Systemvaruhuset AB Solna strandväg 78 171 54 Solna
Tel 08 50 52 10 42 Fax 08 50 52 10 10 Org. Nr 556665-7614 E-post info@systemvaruhuset.se www.systemvaruhuset.se

Introduktion till Java

1 Bakgrund och historik

Java kallades initialt för Oak och var en del av projektet "Green project" som startades 1990 på Sun av James Gosling, Patrick Naughton och Mike Sheridan. Språket var från början tänkt att tillämpas vid utveckling av små inbyggda system som exempelvis program för mikrovågsugnar, TV-apparater, stereoanläggningar etc.

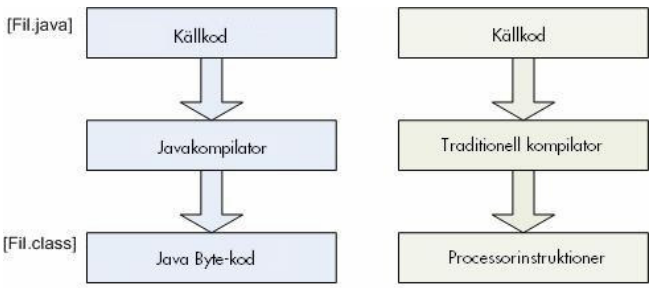
Problemet med traditionella programmeringsspråk som exempelvis C/C++ är att de måste kompileras för en specifik processor vilket fördyrar utvecklingen. Suns mål var att ta fram ett programspråk med vilket processoroberoende/plattformsoberoende program kunde utvecklas.

2 Vad kännetecknar Java?

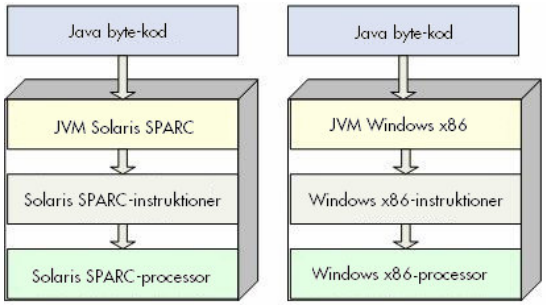
2.1 Plattformsoberoende

Ett program skrivet i Java kallas för plattformsoberoende. En mer precis beskrivning är att det är oberoende av processorns instruktionsuppsättning. Javakällkod kompileras till så kallad Java Byte-kod som i sin tur översätts av en Java Virtual Machine (JVM) till processorinstruktioner. Jämför detta med ett traditionellt programspråk där processorinstruktioner genereras vid kompileringstillfället (Figur 1). Ett javaprogram kan alltså flyttas mellan plattformar förutsatt att det finns en JVM för aktuell plattform (Figur 2).

Figur 1



Systemvaruhuset AB Solna strandväg 78 171 54 Solna
Tel 08 50 52 10 42 Fax 08 50 52 10 10 Org. Nr 556665-7614 E-post info@systemvaruhuset.se www.systemvaruhuset.se



Figur 2

2.2 .class-filer och länkning

Java byte-koden läggs i en eller .class-filer och om en .class-fil innehåller en main-metod (det fungerar på samma sätt som i C/C++) så kan Javaprogrammet startas med kommandot:

java –cp . fil

-cp-paramterern anger classpath, vilket innebär sökväg till .class-filer, i detta fallet aktuell katalog (.). *java*-kommandot startar JVM:en.

I Java länkas programmet när det exekveras till skillnad från exempelvis C/C++ där det länkas ihop när programmet byggs. I C/C++ kontrolleras alltså exempelvis att funktionen *foo()* som anropas finns med i någon objektkodsfil när exe-filen byggs. I java kontrolleras att *foo()* verkligen finns i någon .class-fil först när programmet exekveras.

2.3 Språkets egenskaper

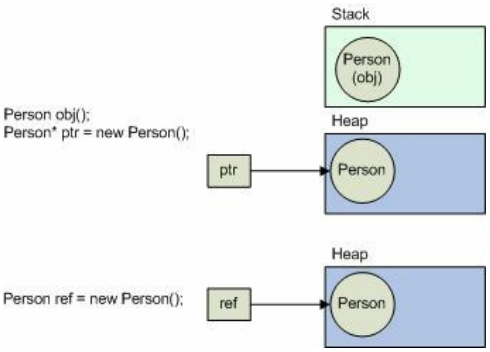
- Java-syntaxen är väldigt lik den för C++. Java är förenklat i jämförelse med C++ och språkkomponenter/egenskaper som ofta orsakade fel så som pekare, strukturer, headerfiler, operatoröverlagring, multipla arv ingår ej i Java.

