

NetBeans 6.5

Avsikt

Att bekanta dig med NetBeans programmeringsmiljö, dvs att med hjälp av NetBeans

1. skapa ett nytt projekt
2. skriva in källkod (sparas som .java-fil)
3. kompilera (översätta) koden till byte-kod (sparas som .class-fil)
4. köra programmet.

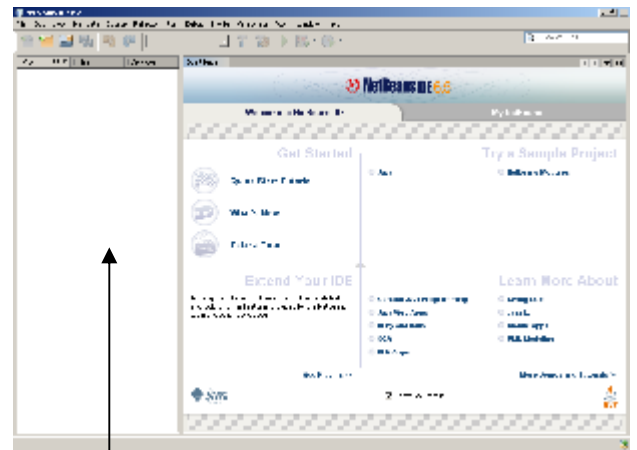
Men innan du börjar med detta måste du installera version 6.5 av NetBeans. Klicka på länken på kurssidan eller gå direkt till <http://java.sun.com/javase/downloads/netbeans.html>, hämta hem NetBeans och installerar programmet. Samtidigt installeras JDK 6.

När du har gjort detta ska du skriva ett mycket enkelt program och i samband med detta gå igenom ovanstående punkter.

Starta NetBeans

- Klicka på **Start** nere i vänstra hörnet.
- Välj **Program - NetBeans - NetBeans IDE 6.5** och klicka med vänster musknapp.

Nu ska ett fönster liknande figuren till höger synas. Eventuellt syns inte fönstret till höger (välkomstsida). Om välkomstsidan syns så kan du stänga den genom att klicka på kryssset.



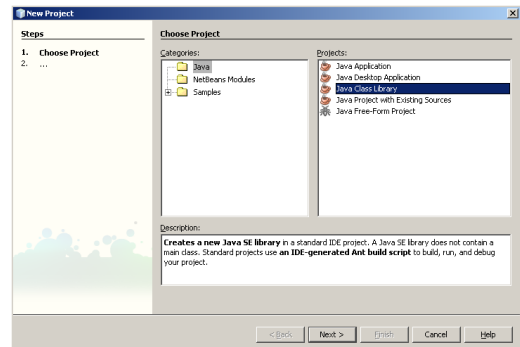
Projektfönster

1. Skapa ett nytt projekt

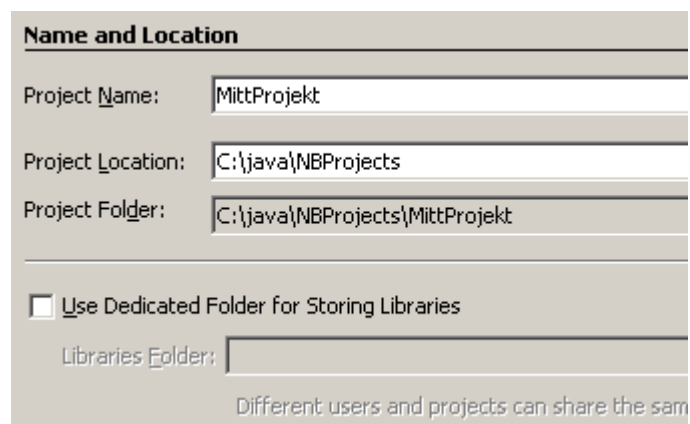
I NetBeans måste man använda sig av projekt. Projektet hjälper till att hålla ordning på filer, paket och mycket annat som kan ingå i programmeringsprojektet.

- Välj **File – New Project...**

Nu visas ett dialog-fönster - **New Project** - med några inställningar. Välj **Java** (under Categories) och **Java Class Library** (under Project). Klicka sedan på **Next >**.



Nu visas ett nytt dialog-fönster – **New Java Class Library** – med några inställningar. Här ska du ange projektets namn och var projektet ska lagras på hårddisken. Kalla projektet för **MittProjekt** och spara projektet i katalogen **C:\java\NBProjects** (går bra att skriva in / ange genom att klicka på Browse). Klicka slutligen på **Finish**.



