Instansieringen av Java

Jeg er blant de som har kjøpt meg noen Java bøker. Kanskje har jeg dermed "kjøpt meg inn i eventyret" som John C. Dvorak (PC Magazine, mai) sier. Jeg har i så fall "hatt hjernen ute på service" - fordi jeg inderlig ønsker at Java ikke blir "dødfødt" (juni).

Jeg mener at noe av det viktigste med Java er 1) at pekere er borte - en stor gruppe feil er dermed fjernet, og 2) at språket har støtte for fleroppgavekjøring - behovet for en egen separat sanntidskjerne er derfor borte.

Med objektorienterte språk lager man objekter av maler som kalles klasser, med noen arvede og noen nye egenskaper. I sanntidssystemer lager man "tasker" eller oppgaver. Disse er ferdige "program" som andre program igjen kommuniserer med vha. postbokser eller kanaler eller noe liknende. Man kan si at disse to løsningsmetodene ikke har vært lette å forene. Java gjør et forsøk på dette - akkurat som Ada-95 og SDL-92. Det nærmeste man ellers kommer i C-verdenen er C++ og separat sanntidskjerne. Verken C++ eller sanntidskjernen forstår hva en "task" er. Huff og huff.

Ingen har vurdert å ha to samtidige sjafører på en buss, fordi man kan ikke dele på et ratt. Javas "monitor" løser problemet med deling av felles ressurser. Engelskmannen C.A.R. Hoare oppfant "monitor" i 1974. I Java er det opp til brukeren å passe på at dette benyttes rett. Her tillates man friskt å modifisere en godt innelukket variabel med kall fra to samtidige brukere! Bare synd Sun ikke kjente til språket occam, som er basert på den samme Hoares "CSP" teori fra 1978. Ada baserer seg også på CSP, men språket regnes for å være tungt. Occam kompilatoren "tar deg" dersom du forsøker å skrive til en og samme variabel fra parallelle tasker - det vil si - språket har en dypere forståelse av hva en "task" er. Det kan utføre "usage check" av variable. Likevel, Java **har** begrepene "thread" og "synchronized", pluss operander på "monitor", så man kan bygge mye oppå disse primitivene. Men uansett hvilken paradigme man bygger på toppen av Java, så vil Java kompilatoren kun forstå Java, ikke den nye paradigmen. Derfor burde man lagt seg litt høyere enn "monitor". Likevel, det går an å se det positive med Java uten å ha hjernen på service!

En annen viktig ting med Java er alle bibliotekene som er definert. Modula-2 kode var ikke skikkelig portabel fordi bibliotekene ikke ble definert da språket var nytt. I Java har man nesten et helt vindus-basert nettverk multitasking operativsystem bygd inn fra første dag!

Ada er et språk hvor skalerbarhet er satt i høysetet. Det skal være mulig å utvikle kjempestore applikasjoner uten å miste oversikten. Jeg tror Java er på rett spor med sitt ene programkode format. Dette inneholder portabel kjørbar kode og type-informasjon.

Det er flott at Java kan benyttes over internett, og at det er portabelt. Det gir håp om at språket i seg selv er så bra som det er. Da kan det kanskje benyttes til mer enn internettprogrammering. Alle de flotte overbygningene over C++ burde så absolutt migrere til Java-verdenen. På den måten kunne grunnmuren stått litt mer i forhold til resten av huset.

Øyvind Teig, Autronica

Innlegget er noe forkortet - Red.