Ordinarie tentamen för ID1004 Objektorienterad programmering, 15 mars 2016

Denna tentamen examinerar 3.5 högskolepoäng av kursen.

Inga hjälpmedel är tillåtna.

Tentamen består av en obligatorisk del och en extra del. För ett godkänt betyg på hela tentamen måste den obligatoriska delen vara godkänd. Detta berättigar till betyg E på tentamen. För högre betyg kan ytterligare uppgifter lösas i den extra delen. Examinator förbehåller sig möjligheten att justera tentamensbetyget med hänsyn till en bedömning av helhetsintrycket.

Den obligatoriska delen ger maximalt 35 poäng. För godkänt krävs 20 poäng eller mer, varav minst 4 poäng på fråga 7.

Läs noga igenom alla frågorna först och planera tiden.

Om inget annat uttryckligen sägs, så visas och efterfrågas kod och uttryck ur Java version 6 eller senare. Var noga med metoders returtyper. Begreppen array och vektor är synonyma.

Lösningsförslag av Fredrik Kilander

Obligatorisk del

Fråga 1 (3p)

Namnge, deklarera och initiera dessa variabler:

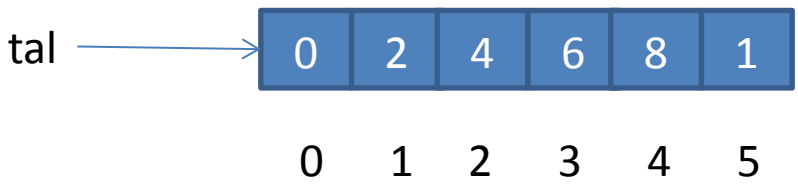
a) En teckenvariabel med värdet `'T'`.
```
char c = 'T';
```
b) En flyttalsvariabel med halva värdet av roten ur 2.
```
double d = Math.sqrt(2.0) / 2.0;
```
c) En heltalsvariabel vars första värde är ett mindre än innehållet i MAX_ANTAL.
```
int i = MAX_ANTAL - 1;
```

Fråga 2 (5p)

Givet denna kod:

```
int [] tal = new int[6];
for (int i = 0; i < tal.length; i++)
  tal[i] = i * 2;
```

a) Rita arrayen tal och dess innehåll: Rita referensvariabeln, referensen, objektet som refereras, markera varje elements index och varje elements innehåll.



Fråga 3 (2p)

Deklarera en array med plats för MAX_ANTAL textsträngar.

```
String [] sa = new String [MAX_ANTAL];
```

Fråga 4 (6p)

De 52 korten i en kortlek numreras 0-51 enligt följande schema:

```
 0,  1,  2, … , 12, // Hjärter ess, två, tre, … , kung
13, 14, 15, … , 25, // Klöver  ess, två, tre, … , kung
26, 27, 28, … , 38, // Ruter   ess, två, tre, … , kung
39, 40, 41, … , 51  // Spader  ess, två, tre, … , kung
```

Givet att variabeln `int kort` innehåller ett tal 0-51:

a) Skriv ett villkor som är sant (ger värdet `true`) om kortet har färgen klöver.

```
(kort / 13) == 1
```

b) Skriv ett villkor som är sant (ger värdet `true`) om kortet har valören kung.

```
(kort % 13) == 12
```

c) Skriv ett villkor som är sant (ger värdet `true`) om kortet är rött (hjärter eller ruter)

```
((kort / 13) == 0) || ((kort / 13) == 2)
```

alternativt

```
((kort / 13) % 2) == 0
```

Fråga 5 (4p)

Skriv en metod `blanda` som tar emot en array av typen `char`, och slumpmässigt kastar om ordningen på elementen. Använd en `for`-loop för att gå igenom arrayen en gång. I varje steg, välj slumpmässigt ett av elementen i arrayen och byt plats på det och det aktuella elementet.

