KOMMENTAR

Øyvind Teig, Autronica

Instansieringen av Java (2)

Dette leserinnlegget viser hvordan det er mulig å få null epler dersom man har et eple på et fat, legger to nye epler på fatet, og fjerner disse to igjen. På denne måten kan jeg rette opp manglende detaljer i mitt forrige leserinnlegg. Slike utregninger er mulig i det fleste programmeringsspråk - som C, C++, Pascal og Modula-2. Jeg skal vise Java-varianten. Jeg vet bare om programmerinsspråket occam som ikke tillater at x = x + (y-y) kan bli noe annet enn x.

Eksemplene nedenfor er påpekt av P.H.Welch ved universitetet i Kent. Han kaller disse tingene "semantisk singularitet". "Semantisk" fordi det omhandler hva som skjer, "singularitet" fordi vi har mistet kontrollen. Se på denne Java-klassen:

```
class  Lagre_x {                                          //  1
  int x;                                                  //  2
  public Lagre_x (int v) {                                //  3
    x = v;                                                //  4
  }                                                       //  5
  public static void Alias (Lagre_x x1, Lagre_x x2) {     //  6
    // x1.x = x1.x + (x2.x - x2.x);                       //  7
    x1.x = x1.x + x2.x;                                   //  8
    x1.x = x1.x - x2.x;                                   //  9
  }                                                       // 10
}                                                         // 11
```

Klassen ovenfor heter "Lagre_x", og konstruktoren ved samme navn - i linje 3-5 - blir automatisk kjørt når det lages to instanser av klassen - i linje 14-15. Verdien obj1.x er nå 1, og obj2.x er 2, fordi a er 1 og b er 2. Neste punkt er at vi legger merke til at metoden definert i linje 6-10 er erklært som "static". Dette betyr at det er en "klasse-metode", til forskjell fra den mer vanlige "instans-metoden". En klasse-metode er det nærmeste Java kommer en vanlig C-prosedyre. I linjene 17 og 19 ser vi hvordan en slik klasse-metode kjøres, via klasse-navnet "Lagre_x", ikke objekt-navnet "obj1" eller "obj2". Dette konseptet benyttes av Javas "System"-klasse, som kun definerer klasse-metoder. På denne måten defineres et sett med system-funksjoner som ikke har noe objekt-rammeverk. "System.out.println" er et slikt eksempel.

```
class  Raritet_alias {                                                                  //  12
  Raritet_alias (int a, int b) {                                                        //  13
    Lagre_x obj1 = new Lagre_x (a);                                                     //  14
    Lagre_x obj2 = new Lagre_x (b);                                                     //  15
    System.out.println ("Initielt: obj1 = " + obj1.x + ", obj2 = " + obj2.x);          //  16
    Lagre_x.Alias (obj1, obj2);                                                         //  17
    System.out.println ("Forskjellige objekt: obj1 = " + obj1.x + ", obj2 = " + obj2.x);  //  18
    Lagre_x.Alias (obj1, obj1);                                                         //  19
    System.out.println ("Samme objekt: obj1 = " + obj1.x + ", obj2 = " + obj2.x);     //  20
  }                                                                                     //  21
}                                                                                       //  22

public class  Raritet_main {                                                            //  23
  public static void  main (String argv []) {                                           //  24
    Raritet_alias  test = new Raritet_alias (1, 2);                                     //  25
  }
}
```

Temaet vårt var alias-feil. "Alias" betyr at en minneposisjon kan omtales ved mer enn et navn. Som regel er dette ikke ønsket, i vårt tilfelle fører det riktig galt av sted. Når Lagre_x kjøres fra linje 17, vil den interne x for hvert objekt leve for seg (OK), men når Lagre_x kalles i linje 19 peker x1.x og x2.x på det samme elementet - fordi Lagre_x har fått obj1 som begge parametere. Etter linje 8 er dermed x1.x blitt 3, men fordi x1.x og x2.x nå begge peker til samme minneposisjon og dermed har verdien 3, blir x1.x lik 3-3 = null etter linje 9. Feilen er et faktum, og læreren burde gi en rød "g" i margen. Av en eller annen grunn lar språkkonstruktørene slike ting passere, ikke spør meg hvorfor. Her ser dere utskriften av programmet ovenfor:

```
Initielt:              obj1 = 1, obj2 = 2
Forskjellige objekt:   obj1 = 1, obj2 = 2
Samme objekt::         obj1 = 0, obj2 = 2
```

Dette kan du teste ved å kjøpe en Java-bok med CD, installere Java, putte koden ovenfor i en fil som skal hete "Raritet_main.java" (husk, du må ha WIN95) - kompilere med "Javac Raritet_main.java" og kjøre vha. "Java Raritet_main". Lykke til!

■

Øyvind Teig. Autronica. 7005 Trondheim. Tel.: 73 58 12 68. Fax: 73 91 93 20.