Oscar Antbring

# PROGRAMMERING 1


DOCENDO

Tryckt av Elanders
Utgiven 2024

ISBN: 978-91-7531-176-0
Artikelnummer: 3118

Författare: Oscar Antbring

Omslag: Docendo AB
Bild på omslaget: © Adobe Stock

# Innehållsförteckning

# 2 Programutveckling i Java



I detta kapitel ska du få läsa om **Java** som är ett av världens mest kända och populära programmeringsspråk. Det är oerhört mångsidigt programmeringsspråk och används för att exempelvis bygga mobila applikationer, konstruera chatbottar eller för att styra smarta prylar i våra hem.

Du kommer vidare att få läsa om hur du installerar rätt mjukvara som krävs för att du ska kunna börja skriva dina första koder. Och i samband med detta skriver vi om ramverket JDK och som består av massa färdiga så kallade klasser.

Men vi inleder med att skriva om Java och vad som kan sägas vara för- och nackdelar med detta programmeringsspråk

## Vad utmärker Java?

Om du skulle läsa på nätet eller i böcker om vilka för- och nackdelar som Java medför som utvecklingsspråk så finns många aspekter i detta och som ofta beror på vad man du ämnar utveckla för typ av program.

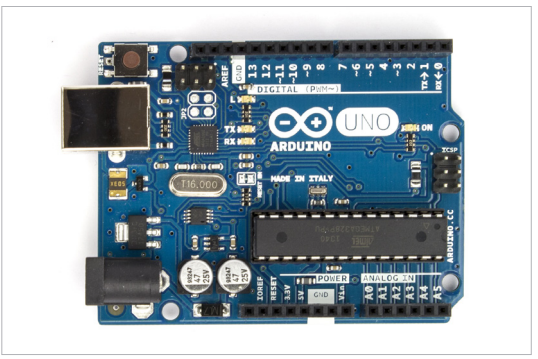
Här följer några generella och ofta omnämnda styrkor och svagheter med Java

### Plattformsoberoende

En av de mest utmärkande fördelarna med Java är att det är *plattformsoberoende*.

För att förstå vad det innebär så behöver vi gräva lite i begreppet *plattform* och vad det innebär. En plattform handlar i princip om vilken datormiljö du syftar till och då en specifik *hårdvara* och ett *operativsystem*.

När det handlar om hårdvara så grupperar man dessa efter vilken processor som finns i datorn och den vanligaste sådan är **x86** (som i tur tur inkluderar flera olika varianter av den) . Eller som du troligtvis känner igen plattformen som: **PC** (även om kopplingen mellan begreppet PC och x86 inte är helt okomlicerad)**.**



Andra exempel på hårdvaruplattformar är ARM (som används i exempelvis Chromebook, mobiltelefoner och Arduino) och Playstation.

I bilden kan du en Arduino som är en så kallad enkorts-dator och som bland annat består av en processor med ARM-arkitekturen.

Även en dator med samma hårdvara (som en PC) kan innefatta flera plattformar och det beror på att en dator kan köras med olika operativsystem.

Ett operativsystem är den viktigaste mjukvaran på en dator och som hanterar datorns hårdvara samt tillåter en miljö för andra program att köra i. Det betyder att program som webbläsare, ordbehandlingsprogram och datorspel behöver ett operativsystem för att kunna köra.

Exempel på operativsystem som finns till PC-kompatibla arkitekturer är OS/2 (”mac”), Linux och Microsoft Windows.

Alla dessa kombinationer av hårdvara och mjukvara bildar en möjlig plattform. Och att Java är plattformsoberoende innebär alltså att den kod som du skriver kan köras oberoende av vilken typ av dator (med operativsystem) som används.

Vi kommer att återkomma lite mer till detta när du kommer läsa om hur kod som du skriver i Java blir ett körbart program.

# Fördelar med Java

Följande fördelar omnämns ofta i samband med att man läser om Java som programmeringsspråk:

## Java är säkert

Det blir genast rätt tekniskt om vi ska gräva i *varför* Java betecknas som ett av de mest säkra programmeringsspråken. Men en viktig anledning till varför det är så är för att Java-program körs i en så kallad *virtuell maskin* som gör att programmet körs i en skyddad miljö.