Om du går till katalogen **NBProjects** så ser du att projektkatalogen **MittProjekt** har skapats och att den bl.a. innehåller katalogen **src**. Det är i katalogen **src** som du ska placera källkods-filer.

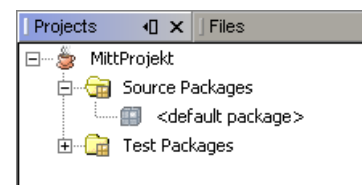
2. Skriva in källkod

Man skriver in källkoden med hjälp av ett speciellt program, en *editor*. En editor är som ett enkelt ordbehandlingsprogram. Men för att underlätta för programmeraren så innehåller ofta en editor speciella hjälpfunktioner.

Skapa källkodsfil i ett projekt

NetBeans innehåller bl.a. en editor. Nu ska du skaffa dig ett tomt dokument att skriva programkod i. Samtidigt ska du passa på att ge dokumentet ett bra namn.

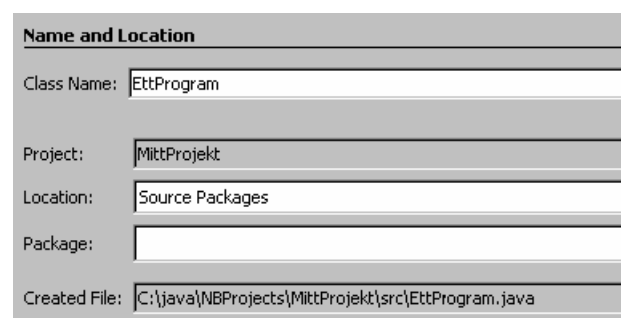
- Högerklicka **MittProjekt** eller **Source Packages** i projektfönstret. Välj sedan **New – Java Class...** i popup-menyn.



I dialogfönstret **New Java Class** ska du ange lite information, nämligen

- klassens namn (ska börja med stor bokstav),
- var klassen ska placeras
- om klassen ska vara i något paket.

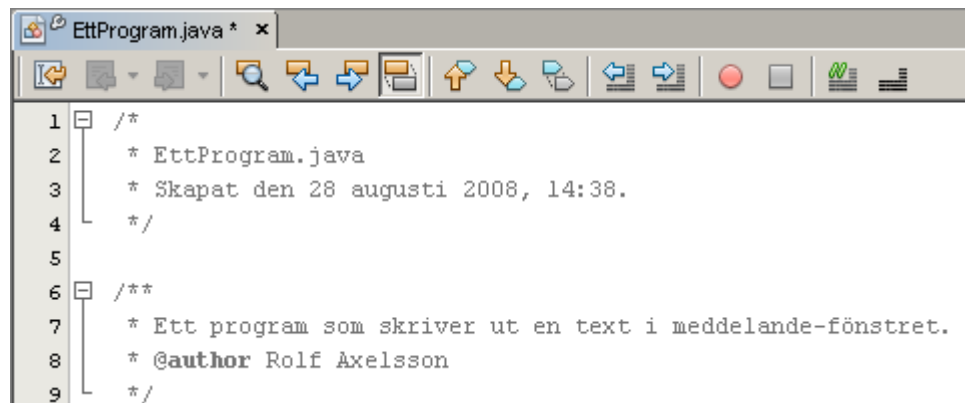
Ange att klassens namn ska vara **EttProgram** och klicka sedan på **Finish**.



Skriva in källkoden

Ändra nu lite i EttProgram.java.

- Ändra kommentarerna överst så de ser ut så här (använd eget namn):



```
1  /*
2   * EttProgram.java
3   * Skapat den 28 augusti 2008, 14:38.
4   */
5
6  /**
7   * Ett program som skriver ut en text i meddelande-fönstret.
8   * @author Rolf Axelsson
9   */
```

- Lägg till metoden **action** så att klassen EttProgram ser ut så här:

```
10 public class EttProgram {
11     public void action() {
12         System.out.println("-----");
13         System.out.println("Mitt första java-program");
14         System.out.println("-----");
15     }
16 }
```

Det är viktigt att göra indragningar i koden (inleda kodraden med ett antal mellanslag), sk indenteringar. Grundregeln för indentering är:

- Öka indenteringen efter {
- Minska indenteringen efter }

När du skriver in metoden **action** och instruktionerna som tillhör **action** (raderna som börjar med **System.**) så visar sig en röd markering till vänster om raden du just jobbar med. Denna röda markering är kvar så länge NetBeans inte tycker att det som står på raden går att översätta till bytekod.

