

Att använda Java SE – JDK 6

Programmeringsspråket Java är utvecklat av det amerikanska företaget Sun Microsystems. Sun erbjuder gratis en utvecklingsmiljö för java-programmering, Java 2 SE (Standard Edition). JDK som dyker upp ibland står för Java Development Kit och används synonymt med Java SE. I denna utvecklingsmiljö ingår bl.a. programmen

- **javac.exe** för att kompilera källkod till bytekod
- **java.exe** för att exekvera bytekod
- **javadoc.exe** för att dokumentera klasser
- **appletviewer.exe** för att köra applets utan browser
- **jar.exe** för att hantera Java ARchive filer

Just nu är den aktuella versionen 6. På kurssidan är det en länk till nerladdningssidan på Sun.

- Om du har hämtat hem NetBeans och installerat programmet så är en version av JSE installerad. Du hittar exe-filerna i Program/java/jdk1.6.0_02 (kan vara annan jdk-version)
- Om du tänker arbeta med Eclipse så ska du följa instruktionerna under rubriken INSTALLATION AV JSE 6 (Windows)

En god idé är att ladda ner dokumentationen JAVA SE 6 Documentation. Dokumentationen är mycket användbar när du skriver program.

INSTALLATION AV JSE 6 (Windows)

Filen du hämtar hem heter **jdk-6u7-windows-i586-p.exe**

Dubbeltklicka filen och klicka vidare vid dialoger så installeras allt enligt planerna.

För att exekvera exe-filerna ovan måste miljövariabeln PATH innehålla sökvägen till katalogen där exe-filerna finns, dvs: C:\Program\Java\jdk1.6.0_07\bin

Kompilatorn (javac.exe) och interpretatorn (java.exe) använder CLASSPATH för att finna användardefinierade filer (både java-filer och class-filer). Ett par sökvägar bör finnas i CLASSPATH:

- Sökväg till aktuell katalog. Detta eftersom java-filen som ska kompileras / class-filen som ska köras normalt är i aktuell katalog. Därför ska du ange en punkt i CLASSPATH.
- Sökväg till egendefinierade paket som används i klasserna som ska kompileras / köras. Vi antar att du kommer att placera sådana paket i katalogen C:\java. Därför ska du ange C:\Java i CLASSPATH.

WINDOWS 95-98

Du måste göra vissa tillägg i **autoexec.bat**. Väljer **Start-Run**, skriv **sysedit** och klicka OK. Nu lägger du till följande i autoexec.bat:

```
SET CLASSPATH=.;C:\JAVA;  
SET PATH= C:\Program\Java\jdk1.6.0_07\bin;%PATH%
```

och sedan sparar du tilläggen. Du måste göra omstart (eller köra AUTOEXEC.BAT) efter ovanstående ändringar.

WINDOWS 2000-XP, (Vista)

Välj Kontrollpanel – System – Avancerat – Miljövariabler och gör ovanstående ändringar, dvs se till att PATH innehåller värdet C:\Program\Java\jdk1.6.0_07\bin och att CLASSPATH innehåller värdena .;C:\JAVA.

Följande stycken är att betrakta som överkurs och inte nödvändiga att utföra som en del av labb 1.

Kompilera med javac.exe


Skriv följande program och spara det under namnet **Test1.java** i katalogen C:\java\java2sdk. För att skriva in programmet kan du t.ex. använda Notepad.

```
import javax.swing.*;

public class Test1 {
    public static void main( String[] args ) {
        String name;
        name = JOptionPane.showInputDialog( "Mata in ett namn" );
        System.out.println( "Du matade in " + name );
    }
}
```

Nu är det dags att kompilera Test1.java.

- Se till att C:\java\java2sdk är aktuell katalog i dos-fönstret.
 - * Om du vill gå ur en katalog skriver du **cd..** och trycker på ENTER.
 - * Om du vill gå in i en katalog så skriver du **cd katalognamn** och trycker på ENTER.
- Kompilera programmet med
javac Test1.java
Om programmet är utan fel så bör det gå bra.



```
C:\>cd java
C:\java>cd java2sdk
```



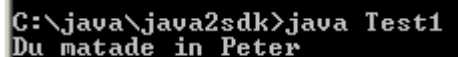
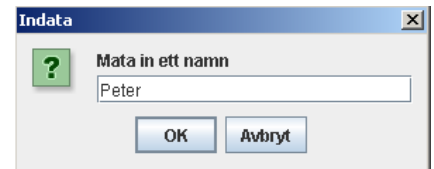
```
C:\java\java2sdk>javac Test1.java
```

Nu ska du kontrollera att filen **Test1.class** finns i katalogen C:\java\java2sdk.