Du kommer också kunna läsa mer om vad en virtuell maskin är lite längre in i detta kapitel.

## Java är objekt-orienterat

I det sista kapitlet i denna lärobok så kommer du lära dig om vad det innebär att jobba med enligt ett objektorientering tankesätt. Dock är det så att du arbetar enligt detta *paradigm* redan när du skriver dina första koder.

> Ett ***programmeringsparadigm*** är ett synsätt på hur program ska konstrueras och fungera. Ett objektorienterat paradigm används i språk som **Java**, **C#** och **Python**.

Men vi tror nog att du gör bäst i att lägga det begreppet lite åt sidan innan du faktiskt kommer fram till just det kapitlet. Likväl har du möjlighet att läsa mer om vad objektorienterad programmering innebär i bokens bilaga. Vi har valt det upplägget så att du själv kan avgöra om och när du vill ta dig an *teorin* som förmedlas där.

## Java har hög prestanda

Även om påståendet att Java har hög prestanda är lite mer komplicerat idag så får vi ändå lov att säga att Java som programmeringsspråk alltid legat i framkant vad gäller att optimera prestanda för program. Men då i jämförelse med programmeringsspråk som fungerar på liknande sätt (med en virtuell maskin).

Vi tänker särskilt på dess inbyggda hantering av flera *trådar* och att Java var väldigt tidigt ute med detta. En process är en program eller en tjänst på datorn som körs. En process kan i sin tur delas upp i just trådar och moderna programmeringsspråk kan köra flera sådana samtidigt.

> Moderna programmeringsspråk kan bryta ner en process i mindre komponenter (så kallade trådar) som kan hanteras bättre och enklare. Och att flera trådar kan köras samtidigt (eng: **multithreading**)

Om du använder Microsoft Windows så kan du öppna aktivitetshanteraren för att se alla processer som för tillfället körs. Du kan också exempelvis öppna kommandotolken (**cmd**) och skriva kommandot **tasklist**.

```
chrome.exe                   25792 Console                    1     73 168 K
FileCoAuth.exe               11188 Console                    1     18 796 K
EXCEL.EXE                    15740 Console                    1     97 448 K
cmd.exe                      37012 Console                    1      3 888 K
conhost.exe                  37104 Console                    1      8 140 K
svchost.exe                  37492 Services                   0      7 640 K
DropboxNativeMessagingHos    37692 Console                    1      6 744 K
ai.exe                       37060 Console                    1     17 652 K
Dropbox.exe                  38188 Console                    1     23 456 K
```

Vissa av dessa processer är sådana som du kan knyta an till specifika program som Excel och Google Chrome i exemplet. Andra processer är knutna till mer obskyra tjänster som puttrar i bakgrunden och som du som användare aldrig kommer tänka på.

## Java har ett enormt ”community”

”*Community*” är inte bara en väldigt rolig tv-serie utan också begreppet för en grupp som verkar för att tillsammans utveckla det som de brinner för. I detta fall programmeringsspråket Java.

Och de personer som är en del av denna gemenskap har en betydande roll i hur Java utvecklar sig som programmeringsspråk. Detta skapar ett genuint engagemang där pengar inte nödvändigtvis är  den högsta motivationsfaktorn.

# Nackdelar med Java

Följande nackdelar omnämns ofta i samband med att man läser om Java som programmeringsspråk:

## Java är långsamt

Java har en bra prestanda jämfört med programmeringsspråk som fungerar på liknande sätt, men är betydligt långsammare än program som är kompilerade för en specifik plattform. Det kan handla om program som är skrivna i C eller C++ exempelvis.

