new Grunka[4];
      alla[0] = new Mojäng("borste");
      alla[1] = new Sak("spis");
      alla[2] = new Mojäng("sko");
      alla[3] = new Sak("visp");
      for (Grunka g : alla)
         System.out.println(g);
   }
}

class Grunka {
   private static int antal = 0;
   private String grej = "el";

   public Grunka() {
      this("bil");
   }

   public Grunka(String sort) {
      antal++;
      grej = grej + sort;
   }

   public String toString() {
      String tmp = "";
      while (--antal > 0) {
         tmp += grej + " ";
      }
      return tmp;
   }
}

class Mojäng extends Grunka {
   private String namn = "skruv";

   public Mojäng(String str) {
      namn = str;
   }

   public String toString() {
      return namn;
   }
}

class Sak extends Grunka {
   public Sak(String str) {
      super(str);
   }

   public String toString() {
      return "linjal " + super.toString();
   }

}
```

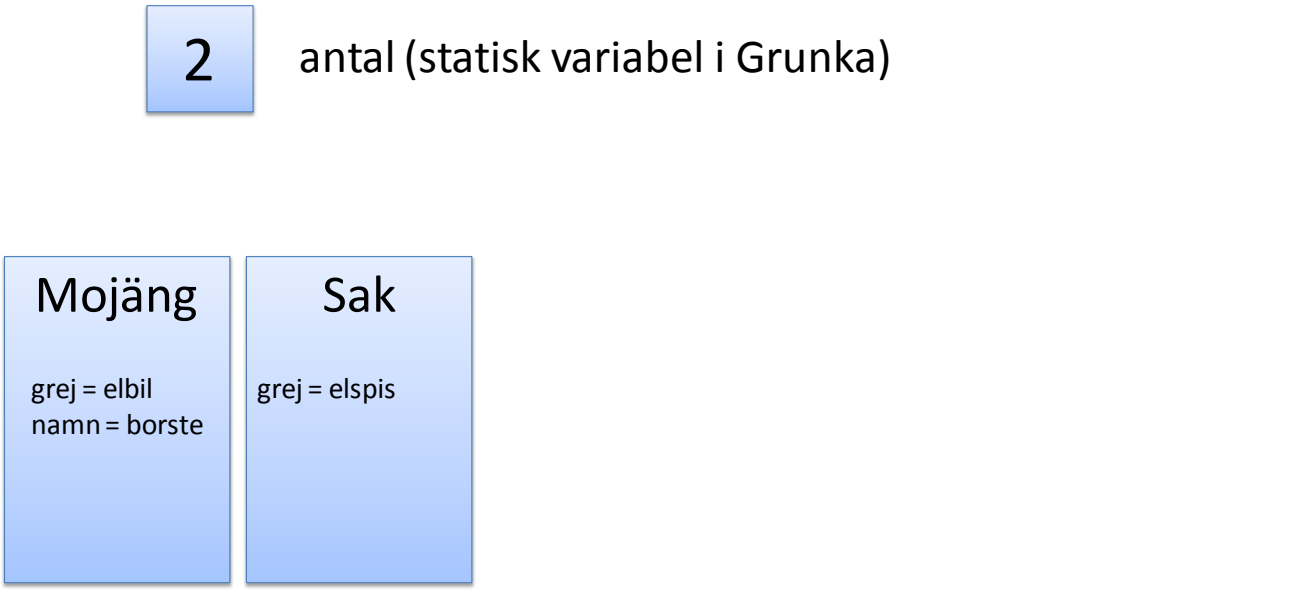
KORREKT SVAR

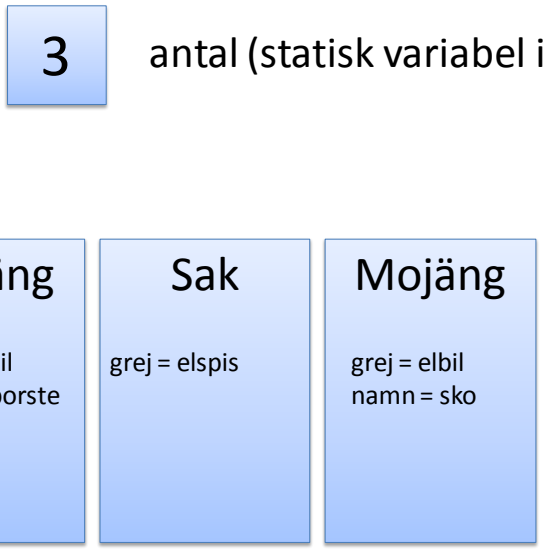
```
borste
linjal elspis elspis elspis
sko
linjal
```

VARFÖR BLIR SVARET SOM DET BLIR?

Följande är vad som händer när vi skapar de fyra objekten







Därefter skriver vi ut de fyra objekten med hjälp av toString. Det första objektet är en Mojäng, som har sin egen toString som bara returnerar instansvariabeln namn:

```
borste
```

Det andra objektet är en Sak som också har en egen toString. Denna returnerar ”linjal ” och resultatet av superklassen Grunkas toString. Grunkas toString bygger upp en sträng av innehåller i instansvariabeln grej. Antalet gånger beror på den statiska variabeln antal. Eftersom vi göra --antal innan vi kontrollerar om den är större än noll så blir det tre kopior av innehåller i grej.

