Övningstenta för TDA548

1. Förklara med en eller ett par meningar. Du får gärna ge kodexempel eller rita. 4p

   a) Sats (statement)

   b) Synlighetsområde (scope)

2. Vilka av raderna nedan ger kompileringsfel? Motivera! 2p

```
int[] ia = {1, 2, 3};      // 1
double[] da = {1, 2, 3};   // 2
ia = da;                   // 3
ia = (int[]) da;           // 4
da = ia;                   // 5
da = (double[]) ia;        // 6
```

3. Om man har en triangel med *minst* en 60° vinkel gäller sambandet $c^2 = a^2 + b^2 - ab$ mellan sidlängderna a, b och c. Skriv ett program som givet ett heltal n beräknar hur många sådana (unika) trianglar det finns vars sidor är heltal och ligger i intervallet [1, n]. Metoden sqrt (roten) är tillåten att använda. Exempel: 6p

```
n   Antal trianglar (exempel)
1    1              (1,1,1)
3    3              (1,1,1) (2,2,2) (3,3,3)
10   12
25   35    ... (3,8,7) (4,4,4) (5,5,5) (5,8,7) (5,21,19) ...
70   112
```

4. Givet två rektangulära matriser med icke-negativa tal och ett positivt heltal $n \geq 0$. Skriv en metod som returnerar antalet par av element ur respektive matris som ger summan n. Metoden skall använda arrayer för matriser. För full poäng krävs en lämplig funktionell nedbrytning. Skriv en kort kommentar vid varje metod, vad är syftet med metoden? Exempel: 12p

```
Matris 1   Matris 2    n   resultat
[0, 2, 3   [0, 0, 1    4   4 (2+2,3+1,1+3,1+3)
 0, 0, 1]   3, 2, 3]

[0, 2, 0   [0, 1, 0    3   1 (2+1)
 1, 0, 0]   0, 0, 0]
```

   TIPS: Man kan byta mellan matrix och array!

5. Ett teleskopord består av en sammansättning av början från ett ord (prefix) med slutet av ett annat ord (suffix) t.ex. "lejon" och "tiger" blir "liger". För att skapa teleskopord kan vi använda följande regler. 8p

   a) Varje teleskopord skall bestå av ett icke-tomt prefix från det första ordet sammanslaget med ett icke-tomt suffix från det andra ordet.

   b) Om prefixet slutar med en vokal måste suffixet börja med en konsonant och tvärtom.

   c) Teleskopordet får inte innehålla något av (de hela) ursprungsorden.

   Skriv en metod som givet två ord (strängar) skapar en lista med teleskopord enligt ovan. Se appendix för tillåtna metoder. Exempel (dubbletter tillåtet):

```
Ord1  Ord2     Lista med teleskopord
lion  tiger    [ler, liger, lir, liger, lior, lioger]
spoon fork     [sork, spork, spok, spork, spook, spoork]
two   words    [tords]
```

6. Rita en bild som visar variabler, värden, referenser och objekt samt hur dessa förhåller sig till varann före, respektive efter anropet av metoden doIt. Rita som vi ritat under kursen, lådor, pilar o.s.v. Ni *måste* rita! 8p

```
MyClass[] a1 = {new MyClass(1), new MyClass(2), new MyClass(3)};
MyClass[] a2 = {new MyClass(3), new MyClass(2), new MyClass(5)};
                // Before
doIt(a1, a2);   // Call
                // After

void doIt(MyClass[] a1, MyClass[] a2) {
    for (int i = 0; i < a1.length; i++) {
        if (a1[i].n == a2[i].n) {
          a1[i] = a2[i];
        } else {
          a1[i].n = a2[i].n;
        }
    }
}

class MyClass {
    int n;
    MyClass(int n) {
        this.n = n;
    }
}
```

7. Mexico är ett tärningsspel om pengar! En omgång går till så att alla spelare kastar 2 sexsidiga tärningar. Den med lägst poängvärde förlorar omgången och måste lägga pengar i potten. Detta upprepas tills endast en av spelarna har pengar kvar. Denne är då vinnaren och får allt i potten. Vem som förlorar omgången beräknas i två steg: 12p

   - Tärningsvärdet beräknas. Tärningen med högst värde ger tiotalssiffran den andra entalssiffran. Exempel: 3 och 4 ger 43, 5 och 5 ger 55, 4 och 2 ger 42.
   - Därefter jämförs värdet enligt följande:
     - Värde 21. Detta värde är maxvärdet (Mexico) mer värt än alla andra värden.
     - Värdena 66, 55, ..., 11. Dessa värden är mindre värda än Mexico men mer värda än övriga poäng. T.ex. är 44 värt mer än 45. Inbördes gäller vanlig numerisk jämförelse.
     - Övriga värden (t.ex. 23, 61,...). Dessa värden är mindre värda än de ovan. Inbördes gäller vanlig numerisk jämförelse.

