TENTAMEN OOP 2013-08-08

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 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)

```
for (int i = 0; i < 2; i++) {
   switch (i) {
   case 2:
      System.out.println("Mellan 2 och 1");
   case 1:
      System.out.println("Avbryt");
      break;
   case 0:
      System.out.println("Mellan 0 och default");
   default:
      System.out.println("Default");
   }
}
```

KORREKT SVAR

```
Mellan 0 och default
Default
Avbryt
```

VARFÖR BLIR SVARET SOM DET BLIR?

I det första varvet i loopen är i=0 så vi skriver ut “Mellan 0 och default”. Eftersom det inte finns något break efter detta så faller vi ner till nästa case, i detta fall default, och skriver ut ”Default”.

I det andra varvet är i=1 så vi skriver ut ”Avbryt”. Eftersom detta följs av ett break så hoppar vi ut ur switch-satsen.

DELUPPGIFT B (2 POÄNG)

```
for (int n = 3; n >= 0; n--) {
   for (int m = 1; m < 4; m++) {
      if (m <= n) {
         System.out.print(m);
      }
   }
   System.out.println();
}
```

KORREKT SVAR

```
123
12
1
<BLANK  RAD>
```

VARFÖR BLIR SVARET SOM DET BLIR?

Den yttre loopen går fyra varv. I det första är n=3, så den inre loopen skriver ut talen 123 på samma rad innan den avslutas och en ny radstartas. I det andra varvet av den yttre loopen är n=2, så 12 skrivs ut. I det tredje är n=1 och 1 skrivs ut.

I det fjärde varvet I den yttre loopen är n=0, så if-satsen blir aldrig sann och vi får en blankrad.

DELUPPGIFT C (2 POÄNG)

```
String s1 = "tenta";
String s2 = s1.toUpperCase();
System.out.println(s1 + s2);
```

KORREKT SVAR

```
tentaTENTA
```

VARFÖR BLIR SVARET SOM DET BLIR?

toUpperCase skapar en ny instans av strängen s1 med enbart stora bokstäver. Den ändrar INTE på s1. println ger en radbrytning efter att den sammanslagna strängen skrivits ut, inte mellan dem.

UPPGIFT 2: KODFÖRSTÅELSE KLASSER OCH OBJEKT (6 POÄNG)

Givet nedanstående tre klasser, vad kommer att skrivas ut om man kör Java-programmet på nästa sida?

```
abstract class Apa {
   private String namn;

   public Apa(String namn) {
      this.namn = namn;
   }

   public String toString() {
      return namn;
   }

}

class Gorilla extends Apa {

   private int vikt;

   public Gorilla(String namn, int vikt) {
      super(namn);
      this.vikt = vikt;
   }

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

class Schimpans extends Apa {

   private static int ålder = 0;

   public Schimpans(int ålder) {
      this("Cheeta", ålder);
   }

   public Schimpans(String namn, int ålder) {
      super(namn);
      Schimpans.ålder += ålder;
   }

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

```java
import java.util.ArrayList;

public class Apor {

   public static void main(String[] args) {
      ArrayList<Apa> apor = new ArrayList<Apa>();

      apor.add(new Schimpans(80));
      apor.add(new Gorilla("Kong", 1234));
      apor.add(new Schimpans("Ola", 26));

      for (Apa a : apor) {
         System.out.println(a);
      }
   }

}
```

KORREKT SVAR

```
Cheeta 106
Kong 1234
Ola 106
```

VARFÖR BLIR SVARET SOM DET BLIR?

Programmet skapar tre apor och lägger dem i en ArrayList. Den första apan är en schimpans som är 80 år gammal. Den första konstruktorn i klassen Schimpans tar emot åldern som parameter och skickar den vidare till den andra konstruktorn tillsammans med ett default-namn: Cheeta. Den andra konstruktorn skickar namnet (Cheeta) till superklassens konstruktor där det sparas och lägger till åldern (80) till den statiska variabeln ålder i klassen Schimpans.

Den andra apan är en gorilla som heter Kong och som väger 1234kg. Konstruktorn i Gorilla gör nästan precis samma sak som den andra konstruktorn i Schimpans: namnet skickas till superklassens konstruktor och vikten sparas, fast nu i en instansvariabel.

Den tredje apan är åter en schimpans. Denna har ett eget namn (Ola) och använder därför den andra konstruktorn direkt. Åldern 26 läggs till till den statiska variabeln ålder som nu alltså har värdet 80+26=106.

Därefter skriver vi ut alla aporna i en loop varvid toString anropas på de olika objekten. Både Gorilla och Schimpans använder sig av superklassen Apa:s toString för att få reda på namnet.

UPPGIFT 3: ANTAL ORD (6 POÄNG)

Skriv ett program som ber användaren skriva in en mening och som därefter skriver ut hur många ord meningen innehåller. Ord separeras av en eller flera av mellanslag, punkt, komma, frågetecken och utropstecken.

Ett exempel på hur programmet ska fungera är detta:

```
Skriv en mening: Här är en mening.