```
linjal elspis elspis elspis
```

Det tredje objektet är en ny Mojäng:

```
sko
```

Det fjärde objektet är en ny Sak. Eftersom antal är statisk har den behållit värdet den fick när vi skrev ut den första saken, så vi får bara ut:

```
linjal
```

UPPGIFT 3: BETYG (6 POÄNG)

Om du tittar på framsidan av tentan så hittar du betygsskalan som används vid rättning. Som du förhoppningsvis redan vet så är sambandet mellan poäng och betyg inte helt okomplicerat, och som examinator önskar man sig då och då ett program som kunde räkna ut betyget åt en. Din uppgift är därför att skriva ett sådant program.

Ett exempel på hur programmet ska fungera är:

```
Poäng på fråga 1: 4
Poäng på fråga 2: 3
Poäng på fråga 3: -1
Felaktig poäng. Intervallet är 0-6
Poäng på fråga 3: 8
Felaktig poäng. Intervallet är 0-6
Poäng på fråga 3: 6
Poäng på fråga 4: 5
Poäng på fråga 5: 4

Poäng: 22
Betyg: C
```

Du ska alltså fråga efter poängen på varje fråga och räkna ut summan samt betyget och skriva ut dessa. Om användaren skriver in ett poängantal som ligger utanför det korrekta intervallet på frågan ska ett felmeddelande skrivas ut och man ska få skriva in ett nytt poängantal. Detta ska upprepas tills man skriver in ett värde i det korrekta intervallet. Du kan anta att användaren alltid skriver in heltal och behöver alltså inte hantera halva poäng, eller att användaren skriver in saker som inte är ett tal överhuvudtaget.

Det extra kravet på betyg A behöver du inte bry dig om.

Tips: villkoren för de olika betygen är inte så komplicerade, men det är lätt gjort att trassla in sig om man inte tar det lugnt. Du bör därför skissa upp hur villkoren hänger samman innan du börjar skriva koden.

LÖSNINGSFÖRSLAG

Det finns flera sätt att kombinera ihop villkoren. En variant skulle vara att ha sju if-satser med alla delar av villkoren för betyget ifråga. Det är dock lätt att det blir grötigt.

```
import java.util.Scanner;

public class Betygsattning {

   private static Scanner scan = new Scanner(System.in);

   private static int läsPoäng(int fråga, int max) {
      System.out.print("Poäng på fråga " + fråga + ": ");
      int poäng = scan.nextInt();
      while (poäng < 0 || poäng > max) {
         System.out.println("Felaktig poäng. Intervallet är 0-" + max);
         System.out.print("Poäng på fråga " + fråga + ": ");
         poäng = scan.nextInt();
      }
```

```
        return poäng;
    }

    public static void main(String[] args) {

        int[] poäng = new int[5];
        int summa = 0;
        int antalNoll = 0;
        for (int fråga = 1; fråga <= 5; fråga++) {
            poäng[fråga - 1] = läsPoäng(fråga, 6);
            summa += poäng[fråga - 1];
            if (poäng[fråga - 1] == 0)
                antalNoll++;
        }

        int summa1och2 = poäng[0] + poäng[1];

        System.out.println("Poäng: " + summa);

        String betyg;

        if (summa1och2 >= 4) {
            if (summa >= 27) {
                betyg = "A";

            } else if (summa >= 24 && antalNoll == 0) {
                betyg = "B";

            } else if (summa >= 21 && antalNoll == 0) {
                betyg = "C";

            } else if (summa >= 18 && antalNoll <= 1) {
                betyg = "D";

            } else if (summa >= 15) {
                betyg = "E";

            } else if (summa >= 12) {
                betyg = "Fx";

            } else {
                betyg = "F";
            }
        } else {
            if (summa >= 12) {
                betyg = "Fx";
            } else {
                betyg = "F";
            }
        }

        System.out.print("Betyg: " + betyg);

    }


}
```

UPPGIFT 4: HELTALSMÄNGD (6 POÄNG)

En mängd är en datasamling i stil med **`ArrayList`** där elementen inte (nödvändigtvis) har någon ordning, och där det inte förekommer dubbletter. Din uppgift är nu att skriva en komplett klass som representerar en mängd av heltal i intervallet 0-1000. Man ska kunna göra tre saker med objekt av klassen:

- Lägga till ett heltal. Om talet redan finns händer inget.
- Ta bort ett heltal. Om talet inte finns händer inget.
- Kunna skriva ut mängden som en sträng

Ett exempel är kanske på sin plats. Om du kallar din klass för Set (engelska för mängd) ska man kunna skapa objekt av den:

```
Set s = new Set();
```

Lägga till tal i den:

```
s.add(43);
s.add(12);
s.add(43);
s.add(831);
```

Försöker man lägga till heltal utanför intervallet 0-1000 så händer ingenting:

```
s.add(-3);
```

Skriva ut den som en sträng:

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

Ovanstående rad skriver ut talen 12, 43 och 831 i någon godtycklig ordning. Observera att 43 bara ska komma med en gång. Dubbletter är ju inte tillåtna i en mängd.

Ta bort tal ur mängden:

```
s.delete(12);
```

Om vi nu skriver ut mängden så ska vi bara få talen 43 och 831 i någon valfri ordning:

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

Tar man bort ett tal som inte finns så händer ingenting. Den sista utskriften nedan ska alltså vara densamma som för ovanstående rad.

```
s.delete(633);
System.out.println(s);
```

LÖSNINGSFÖRSLAG 1: EN ARRAY AV BOOLEANS

```
public class SmallIntSetV1 {

   private static final int MAX = 1000;

   // +1 eftersom vi annars bara får index mellan 0 och 999 och därför inte kan
   // lägga till 1000 till mängden
   private boolean[] numbers = new boolean[MAX + 1];

   private boolean okNumber(int i) {
      return i >= 0 && i <= MAX;
   }

   public void add(int i) {
      if (okNumber(i))
         numbers[i] = true;
   }

   public void delete(int i) {
      if (okNumber(i))
         numbers[i] = false;
   }

   public String toString() {
      String result = "";
      for (int i = 0; i <= MAX; i++) {
         if (numbers[i]) {
            if (result.length() > 0) {
               result += ", ";
            }
            result += i;
         }
      }
      return result;
   }

}
```

LÖSNINGSFÖRSLAG 2: EN ARRAYLIST

```
import java.util.ArrayList;

public class SmallIntSetV2 {

   private static final int MAX = 1000;
   private ArrayList<Integer> numbers = new ArrayList<Integer>();

   private boolean okNumber(int i) {
      return i >= 0 && i <= MAX;
   }

   public void add(int i) {
      if (okNumber(i) && !numbers.contains(i)) {
         numbers.add(i);
      }
   }

   // Konverteringen från int till Integer är viktig eftersom en ArrayList har
   // två remove-metoder. Den ena tar en int som representerar indexet, den
   // andra det objekt som ska tas bort. Om vi inte gjorde konverteringen så
   // skulle den första versionen användas.
   //
   // Att missa en sådan sak på tentan skulle INTE ge några poängavdrag.
   public void delete(int i) {
      if (okNumber(i))
         numbers.remove((Integer) i);
   }

   public String toString() {
      return numbers.toString();
   }


}
```

UPPGIFT 5: BETYGSRAPPORT (6 POÄNG)

På nästa sida hittar du tre klasser som kan användas för att representera betyg: den abstrakta superklassen **`Exam`** och de bägge subklasserna **`GradedExam`** och **`PassFailExam`**. Din uppgift är att skriva en metod som tar emot två parametrar: en **`ArrayList<Exam>`** och en **`String`** som representerar namnet på en kurs. Metodens uppgift är att returnera medelvärdet av alla **`GradedExam`**s på den givna kursen. Om det inte finns några graderade betyg för den givna kursen ska metoden returnera **`0.0`**.

OBS – metoden som du skriver måste ju i Java ligga inuti någon klass. Det kan du strunta i på denna uppgift, det räcker alltså med själva metoden. Observera också att metoden ska returnera värdet, inte skriva ut det.

LÖSNINGSFÖRSLAG

```
static double mean(ArrayList<Exam> exams, String course) {
   int sum = 0;
   int count = 0;

   for (Exam e : exams) {
      if (e.getCourse().equals(course) && e instanceof GradedExam) {
         GradedExam ge = (GradedExam) e;
         sum += ge.getGrade();
         count++;
      }
   }

   if (count == 0)
      return 0.0;
   else
      return ((double) sum) / count;

}
```

```
abstract class Exam {

   private String course;

   public Exam(String course) {
      this.course = course;
   }

   public String getCourse() {
      return course;
   }

   public abstract String getGradeAsText();

}



class GradedExam extends Exam {

   private int grade;

   public GradedExam(String course, int grade) {
      super(course);
      this.grade = grade;
   }

   public int getGrade() {
      return grade;
   }

   public String getGradeAsText() {
      return "" + grade;
   }

}



class PassFailExam extends Exam {

   private boolean passed;

   public PassFailExam(String course, boolean passed) {
      super(course);
      this.passed = passed;
   }

   public boolean getPassed() {
      return passed;
   }

   public String getGradeAsText() {
      if (passed)
         return "Pass";
      else
         return "Fail";
   }

}
```