## Komplicerad kod

Java kan väl inte sägas vara utmärkande i att ha en komplicerad kod med många ”ord” men i jämförelse med ett språk som **Python** så är det definitivt så. Jämför du en kod mellan dessa programmeringsspråk så framför koden i det sistnämnda som enklare att utläsa och mindre komplex

## Utmaningar vid utveckling av GUI

Ett så kallat ”**GUI**” är kort för ”**graphical user interface**”. Alltså att vi skapar ett grafiskt gränssnitt för vårt program, vilket förvisso inte är något som du kommer göra i och med denna lärobok men förr eller senare så blir det aktuellt så klart.

Det finns ingen enhetlig stil knuten till Java och för komplexa skrivbordsprogram så är det säkerligen bättre att jobba i andra programmeringsspråk såsom **C#** om du ska skriva skrivbordsprogram för Windows.

# Att utveckla program i Java

I detta avsnitt så kommer du att kunna läsa om vad du behöver installera för program för att kunna börja utveckla program i Java. Och det ska sägas direkt, det är rätt komplext detta. Men vi ska göra vårt bästa i att berätta om det på ett greppbart sätt.

Om du känner att detta avsnitt tills vidare är ointressant så kan du fortsätta och läsa om hur du installerar och kommer igång med **IntelliJ**. Det du kan läsa om här är alltså ingen förutsättning att förstå för att du ska kunna börja skriva dina första koder.

Java består i alla fall av tre grundläggande paket av mjukvara som samverkar för att kunna utveckla program i Java och köra dessa.

# 3 Hello World



Stunden är kommen för att du ska börja skriva dina första körbara koder. Sedan kommer du bara att utveckla dina kunskaper genom kommande kapitel och dina program kommer bli alltmer omfattande och användbara. Och roliga att skriva koden för.

Detta kapitel kommer att bli ett förhållandevis kort sådant och utgör startskottet på din resa i kodandets värld. Du ska skriva ett program som skriver ut en klassisk text ”*Hello World*” som brukar utgöra det allra första programmet som en programmere skriver.

Ja, utskriften ”*Hello World*” har en liten särskild plats i många programmerares hjärtan. Den brukar alltså utgöra ett grundläggande program som inte tar emot någon text alls från tangentbordet, utan enbart skriver ut en text. Programmet bygger därmed på en kod som innefattar den allra mest grundläggande syntax i ett programmeringsspråk.

Dessutom så brukar programmeraren vilja testa utvecklings- och körmiljön så att koden kan kompileras korrekt och bli ett körbart program, och detta ska inte underskattas. För att du ska kunna köra mer komplexa program längre fram så är det ju rimligt att ditt ramverk kan hantera en enkel utskrift. Vi kan kalla detta för en hälsocheck av utvecklingsmiljön.

## Utskriften ”Hello World”

I kodexempel (1) så har vi skrivit en programsats som skriver ut texten ”*Hello World*” på skärmen.

```
1.  //Exempel 1: Utskrift av ”Hello world”
2.
3.  public class Main
4.  {
5.      public static void main(String[] args)      //Metoden main()
6.      {
7.          //Kodblock som tillhör metoden main() börjar : Du skriver all din kod här.
8.          System.out.println(”Hello world!”);          //En programsats för utskrift
9.          //Kodblock som tillhör metoden main() slutar : Du slutar skriva all din kod här
10.     }
11. }
```

Du får nu inledningsvis ha inställningen att den kod som *du* skriver ska hamna innanför de innersta måsvingarna. Den kod som tillhör en uppsättning måsvingar kallas för ett *kodblock*. Det finns en del kryptisk kod utanför detta kodblock och som du bör förbise så här i början men vill du sätta tänderna i *hela* grundkoden som du kan se i exemplet (1) så kan du läsa om detta i bilagan.

Den programsats (eng: **statement**) som skapar själva utskriften kan se se på kodrad 8.

> En *programsats* är en åtgärd i koden som avslutas med ett semikolon. Och som kan liknas vid en mening i vanligt skriftspråk.

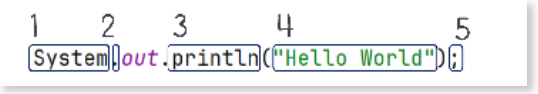
När vi syftar till en programsats så handlar det om en kod som avslutas med ett semikolon. Och koden gör *en* sak. Längre fram i boken så kommer du börja jobba med villkor och då kan man också prata om *uttryck* (engelska **expressions**).