   Vi skall implementera en OO-modell för Mexico. För listor skall ni använda List/ArrayList (inte arrayer), se appendix.

   a) Skriv en klass MexicoDice för tärningarna. Klassen innehåller endast en metod roll(). Metoden returnerar ett tärningsvärde enligt ovan.

   b) Skriv en klass Player. En spelare har ett namn och en poäng (tärningsvärde) för aktuell omgång. Namnet sätts då spelaren skapas. Klassen skall ha metoder för att sätta och avläsa poängen.

   c) Skriv en klass Mexico för hela spelet. Spelet har en lista med spelare och en tärning vilka både sätts då klassen instansieras. Skriv en metod getPlayerLoosingRound(), Metoden låter alla spelare kasta och returnerar spelaren med lägst poäng (den som förlorade rundan).

   Klasserna skall vara så icke-muterbara som möjligt och dölja så mycket som möjligt av sin data (information hiding).

8. Betrakta koden nedan och ange för varje rad a)-h) *en* av följande 8p

   - Kompilerar ej
   - Körningsfel
   - Om inget av ovan, ange vad som skrivs ut.

   Du måste ge en kort motivering för varje svar (OBS! Flera satser på varje rad p.g.a. utrymme)!

```
a) B b = new B(); b.doA();
b) IA a = new X(); a.doA();
c) C c = new B(); c.doC();
d) B b1 = new C(); b1.doX();
e) IX x = new C(); X x1 = (X) x; x1.doA();
f) IX x2 = new C(); B b2 = (B) x2; b2.doC();
g) A a1 = new B(); a1.doA();
h) IA a2 = new A(); a2.doA();

// --- Interfaces and classes
public interface IA { public void doA(); }
public interface IX { public void doX(); }

public abstract class A implements IA {
    public void doA() { out.println("A.doA()");}
    public abstract void doC();
}
public class B extends A {
    public void doC() { out.println("B.doC()");}
}
public class C extends B implements IX {
    public void doA() { out.println("C.doA()"); }
    public void doX() { out.println("C.doX()"); }
    public void doC() { out.println("C.doC()"); }
}
public class X implements IX {
    public void doX() { out.println("X.doX()"); }
    public void doA() { out.println("X.doA()"); }
}
```

APPENDIX

```
// Tillåtna metoder för uppg. 5
String
- equals(s), avgör om en sträng innehåller samma tecken som en annan.
- charAt(int i), ger tecknet vid index i.
- indexOf(char ch), ger index för tecknet ch, -1 om tecknet saknas.
- length() ger längden av strängen.
- subString(int start, int end), ger en delsträng från
  start (inkl.) till end-1.
- subString(int start), ger en delsträng från start (inkl.)
  till strängens slut.
- toCharArray(), gör om strängen till en array med tecken.
- contains(s), avgör om s finns i strängen.
- endsWith(s), sant om strängen avslutas med s.
StringBuilder
- append(String s), lägger till strängen s sist i
  Stringbuilder-objektet.
- append( char ch ), som ovan
- setLength(), sätter aktuell längd, setLength(0)
  raderar alla tecken.
- toString(), omvandlar StringBuilder-objektet till en String.

// Tillåtna metoder för uppg. 7
List/ArrayList
- get(i), ger objektet för index i
- add(o), lägger till objektet o sist i listan
- set(i, o), lägger till objektet vid index i, flyttar övriga till höger.
- remove(o), tar bort objektet o ur listan, returnerar
  true om detta lyckades annars false
- remove(i), tar bort och returnerar objektet vid index i ur listan
- removeAll( list ), tar bort alla element i list.
- contains(o), sant om objektet o finns i listan.
- indexOf(o), ger index för objektet
- size(), ger längden på listan
Klassen Random med metoden nextInt() är tillåten.
```