Exekvera med java.exe

Om ovanstående program lät sig kompileras utan fel så bör det vara körbart med hjälp av java.exe.

- Se till att C:\java\java2sdk är aktuell katalog i dos-fönstret.
- Exekvera programmet med
java Test1
Körresultatet bör vara som det i figuren (om du matar in Peter och trycker på ENTER).



```
C:\java\java2sdk>java Test1
Du matade in Peter
```

Kompilera till paket

När du ska placera en klass i ett paket bör du alltid spara java-filen i en katalog med paketets namn.

Skriv nedanstående program (kopiera till Notepad – enstaka tecken blir fel) och spara det under namnet Test2.java. Spara programmet i katalogen C:\java\java2sdk\test.

```
package test;

public class Test2 {
    // Skriver ett tal i procentform med 2 decimaler
    public static void procent(double decimal) {
        String procentStr = String.format( "%.2f", decimal*100 );
        System.out.println( procentStr + "%" );
    }
}
```

Nu är det dags att kompilera Test2.java. Följande gäller:

- Aktuell katalog ska vara C:\java\java2sdk.
- Kompileringen sker med instruktionen

```
javac test/Test2.java
```

```
C:\java\java2sdk>javac test/Test2.java
```

När kompileringen gått felfritt ska du kontrollera att filen **Test2.class** finns i katalogen C:\java\java2sdk\test.

Om paketet test ska kunna användas av olika projekt så måste du placera katalogen test på ett ställe dit någon sökväg i CLASSPATH leder. Det lämpligaste stället är säkert C:\java.

Kopiera därför katalogen **test** till C:\java.

Om du däremot endast tänker använda paketet test tillsammans med program som ligger i C:\java\java2sdk så behöver du inte kopiera katalogen till C:\java. Men detta förutsätter att C:\java\java2sdk är aktuell katalog vid programkörningen.

Skriv (kopiera) nedanstående testprogram, Test2Test.java, och spara filen i C:\temp. Kompilera sedan programmet och kör det. Glöm inte att C:\temp ska vara aktuell katalog vid kompilering och exekvering.

```
C:\java\java2sdk>cd \temp
C:\temp>javac Test2Test.java
C:\temp>java Test2Test
0.032 i procentform blir 3.20%
```

```
import test.*; // Även import test.Test2; går bra

public class Test2Test {
    public static void main(String[] args) {
        System.out.print("0.032 i procentform blir ");
        Test2.procent(0.032);
    }
}
```

Testa klasser som placeras i ett paket

Ofta har man en main-metod i en klass som ska placeras i ett paket, i vart fall medan man utformar klassen. Men detta innebär en liten teknikalitet när man ska starta exekveringen i main-metoden. På samma sätt som när man kompilerar klassen så måste paketnamnet vara med när man exekverar main-metoden.

Skriv till nedanstående main-metod i klassen Test2.java. Du ska använda Test2.java som är lagrad i katalogen C:\java\java2sdk\test.

```
package test;
import javax.swing.JOptionPane;

public class Test2 {
    // Skriver ett tal i procentform med 2 decimaler
    public static void procent(double decimal) {
        String procentStr = String.format( "%.2f", decimal*100 );
        System.out.println( procentStr + "%" );
    }

    public static void main(String[] args) {
        String tal = JOptionPane.showInputDialog( "Ange ett decimaltal" );
        double decimaltal = Double.parseDouble( tal );
        System.out.print("Procentform av " + tal + ": ");
        Test2.procent(decimaltal);
    }
}
```

Kompilera filen i vanlig ordning.