Meningen består av 4 ord
```

I ovanstående exempel så finns det precis en avdelare efter varje ord. Som det står i uppgiften kan det dock finnas flera, och de kan dessutom komma innan meningen startar:

```
Skriv en mening: ,. Här   är en!?konstig mening
Meningen består av 5 ord
```

Två metoder i klassen String som kan användas 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.**

LÖSNINGSFÖRSLAG

```
import java.util.Scanner;

public class WordCount1 {

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

      System.out.print("Skriv en mening: ");
      String str = keyboard.nextLine();

      // Antal ord i den inskrivna meningen
      int words = 0;

      // Innan vi har sett några tecken så kan vi inte ha påbörjat ett ord.
      boolean wordStarted = false;

      // Gå igenom varje tecken i strängen
      for (int n = 0; n < str.length(); n++) {
         switch (str.charAt(n)) {
         case ' ':
```

```
            case '.':
            case ',':
            case '!':
            case '?':
               // Om vi stöter på något av avgränsningstecknen *PRECIS EFTER*
               // att vi/ stött på något annat tecken så vet vi att ett ord
               // just avslutats och kan därför räkna upp variabeln words.
               if (wordStarted) {
                  words++;
                  // Här sätter vi tillbaka wordStarted till false så att vi
                  // inte dubbelräknar ord som följs av flera avgränsare.
                  wordStarted = false;
               }
               break;
            default:
               // Alla andra tecken vi stöter på måste vara en del av ett ord
               // så vi noterar att vi sett detta.
               wordStarted = true;
               break;
            }
         }

         // Om det sista ordet inte avslutades med en avdelare så kommer det inte
         // räknas av loopen ovan.
         if (wordStarted) {
            words++;
         }

         System.out.println("Meningen består av " + words + " ord");
      }

}
```

ALTERNATIVA LÖSNINGSVARIANTER

Ett alternativt sätt att lösa uppgiften är att använda metoden split i String. Denna metod är något enklare, men kräver att man känner till metoden och hur den fungerar.

UPPGIFT 4: TID (6 POÄNG)

Denna uppgift går ut på att skriva en klass[1] som representerar en tidpunkt inom ett dygn. Följande krav finns:

- Tidpunkten representeras med två heltal, ett för timmar och ett för minuter.
- Klassen ska ha två konstruktorer. Den ena ska vara parameterlös och sätta tiden till 00:00, och den andra ska ta två parametrar: timmar och minuter. Du kan anta att den som använder klassen alltid kommer att ange korrekta tidpunkter till den andra konstruktorn och behöver därför inte lägga in någon felhantering i denna.
- Det ska gå att läsa av timmar och minuter som heltal med hjälp av metoder, men inte ändra på dem på annat sätt än att skapa ett nytt objekt.
- Det ska gå att omvandla objekt av klassen till textsträngar på Javas vedertagna sätt med en toString-metod.

LÖSNINGSFÖRSLAG

```
public class Time {

   private int hour;
   private int minutes;

   public Time() {
      // Vi behöver inte anropa den andra konstruktorn eftersom hour och
      // minutes är satta till 0 per default.
   }

   public Time(int hour, int minutes) {
      this.hour = hour;
      this.minutes = minutes;
   }

   public int getHour() {
      return hour;
   }

   public int getMinutes() {
      return minutes;
   }

   public String toString() {
      return hour + ":" + minutes;
   }

}
```

[1] Observera att uppgiften bara ber dig att skriva en klass. Du ska alltså INTE skriva ett program som använder sig av din klass.

UPPGIFT 5: DRAKTÅR (6 POÄNG)

En egenskap som skiljer en kejserlig kinesisk drake från andra kinesiska drakar är att den har fem tår på varje fot istället för det normala tre eller fyra. Vi vill nu skapa klasser som kan representera detta och har därför skapat de tre klasserna på nästa sida. Din uppgift är nu att kompletter dessa klasser så att toString-metoden i klassen Drake fungerar. Till exempel ska nedanstående två rader:

```
System.out.println(new KejserligDrake("Huangdi"));
System.out.println(new KinesiskDrake("Long"));
```

ge följande utskrift:

```
Huangdi är en drake med fem tår.
Long är en drake med tre eller fyra tår.
```

En restriktion i uppgiften är att du INTE får ändra på någonting av det som redan står i klasserna. Du får lägga till hur många metoder, variabler, etc. du anser nödvändigt, men inte lägga till några parametrar till redan existerande metoder, ändra typer på variabler, etc. Du får inte heller överskugga toString i subklasserna. Det var ju toString i klassen Drake som skulle fungera.

```
abstract class Drake {

   private String namn;

   public Drake(String namn) {
      this.namn = namn;
   }

   public String namn() {
      return namn;
   }

   public String toString() {
      return namn + " är en drake med " + tår() + " tår.";
   }
}
```

```
class KinesiskDrake extends Drake {

   public KinesiskDrake(String namn) {
      super(namn);
   }

}
```

```
class KejserligDrake extends KinesiskDrake {

   public KejserligDrake(String namn) {
      super(namn);
   }

}
```

LÖSNINGSFÖRSLAG

```
abstract class Drake {

   private String namn;

   public Drake(String namn) {
      this.namn = namn;
   }

   public String namn() {
      return namn;
   }

   // Metoden tår MÅSTE finnas i klassen Drake, annars skulle vi inte kunna
   // anropa den i toString. Eftersom antalet tår varierar mellan olika
   // draktyper så gör vi den abstrakt här.
   protected abstract String tår();

   public final String toString() {
      return namn + " är en drake med " + tår() + " tår.";
   }
}

class KinesiskDrake extends Drake {

   public KinesiskDrake(String namn) {
      super(namn);
   }

   // Sedan implementerar vi den här
   public String tår() {
      return "tre eller fyra";
   }

}

class KejserligDrake extends KinesiskDrake {

   public KejserligDrake(String namn) {
      super(namn);
   }

   // och här
   public String tår() {
      return "fem";
   }


}
```