Vi kommer så långt som möjligt att använda begreppet programsats i boken. Det finns många definitioner på en programsats och vi kan exempelvis kalla den för en enskild enhet för exekvering som inte utvärderar någonting. Vi har nämnt att en annan typ av kod i stället *kan* utvärdera något och då kallas det för uttryck.

Om du befinner dig i ditt hus och du bestämmer dig för att gå ut oavsett väder så är det en sats. Men om du i stället funderar över att gå ut, men bara om temperaturen är över 20 grader, så är det i stället ett uttryck. Innan du går ut kommer du först utvärdera om vädret är tillräckligt bra.

## Programsatsen under lupp

Även om konceptet med ett objektorienterat paradigm är något som du direkt uttryckligen kommer att arbeta med förrän i slutet av läroboken. Däremot kan du ha den insikten att kod i ett språk som Java konstrueras utifrån *klasser* (och därmed objekt) samt metoder. Dessa begrepp förklarade vi i föregående kapitel.




Och programsatsen i fråga beskriver just detta sätt att bygga koden. **System** (1) är en vanlig klass som ingår i Javas bibliotek och som främst innehåller metoder för att hantera in- och utmatningar.

För att använda resurser som ingår i en klass så skriver du klassens namn följt av en punkt (**2**). Detta tecken kallas för en medlemsoperator (eng: **member operator**) och då syftar namnet alltså till att vi berättar för koden att vi nu vill komma åt de medlemmar som tillhör klassen och som föregår punkten.

# 4 Variabler



Med all respekt för programmet som kan skriva ut ”*Hello World*” men det händer inte så där väldigt mycket. För att vi ska kunna ta nästa steg och involvera användaren i programmet så behöver vi lära oss att använda variabler.

En fundamental komponent i alla programmeringsspråk är *variabler* som är ett sätt för oss att hantera värden. Dessa värden kan vara av olika typer som text, heltal och decimaltal. Vi kan ge en variabel ett värde direkt i koden, men ännu mer spännande blir det när vi ber användaren att mata in värdet. Då öppnas äntligen dörrarna för att vi ska kunna skapa *interaktiva* program. Alltså att användaren får en aktiv roll i programmet, att det sker ett samspel mellan användaren och datorn.

I detta kapitel så kommer du att få lära dig att hantera variabler och olika typer av värden. Detta kommer du behöva tampas med ett tag. Sedan kommer du också att lära dig hantera inmatningar från användaren, som då alltså är en förutsättning för att skapa en interaktion med programmet.

## Att skapa en variabel

En *variabel* är oavsett en plats i ett program där vi kan lagra ett värde, och för att åstadkomma detta så behöver du tänka på tre moment.

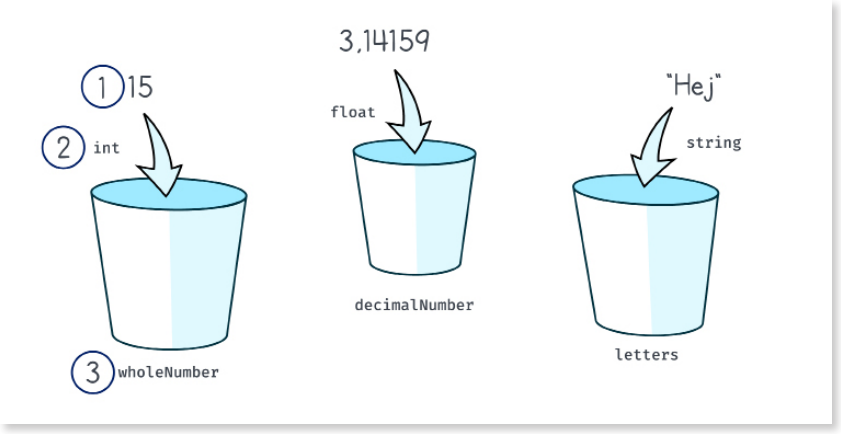
Nedanstående beskrivning utgår ifrån scenariot att du skapar tre olika variabler i koden och som alla kan lagra ett specifikt typ av värde.

```
1.  //Exempel 1: Tre variabler med olika datatyper
2.
3.  int wholeNumber = 15;
4.  double decimalNumber = 3.14159;
5.  String letters = ”hej”;
```

