## Uppgift 1 (5 poäng)

Redovisa hur det understrukna uttrycket evalueras och vilket värde det har :

Ingående variablers värden ges omedelbart före uttrycket

**a)**

```
int x = 1, y = 2;
int z = 5-4*y+x/2;
```

Multiplikation och division har samma prioritet och högre än addition / subtraktion. Vid lika prioritet utförs operationerna från vänster till höger.

```
5-4*y+x/2 = 5 -(4*y) + (x/2) = 5 - 8 + (1/2) = 5 - 8 + 0 [ty
heltalsdivision 1/2 = 0] = -3
```

**b)**

```
int x = 1, y = 2;
boolean ok = x >= 0 && y > 2;
```

Relationsoperatorerna har samma prioritet och högre än de logiska operatorerna.

```
x >= 0 && y > 2 = (x >= 0) && (y > 2) =(false) && (false) =
false
```

**c)**

```
double w = 1, y = 3.2, z = 12.2;
double res = z/y/w;
```

Från vänster till höger :

```
z/y/w = (12.2/3.2)/1 = 3.8124999999999996
```

**d)** Evaluera nedanstående boolska uttryck för varje kombination av värden (T=true och F=false) på a, b och c

```
a || b && c
```

Här måste man veta att and (&&) har högre prioritet än or (||) och det är lämpligt att man sätter upp ett par kolumner och räknar ut deluttrycken separat.

| a | b | c | b && c | a \|\| (b && c) |
|---|---|---|---|---|
| **T** | **T** | **T** | **T** | **T** |
| **T** | **T** | **F** | **F** | **T** |
| **T** | **F** | **T** | **F** | **T** |
| **T** | **F** | **F** | **F** | **T** |
| **F** | **T** | **T** | **T** | **T** |
| **F** | **T** | **F** | **F** | **F** |
| **F** | **F** | **T** | **F** | **F** |
| **F** | **F** | **F** | **F** | **F** |

## Uppgift 2 (2 poäng)

Förklara kort begreppet undantag (*exception*) resp. signatur i Java.

Undantag :

Undantag är en mekanism för att ta hand om vissa felsituationer som kan uppstå under körningen av ett Java-program. När ett fel uppstår skapas ett speciellt undantagsobjekt som 'kastas' till ett 'catch-block', dvs exekveringen avbryts där undantaget genereras och återupptas i catch-blocket.

Signatur :

En metods namn plus antal, typ och ordning för dess parametrar. Används för att systemet skall kunna skilja på överlagrade metoder, exempelvis println(int) och println(String). Observera att en metods returtyp inte ingår i signaturen.

## Uppgift 3 ( 2 poäng)

Beskriv effekterna som modifieraren har för:

**a)** en **metod** som är deklarerad *public*
En *public* metod kan nås av vem som helst utanför och innanför klassen, metoden ärvs av härledda klasser.

**b)** en **metod** som är deklarerad *private*
En *private* metod är bara tillgänglig inom den egna klassen, den ärvs inte heller av härledda klasser.

**c)** ett **attribut** som är deklarerat *static*
En *static* variabel / attribut är knuten till klassen och därför gemensam för alla objekt i klassen. Alla objekt manipulerar ett och samma värde.

**d)** en **metod** som är deklarerad *static*
En *static* metod är också knuten till klassen och kan anropas via klassnamnet utan att något objekt skapats explicit.

## Uppgift 4 (6 poäng)

Spåra följande loopar så att det klart framgår hur de ingående variablerna ändrar värde

```
1.   int j,sum=0;

     for (int i = 2; i < 24; i = i+2)
     {
      j = 24-i;
      while (j > i)
      {
          sum = sum + i;
          j = j - 2*i;
      }

     }// for
```

***Så här löper värdena på variablerna***

| i | j | sum |
|---|---|---|
| 2 | | |
| | 22 | 2 |
| | 18 | 4 |
| | 14 | 6 |
| | 10 | 8 |
| | 6 | 10 |
| | 4 | villkoret j>i falskt |
| 4 | | |
| | 20 | 14 |
| | 12 | 18 |
| | 8 | villkoret j>i falskt |
| 6 | | |
| | 18 | 24 |
| | 6 | villkoret j>i falskt |
| 8 | | |
| | 16 | 32 |
| | 0 | villkoret j>i falskt |
| 10 | | |
| | 14 | 42 |
| | -4 | villkoret j>i falskt |
| 12 | 12 | villkoret j>i falskt |
| 14 | 10 | villkoret j>i falskt |
| 16 | 8 | villkoret j>i falskt |
| 18 | 6 | villkoret j>i falskt |
| 20 | 4 | villkoret j>i falskt |
| 22 | 2 | villkoret j>i falskt |

**Anrop av metoden** : `loopa(8, 128, 10);`

```
public void loopa(int start, int stop, int breakVal)
{
   int mult, count = 0;

   for (mult = start; mult <= stop; mult = mult+start)
   {
      System.out.print (mult + "  ");
      count++;
      if (count % breakVal == 0)
         System.out.println();
   }// for

}// loopa
```

Anmärkning : `System.out.print()` skriver ut en sträng utan att göra ett efterföljande radbyte, medan `System.out.println()` genererar ett radbyte efter utskriften.