Spara programmet efter att du ändrat i programmet. Du sparar programmet genom att välja **File – Save**. Om du tittar i **src**-katalogen ser du filen **EttProgram.java**. NetBeans placerar källkodsfilerna som tillhör projektet i **src**-katalogen.

Ett par kommentarer till programmet:

- Klassens namn är alltid samma som filens namn (utom suffixet .java). I ovanstående exempel heter klassen *EttProgram* och filen *EttProgram.java*. Eftersom klassnamn alltid ska börja med stor bokstav så måste filens namn börja med stor bokstav.
- Raden med klassnamn inleds som regel med **public**.
- I det här programmet är det koden i metoden **action** som ska exekveras lite senare i denna instruktion.

Om de gråfärgade kommentarerna:

I NetBeans är kommentarer gråfärgade. Kommentarer är dokumentation av java-programmet till för att förtydliga koden för dig själv och andra som jobbar med programmet. Det är viktigt att du skriver rikligt med kommentarer i dina program. Kommentarer tillhör inte programmet och påverkar inte programkörningen.

- Det är två typer av kommentarer i ovanstående exempel

Javsdoc-kommentar.

```
/**
 * Här skriver du kommentarer.
 * Varje rad börjar med en stjärna.
 */
```

Flerradskommentar

```
/*
 * Kommentar över flera rader.
 * Är bra om lite mer komplicerade saker ska förklaras
 */
```

- På kursen kommer du att stifta bekantskap med ytterligare ett sätt att skriva kommentarer, s.k. enradskommentar.

```
// Enradskommentar. Allt till höger om // är kommentar
```

3. Kompilera programmet

För att erhålla ett körbart program måste *källkoden* du skrivit översättas till *maskinkod*. Maskinkod är instruktioner som kan utföras av datorn. Att översätta källkoden till maskinkod kallas för att *kompilera* programmet. Det är ett speciellt program, en s.k. *kompilator*, som utför denna översättning.

I Java sker översättningen till maskinkod i två steg:

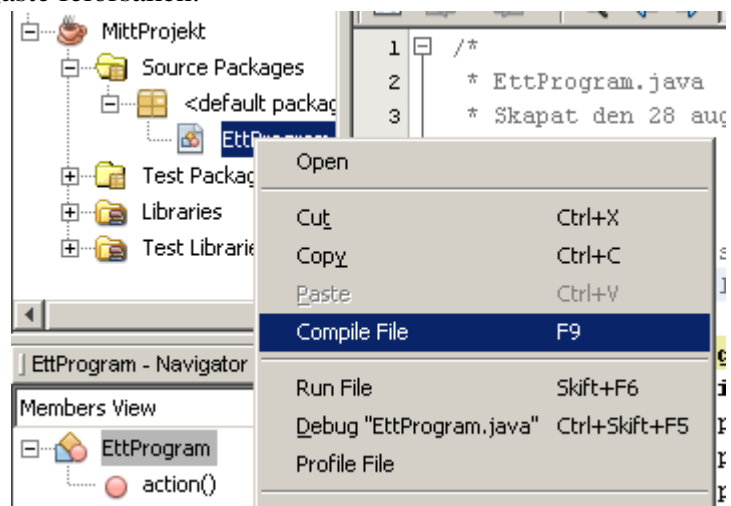
1. Först översätts källkoden till något som kallas *Java-bytekod*. Resultatet av denna översättning lagras i en class-fil. Programmet som utför denna översättning heter **javac.exe**. Program i java som är färdiga att exekveras (köras) lagras som en eller flera class-filer.
2. Sedan översätts bytekoden till maskinkod. Samtidigt körs maskinkoden. Översättningen av byte-koden och exekveringen av programmet sköts av programmet **java.exe** (kallas för JVM – Java Virtual Mashine).

För att källkoden skall kunna översättas till bytekod så måste den vara skriven enligt vissa mycket bestämda regler, den måste följa en bestämd *syntax*.

Editorn markerar med en röd symbol i kanten om den finner några felaktigheter i koden. Detta ger dig en möjlighet att rätta till felen innan du låter kompilera programmet. *Kompilatorn* hittar de syntaxfel som inte rättats till och ger anvisning om var felet kan finnas. Eftersom kompilatorn inte kan veta avsikten med programmet kan den inte heller veta exakt vad som är fel. Kompilatorn anger troligaste felorsaken.