En slumptalsgenerator `Random rnd` kan antas finnas tillgänglig. Metodanropet `rnd.nextInt(n)` returnerar ett heltal r, 0 <= r < n.

```
void blanda (char [] ca) {
  for (int i = 0; i < ca.length; i++) {
    int r = rnd.nextInt(ca.length);
    char temp = ca[i];
    ca[i] = ca[r];
    ca[r] = temp;
  }
}
```

Fråga 6 (6p)

I en påse finns två blå, tre röda, fyra gula och ett svart piller. Använd en array av `char` för att modellera påsen. Anropa metoden `blanda` från fråga 5 för att randomisera elementens ordning. Använd en `while`-loop för att dra tre piller ur påsen utan återläggning, och skriva ut deras färg. Skriv Java-kod för att visa hur det kan göras.

```
char [] piller = {'b', 'b', 'r','r','r','g','g','g','g','s'};
blanda(piller);
int antal = 0;
while (antal < 3) {
```

```
  System.out.println (piller[antal++]);
}
```

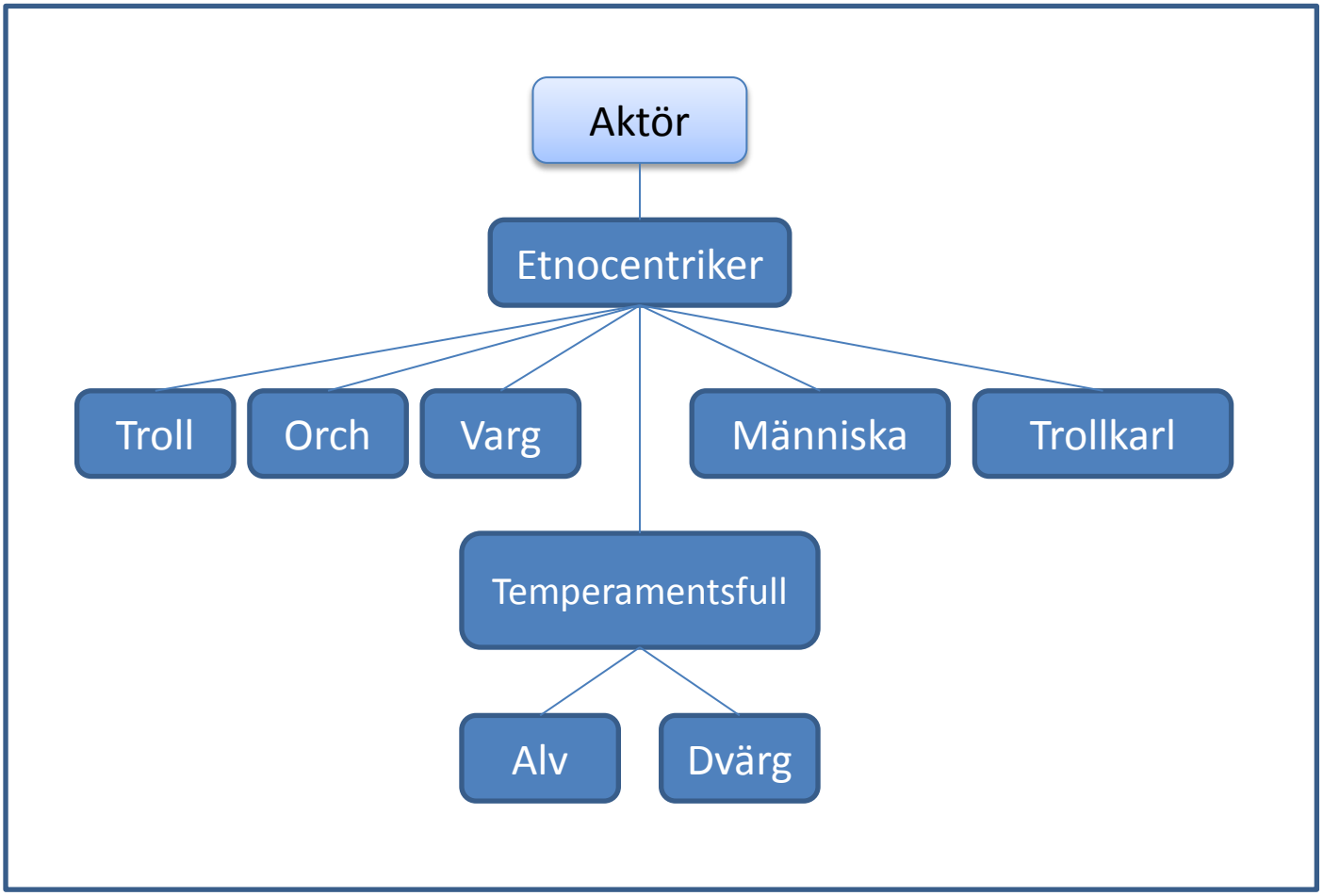
Fråga 7 (9p)