Här är utskriften, med count inom parentes

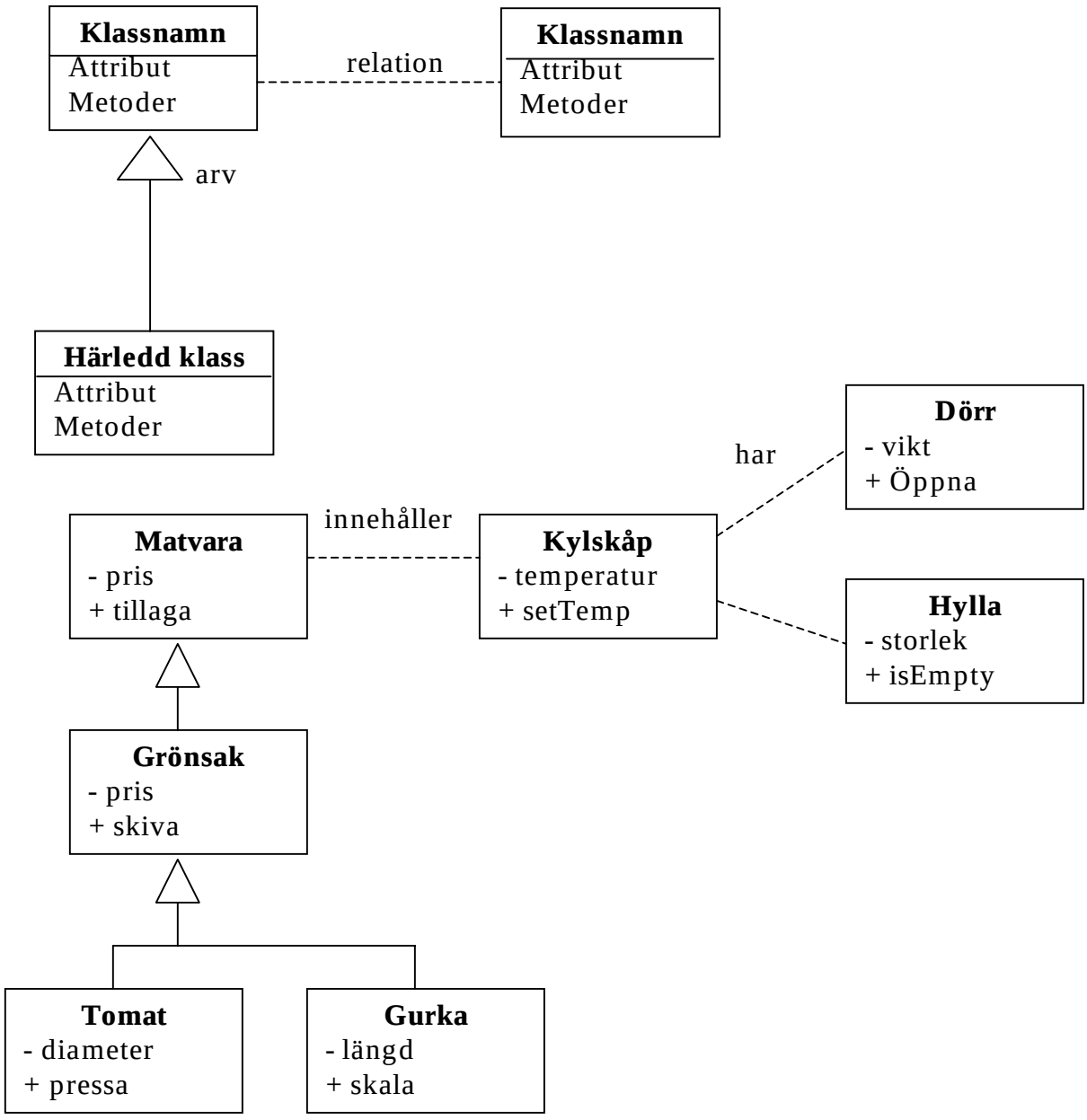
8(1) 16(2) 24(3) 32(4) 40(5) 48(6) 56(7) 64(8) 72(9) 80(10)
88(11) 96(12) 104(13) 112(14) 120(15) 128(16)