- Java är objektorienterat och har stöd för arv, runtime-polymorfism och dynamisk bindning. Java har även stöd för kodgenerik (compile-time-polymorfism) sedan version 5. Kodgeneriken motsvarar C++ templates.

Systemvaruhuset AB Solna strandväg 78 171 54 Solna
Tel 08 50 52 10 42 Fax 08 50 52 10 10 Org. Nr 556665-7614 E-post info@systemvaruhuset.se www.systemvaruhuset.se

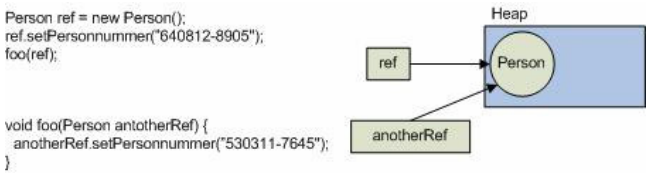
- Java innehåller en s.k. Garbage Collector som städar upp ej använt minne. I C++ är programmeraren ansvarig för att allokera minne när ett objekt skapas och avallokera minnet när det ej används mer av programmet. Detta är en stor felkälla i C++-program.

- I C++ kan programmeraren styra om ett objekt skall läggas på stacken eller på heapen. I Java skapas alltid en referens till ett objekt på heapen, ett objekt kan ej skapas på stacken.



Figur 3: *Person obj();* och *Person* ptr = new Person();* visar hur ett objekt kan skapas på stacken respektive på heapen i C++. I Java skapas alltid en referens till ett objekt på heapen (*Person ref = new Person();*)

- I Java gäller call-by-reference för objekt vid metodanrop. Detta innebär att metoden som anropas får en egen referens till objektet. Metoden kan ändra på objektets tillstånd och det nya tillståndet kommer således att gälla för alla andra referenser till objektet i programmet. Javaspråket saknar en konstruktion för att förhindra metoder från att förändra objekt som refereras via parametrar (om nyckelordet *final* används innebär detta endast att referensen inom metoden inte kan tilldelas ett nytt objekt). Detta är en betydande nackdel med språket.

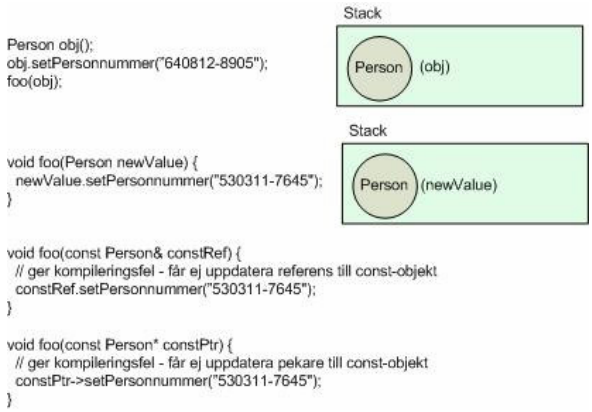


Figur 4

Systemvaruhuset AB Solna strandväg 78 171 54 Solna
Tel 08 50 52 10 42 Fax 08 50 52 10 10 Org. Nr 556665-7614 E-post info@systemvaruhuset.se www.systemvaruhuset.se

C++ ger möjligheter att styra detta (Figur 5). För det första genom att helt enkelt anropa metoden med att kopiera över värdet (call-by-value) istället för att skicka en referens eller pekare som parameter.

Vidare erbjuder C++ möjlighet att anropa metoden med referenser eller pekare (vilket blir effektivare) och deklarera objekten som refereras eller pekas till som ”const” (endast för läsning).



Figur 5: *void foo(Person newValue)*, Person-objektsparametern till foo får ett värde genom att värdet kopieras från objektet som används vid anropet till foo (*foo(obj);*). Om foo-metoden uppdaterar parameterobjektets tillstånd kommer detta alltså inte att slå igenom för objektet *obj* som användes vid anropet.

void foo(const Person& constRef), parametern är en referens till en const-objekt, för vilket tillståndet inte kan uppdateras i foo-metoden. Detsamma gäller om parametern är en pekare till ett const-objekt (*void foo(const Person * constPtr)*). (Om foo-metoden med pekarparametern används så skall foo anropas enligt *foo(&obj)* och inte foo(obj)).

Observera att för primitiva datatyper som int, char, boolean gäller call-by-value i Java.

3 Läs mer

- Java-ordlista: http://mindprod.com/jgloss/jgloss.html