Normalt sett kompilerar och exekverar du programmet på en gång när du arbetar i NetBeans. Men om du önskar att endast kompilera EttProgram.java så ska du *högerklicka*

EttProgram.java i Projektfönstret och sedan välja **Compile file**. Det går också bra att se till att EttProgram.java är markerad i projektfönstret och sedan trycka på F9.



Om kompileringen lyckas, dvs programmet inte innehåller några syntaxfel, så visas texten

BUILD SUCCESSFUL ...

neri i meddelandefönstret.

Men om programmet innehåller felaktigheter så får man ett eller flera felmeddelanden i meddelandefönstret. Om du klickar på länken i meddelandefönstret så markeras raden som innehåller felet. Rätta till felet och eventuella fler felaktigheter. Kompilera sedan programmet på nytt.

Resultatet av en lyckad kompileringen är att det skapas en **.class**-fil i katalogen **build/classes**. Kontrollera detta med hjälp av Utforskaren.

4. Köra programmet

För att köra programmet måste du skriva en **main**-metod. I java startar alltid ett program i en main-metod. Nedanstående main-metod kan placeras i vilken klass som helst (även i klassen EttProgram). Det enda viktiga är att klassen EttProgram är tillgänglig.

Skapa klassen **Main** (OBS! börjar på stor bokstav!) på samma sätt som du skapade EttProgram. Se till att Main-klassen har följande innehåll (ta inte bort kommentarerna):

```
public class Main {  
    public static void main( String[] args ) {  
        EttProgram prog = new EttProgram();  
        prog.action();  
    }  
}
```

Kodraden

```
EttProgram prog = new EttProgram();
```

gör att koden du skrivit i klassen EttProgram kan användas. Raden

```
prog.action();
```

gör att koden i metoden action exekveras.

```
public class EttProgram {  
    public void action() {  
        System.out.println("-----");  
        System.out.println("Mitt första java-program");  
        System.out.println("-----");  
    }  
}
```

Nu ska du kompilera klassen Main på samma sätt som du gjorde med EttProgram. Och får du något kompileringsfel så måste du åtgärda det och sedan kompilera klassen på nytt.

När programmet är utan syntax-fel går det att köra. I java körs byte-koden (lagras i class-filen) med hjälp av ett program, en s.k. "Java Virtual Mashine". Det är ett speciellt program som exekverar bytekoden i class-filer.

För att exekvera programmet kan du göra något av följande:

- Högerklicka på **Main.java** - Välj **Run File**
- Högerklicka i **Main-dokumentet** - välj **Run Main.java**
- Markera Main.java och tryck på **SHIFT-F6**.

Resultatet av körningen är en utskrift i meddelandefönstret (Output). Utskriften blandas med några utskrifter från NetBeans-miljön.

```
Output - MittProjekt (run-single)  
  
init:  
deps-jar:  
compile-single:  
run-single:  
-----  
Mitt första java-program  
-----  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Passa på att skriva in vettiga kommentarer överst i Main.java. De kan t.ex. se ut så här:

```
1  /*  
2   * Main.java  
3   * Skapad den 28 augusti 2008, 15:03  
4   */  
5  
6  /**  
7   * Klassen Main startar programmet EttProgram genom att anropa metoden action.  
8   * @author TSROAX  
9   */
```

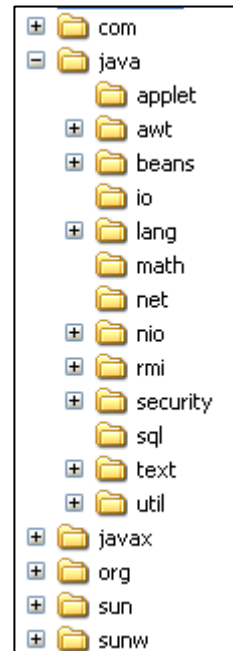
Paket

I programmeringsspråket java finns det ett stort antal färdigskrivna klasser som programmerare kan använda sig av. Dessa klasser lagras i en katalogstruktur där varje katalog kallas för ett paket.

I figuren till höger ser du huvudpaketerna, bl.a *java* och *javax*. Dessa paket innehåller i sin tur paket, s.k. *underpaket*.



- Paketet *java* är öppnat i figuren till höger. Du ser att *java* innehåller ett antal paket. Dessa är: *applet*, *awt*, *beans*, *io*, *lang*, *math*, *net*, *nio*, *rmi*, *security*, *sql*, *text*, *util*. Dessa paket innehåller i sin tur klasser och ofta även underpaket.
De kursiverade paketen innehåller klasser som kommer att användas på kursen.
Paketerna namn uttrycker hela katalogstrukturen. T.ex. heter ovanstående underpaket: *java.applet*, *java.awt* osv
- Paketet *javax* innehåller också ett antal underpaket. Ett viktigt sådant på kursen är *swing*. Klassen *JOptionPane*, vilken finns i *javax.swing*, ska du använda i exemplet nedan.



Importera paket

Ett speciellt viktigt paket i java är *java.lang*. Alla klasser som finns i paketet *java.lang* kan användas direkt i java-program. Och endast dessa klasser. Vill man använda en klass som finns i ett annat paket (det vill man) så måste man ange detta i sitt program – man måste *importera* klassen.

Med raden