I kodexemplet (1) kan du se en kod som skapar de tre variabler vilket involverar tre moment:

## Namnge variabeln

Du behöver först bestämma namnet på variabeln (1) och detta kommer du att märka är inte så lätt som det låter. I alla fall när det kommer till att välja *bra* namn för variabler. Men försök att välja ett namn som återspeglar vad variabeln lagrar för värde.




Vi kommer att återkomma till namngivning av variabler titt som tätt genom boken. I bilden så har vi exempelvis gett en variabel namnet ”*wholeNumber*” och för detta exempel får vi lov att säga att det är ett tydligt och lämpligt namn. Men i kommande program så är det troligtvis inget vidare bra namn.

## Fastställa en datatyp

Utöver att ge en variabel ett namn så behöver du också fastställa vilken typ av värde som den ska kunna lagra (2). Inledningsvis så kommer du att jobba med tre sådana:

- Datatypen **int** som kan lagra heltal.
- Datatypen **String** som kan lagra tecken, bokstäver och siffror.
- Datatypen **double** som kan lagra decimaltal.

Vi kan tänka oss att dessa hinkar ser olika ut beroende på typen den hanterar. Exempelvis kan olika datatyper hantera olika stora värden, och då kan hinkarna vara mindre eller större.

Om du skulle försöka stoppa in bokstäver i en hink som är gjord för att lagra heltal, så kommer heltalshinken protestera så våldsamt att programmet kraschar (om användaren har matat in en uppsättning tecken när programmet förväntar sig ett tal).

Denna problematik kommer vi återkomma till, både i detta kapitel men framför allt i kapitlet om undantagshantering längre fram i boken.

# PROGRAMMERING 1 MED JAVA

Boken är skriven för dig som vill lära dig programmering med Java från grunden. Programmering är utvecklande på flera sätt, du kommer att utveckla din problemlösningsförmåga och din logiska tankeförmåga. Likaså kommer du att få träna upp ditt tålamod och din förmåga att uppmärksamma detaljer och vår utgångspunkt när vi skrev boken är att programmering är lika mycket ett kreativt område som ett tekniskt.

I denna lärobok så får du lära dig det som krävs för att du ska få en stabil grund att stå på inför ytterligare studier och med mer avancerad programmering. Du kommer att få lära dig om variabler och vektorer som är två olika sätt att lagra värden på. Vidare kommer du att få lära dig att använda kontrollstrukturer som används för att styra program i olika riktningar. Som vi ofta vill ska styras av användaren till programmering. Vidare så kommer du att få lära dig att använda funktioner som används för att gruppera koder som fyller ett särskilt syfte och att vi sedan kan använda den uppsättning kod när vi behöver den. På samma sätt som att vi har olika verktyg i en verktygslåda och som alla har en specifik uppgift, en funktionalitet.

Du kommer få läsa om vad som anses vara en bra kod. Hur du kommenterar och dokumenterar kod, hur du strukturerar koden och hur du ska tänka för att eftersträva enkel kod i förmån för onödigt svår kod. Likaså kommer du att få lära dig om vad som utmärker Java som programmeringsspråk och vad som krävs för att du ska kunna skriva dina allra första program. Boken är grundad i ett paradigm eller tankesätt som kallas för  objektorienterad programmering. Det innebär att du lär dig att skriva klasser som representerar saker eller så kallade entiteter i vår omvärld, och att du sedan kan skapa förekomster av dessa som kallas för objekt.

Boken är skriven utifrån gymnasieskolans läroplan och de riktlinjer som ligger till grund för kursen Programmering 1.

ISBN 978-91-7531-176-0
9 789175 311760

DOCENDO