Med hjälp av gränssnitt, klasser och/eller arv, modellera följande:

I en spelvärld finns följande aktörer: Troll, alver, orcher, dvärgar, vargar, trollkarlar och människor. Alver, människor och trollkarlar är alltid vänner. Människor, trollkarlar och dvärgar är alltid vänner. Alver och dvärgar är vänner eller fiender beroende på om individens humör är mindre än 0.2. Troll, orcher, och vargar är bara vänner med sin egen sort, och fiender med alla andra.

När två aktörer möts genomför de en förhandling för att se om de är vänner eller fiender. Denna förhandling går till så att båda aktörerna anropar metoden `antagonist` i den andra aktören och sänder med en referens till sig själv som parameter. Aktören svarar `true` om det är en fiende som anropar och `false` om det är en vän.

a) Skapa en klasshierarki för ovanstående modell. Det är inte nödvändigt att skriva full Java-kod, det går bra att rita diagram också, men beskrivningen måste gå att översätta till Java-kod utan att behöva gissa för mycket.
b) Skapa metoden `antagonist` för alver, orcher och trollkarlar.
c) Skriv ett stycke Java-kod som skapar tre aktörer och låter var och en av dem anropa metoden `antagonist` i de andra aktörerna.



```
abstract class Aktör {
  public abstract boolean antagonist (Aktör a);
}

// Etnocentriker tycker bara om sin egen sort
class Etnocentriker extends Aktör {
  public boolean antagonist (Aktör a) {
    return !(a.getClass().equals(getClass()));
  }
}

class Troll extends Etnocentriker {}
class Orch extends Etnocentriker {}
class Varg extends Etnocentriker {}

// Människor och trollkarlar är lite mer vidsynta
class Människa extends Etnocentriker {
  public boolean antagonist (Aktör a) {
    if (a instanceof Alv ||
          a instanceof Dvärg ||
          a instanceof Trollkarl)
      return false;
    else
      return super.antagonist(a);
  }
}

class Trollkarl extends Etnocentriker {
  public boolean antagonist (Aktör a) {
    if (a instanceof Alv ||
          a instanceof Dvärg ||
          a instanceof Människa)
      return false;
    else
      return super.antagonist(a);
  }
}

// Temperamentsfulla aktörer har även humör
class Temperamentsfull extends Etnocentriker {
  protected double humör = 0.4 * Math.random();
}

class Alv extends Temperamentsfull {
  public boolean antagonist (Aktör a) {
    if (a instanceof Människa ||
          a instanceof Trollkarl)
      return false;
    else if (a instanceof Dvärg)
      return humör < 0.2;
    else
      return super.antagonist(a);
```

```
  }
}

class Dvärg extends Temperamentsfull {
  public boolean antagonist (Aktör a) {
    if (a instanceof Människa ||
           a instanceof Trollkarl)
      return false;
    else if (a instanceof Alv)
      return humör < 0.2;
    else
      return super.antagonist(a);
  }
}

public class Fraga7 {

  public static void main (String [] args) {
    Aktör [] aktörer = {
      new Orch(),
      new Människa(),
      new Orch(),
      new Alv(),
      new Trollkarl(),
      new Dvärg(),
      new Människa()
    };

    for (int i = 0; i < aktörer.length - 1; i++) {
      for (int j = i + 1; j < aktörer.length; j++) {
           System.out.printf ("%s(%d) är %s med %s(%d)%n",
                                  aktörer[i].getClass().getName(),
                                  i,
                                  aktörer[i].antagonist(aktörer[j]) ?
fiende" : "vän",
                                  aktörer[j].getClass().getName(),
                                  j);

           System.out.printf ("%s(%d) är %s med %s(%d)%n",
                                  aktörer[j].getClass().getName(),
                                  j,
                                  aktörer[j].antagonist(aktörer[i]) ?
fiende" : "vän",
                                  aktörer[i].getClass().getName(),
                                  i);
           System.out.println();
      }
    }
  }
}
```