```
import javax.swing.JOptionPane;
```

anger man att klassen *JOptionPane* i paketet *javax.swing* ska användas i programmet (egentligen i den klass man arbetar med).

Nu ska du ändra i programmet för att testa att mata in tecken via dialog-fönster.

Gör de ändringar som du ser i figurerna nedan. Efter figurerna förklaras de kortfattat.

```
1  /*
2   * EttProgram.java
3   * Skapat den 28 augusti 2008, 14:38.
4   */
5  import javax.swing.JOptionPane;
6  /**
7   * Ett program som skriver ut en text i meddelande-fönstret.
8   * @author Rolf Axelsson
9   */
10 public class EttProgram {
11     public void action() {
12         String name = JOptionPane.showInputDialog( "Ange ditt namn" );
13         System.out.println( "-----" );
14         System.out.println( "Ett program skrivet av " + name );
15         System.out.println( "-----" );
16     }
17 }
```

```

1  /*
2   * Main.java
3   * Skapad den 28 augusti 2008, 15:03
4   */
5
6  /**
7   * Klassen Main startar programmet EttProgram genom att anropa metoden action.
8   * @author TSROAX
9   */
10 public class Main {
11     public static void main( String[] args ) {
12         EttProgram prog = new EttProgram();
13         prog.action();
14         javax.swing.JOptionPane.showMessageDialog( null, "Nu är programmet slut!" );
15     }
16 }

```

För att kompilatorn skall hitta metoden **JOptionPane.showInputDialog(String)** så måste du tala om för kompilatorn var klassen finns. Ett sätt att göra detta är att skriva

```
import javax.swing.JOptionPane;
```

Denna rad ska stå ovanför klassen (se EttProgram).

Det går även bra att importera samtliga klasser som finns i ett paket:

```
import javax.swing.*;
```

Om du skriver på detta sätt kan alla klasser i *javax.swing*-paketet användas i programmet. Du kan testa genom att byta JOptionPane mot * i import-raden i EttProgram.

Det går bra att använda en klass även om man inte importerat klassen / paketet med klassen. Men man måste i detta fallet ange klassens fullständiga namn, dvs. även det paket som klassen är placerad i. Detta ges exempel på i klassen Main:

```
javax.swing.JOptionPane.showMessageDialog( null, "Nu är ..." );
```

Flera main-metoder

Slutligen kan du kopiera main-metoden i klassen Main och klistra in den sist i klassen EttProgram. Sedan kan du högerklicka EttProgram och välja Run File. EttProgram kompileras först och sedan startar programmet i main-metoden i EttProgram.

Detta visar att main-metoden kan placeras i vilken klass som helst.

```

1  /*
2   * EttProgram.java
3   * Skapat den 28 augusti 2008, 14:38.
4   */
5  import javax.swing.JOptionPane;
6  /**
7   * Ett program som skriver ut en text i meddelande-fönstret.
8   * @author Rolf Axelsson
9   */
10 public class EttProgram {
11     public void action() {
12         String name = JOptionPane.showInputDialog( "Ange ditt namn" );
13         System.out.println( "-----" );
14         System.out.println( "Ett program skrivet av " + name );
15         System.out.println( "-----" );
16     }
17
18     public static void main( String[] args ) {
19         EttProgram prog = new EttProgram();
20         prog.action();
21         javax.swing.JOptionPane.showMessageDialog( null, "Nu är programmet slut!" );
22     }
23 }

```