## Uppgift 5 (6 poäng)

**Problembeskrivning:**

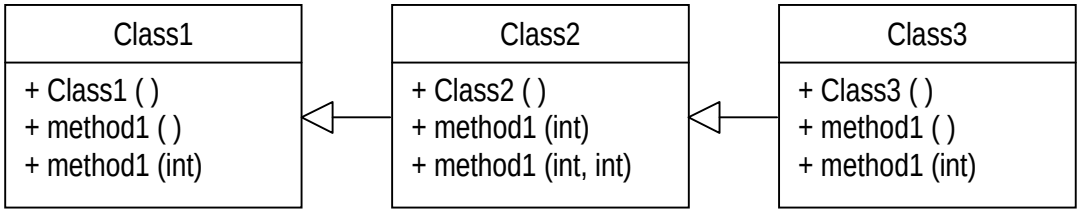
På ditt nya jobb som kylskåpsdesigner/testare så får du i uppdrag att utarbeta ett system som simulerar ett kylskåp och olika typer av innehåll och hur de klarar sig beroende på temperaturen. Intressant är att se om gurkor tål högre temperatur än andra grönsaker, speciellt tomater. Fisk och korv har kunderna också uttryckt intresse av, och de vill också ha större hyllor och en lättare dörr, något som måste tas med i beräkningen över hur mycket ström kylskåpet använder för att hålla en viss temperatur.

Gör en analys, dvs ta fram tänkbara klasser och beroenden och rita ett klassdiagram över resultatet. I diagrammet skall ingå minst fem klasser, ett arv och en relation, t.ex ’använder’, ’har’, ’består av’. Det finns inga krav på vilka klasser du beskriver annat än att de skall ha med problembeskrivningen att göra. Varje klass skall ha minst ett attribut och en metod. Använd UML-notation enligt nedan:



## Uppgift 6 (4 poäng)

Utgå från följande situation




och följande deklarationer:

```
Class1 c1 = new Class1();
Class2 c2 = new Class2();
Class3 c3 = new Class3();
```

Vilken *version* av `method1` anropas efter följande satser (skriv e/t om kombinationen **e**j är **t**illåten). Motivera dina svar!

| | **Klass** | **Motivering** |
|---|---|---|
| `c1.method1(1);` | **Class1** | **Finns en sådan metod definierad i klassen** |
| `c1.method1(1, 2);` | **e/t** | **Class1 saknar en sådan metod** |
| `c1 = c2;` | | **Nu referar c1 till ett objekt av klass Class2** |
| `c1.method1();` | **Class1** | **Objektet har ingen omdefinierad version av denna metod så den ärvda används** |
| `c1.method1(1);` | **Class2** | **Objektet (av Class2) har en omdefinierad version av denna metod** |
| `c1.method1(2, 1);` | **e/t** | **Class1 saknar en sådan metod, så trots att objektet har en sådan så ser inte referensvariablen c1 det. Kompilatorn kan bara utgå ifrån definitionen av Class1 när den avgör vad som är tillåtet, den kan inte känna till att c1 refererar till ett objekt av typen Class2 här.** |
| `c1 = c3;` | | **Nu referar c1 till ett objekt av klass Class3** |
| `c1.method1();` | **Class3** | **Objektet (av Class3) har en omdefinierad version av denna metod** |
| `c1.method1(1);` | **Class3** | **Objektet (av Class3) har en omdefinierad version av denna metod** |
| `c1.method1(2, 3);` | **e/t** | **Class1 saknar en sådan metod.** |

## Uppgift 7 (3 poäng)

Ge exempel och förklara skillnaden mellan formella och aktuella parametrar
*Formella parametrar är specifikationen av värden (antal, typ & ordning) som en metod kräver för att kunna anropas.*
*Aktuella parametrar är de värden (variabler, explicita värden, evaluerade uttryck) som ges vid anropstillfället.*
*De formella parametrarna fungerar som lokala namn för de värden som överförs.*

## Uppgift 8 ( 4 poäng)

Anta att koden nedan är given för klasserna `Person` och `Anstalld`. Fyll i koden som saknas för att implementera konstruktorerna i `Anstalld` klassen. Du får själv bestämma vad som skall vara standardvärden.
**Tips**: Använd `this` och `super`.

```
public class Person
{
    private String name;
    private int yearOfBirth;

    /**
     * Instantiera ett Person objekt med namn n och födelseår y
     */
    Person (String n, int y)
    {
        name = n;
        yearOfBirth = y;
    }
}


public class Anstalld extends Person
{
    private String room;

    /**
     * Instantiera ett Anstalld objekt med namn n, födelseår y
     * och rumsnummer room
     */
    Anstalld (String n, int y, String room)
    {
        // här saknas kod
      super(n,y);  // använd basklassens konstruktor
      this.room = room; // attributet nås med this
    }
```

```
  /**
     * Instantiera ett Anstalld objekt med standardvärden för
     * namn,födelseår och rumsnummer.
     */
    Anstalld ()
    {
        // här saknas kod, använd dig av konstruktorn ovan
      this("<name>",9999,"<room>");
      // konstruktor i klassen med annan signatur än denna
    }

}
```