Extra del

Med godkänd obligatorisk del, ger poäng på extradelen betyg: 5-6 D, 7-8 C, 9-11 B och 12-14 A.

Fråga 8 (6p)

Klassen `Complex` representerar ett komplext tal, det vill säga ett tal z = (a + bi), där a är realdelen och b imaginärdelen:

```
class Complex {
  public final double a;
  public final double b;

  public Complex (double a, double b) {
    this.a = a;
    this.b = b;
  }

  public Complex add (Complex other) {
    return new Complex (a + other.a, b + other.b);
  }

  public String toString () {
    return "(" + a + " + " + b + "i)";
  }
}
```

Klassen `Complex` kan användas på detta sätt:

```
Complex z1 = new Complex (1, 2);
Complex z2 = new Complex (3, 4);
Complex z3 = z1.add(z2);
System.out.println(z3);
```

När ovanstående kodfragment exekveras erhålls följande utskrift:

```
(4.0 + 6.0i)
```

Addition av två komplexa tal z1=(a+bi), z2=(c+di) definieras som:

z1 + z2 = z3

(a + bi) **+** (c + di) = (a **+** c) + (b **+** d)i

a) Skriv färdigt metoden `add`, enligt den givna definitionen. Tänk på att tecknet ’+’ har två olika betydelser här: dels som addition, och dels som avdelare mellan realdelen och imaginärdelen.

b) Skriv färdigt metoden `toString`.

Fråga 9 (8p)

Betrakta interface-klassen `SelectString` som kan användas för att välja ut strängar:

```
public interface SelectString {
  public boolean selected(String s);
}
```

a) Skriv en klass som implementerar `SelectString` så att den väljer ut strängar som är längre än 5 tecken och vars första tecken är en bokstav.

```
class MyChoice implements SelectString {
  public boolean selected (String s) {
    return (5 < s.length()) && Character.isLetter(s.charAt(0));
  }
}
```

b) Den statiska metoden `String [] selectedStrings(String [] sa, SelectString sel)` returnerar en array med utvalda strängar. Skriv metoden `selectedStrings`.

```
static String [] selectedStrings (String [] sa, SelectString sel){
  int count = 0;
  for (String s : sa)
    if (sel.selected(s))
      count++;

  String [] rtn = new String[count];

  count = 0;
  for (String s : sa)
    if (sel.selected(s)
      rtn[count++] = s;

  return rtn;
}
```

c) Skapa en array med strängar och anropa metoden `selectedStrings` tillsammans med en instans av klassen från deluppgift a.

```
String [] myStrings = {"foo", "baaaar", "123"};
String [] rs = selectedStrings(myStrings, new MyChoice());
```

Bilaga

Följande dokumentation kan vara till hjälp att förstå och lösa vissa uppgifter.

`char`	**`String.charAt`**`(int index)` Returns the `char` value at the specified index.
`int`	**`String.length`**`()` Returns the length of this string.
`static boolean`	**`Character.isLetter`**`(char ch)` Determines if the specified character is a letter.
`static double`	**`Math.sqrt`**`(double a)` Returns the correctly rounded positive square root of a `double` value.
`Class``<?>`	**`Object.getClass`**`()` Returns the runtime class of this `Object`.