Exekvera slutligen filen med

```
java Test2 // fel klassnamn, ska vara test/test2
```

```
C:\temp>cd \java\java2sdk
C:\java\java2sdk>javac test/Test2.java
C:\java\java2sdk>java Test2.java
Exception in thread "main" java.lang.NoClassDefFoundError: Test2/java
```

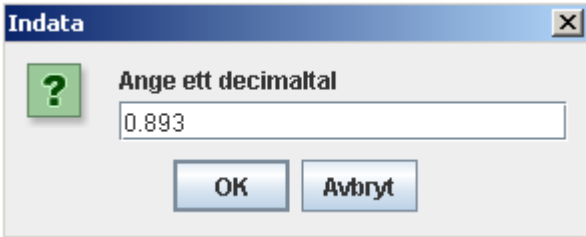
Som du märker signalerar systemet ett fel, nämligen att klassen inte hittas. När en klass placeras i ett paket så ingår nämligen paketnamnet i klassnamnet. Klassen ska heta: `test/test2`

För att exekvera mainmetoden ska du använda något av nedanstående skrivsätt:

```
java test/Test2          eller
java test.Test2
```

Nu bör det fungera.

```
C:\java\java2sdk>java test/Test2
```



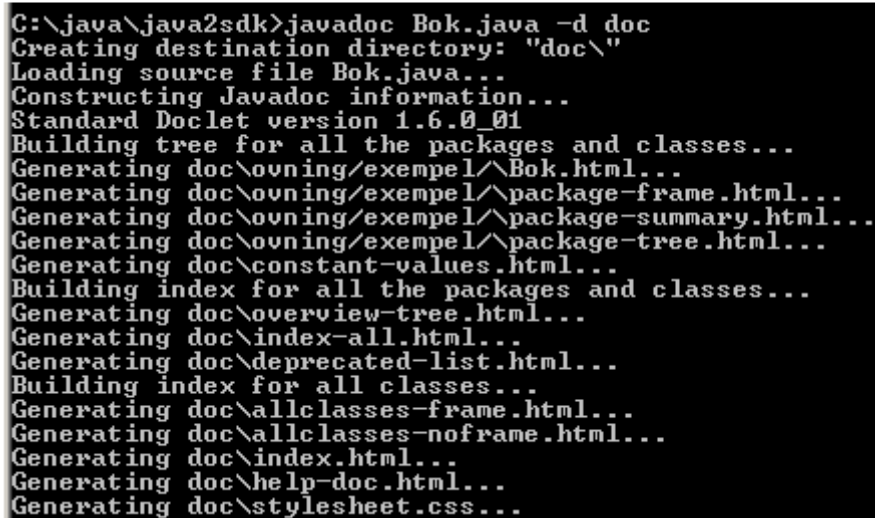
```
C:\java\java2sdk>java test/Test2
Procentform av 0.893: 89,30%
C:\java\java2sdk>
```

Använda javadoc.exe

Programmet javadoc.exe hjälper Java-programmerare med användar-dokumentationen av egendefinerade klasser. Genom att skriva "javadoc-kommentarer" på speciella ställen i klassen och med hjälp av speciella "javadoc-taggar" kan man uppnå en god dokumentation.

Vi börjar med att köra javadoc.exe med en fil utan speciella javadoc-kommentarer. Gör följande:

- Hämta Bok.java (finns i JSE.zip på kurssidan) till katalogen C:\java\java2sdk.
- Skapa katalogen C:\java\java2sdk\doc. I denna katalog ska vi placera den automatgenererade dokumentationen.
- Se till att C:\java\java2sdk är aktuell katalog och skriv (följt av ENTER)
javadoc Bok.java -d doc



```
C:\java\java2sdk>javadoc Bok.java -d doc
Creating destination directory: "doc\"
Loading source file Bok.java...
Constructing Javadoc information...
Standard Doclet version 1.6.0_01
Building tree for all the packages and classes...
Generating doc\ovning\exempel\Bok.html...
Generating doc\ovning\exempel\package-frame.html...
Generating doc\ovning\exempel\package-summary.html...
Generating doc\ovning\exempel\package-tree.html...
Generating doc\constant-values.html...
Building index for all the packages and classes...
Generating doc\overview-tree.html...
Generating doc\index-all.html...
Generating doc\deprecated-list.html...
Building index for all classes...
Generating doc\allclasses-frame.html...
Generating doc\allclasses-noframe.html...
Generating doc\index.html...
Generating doc\help-doc.html...
Generating doc\stylesheet.css...
```

Nu skapas ett större antal html-filer. -d doc innebär att de placeras i underkatalogen **doc**. För tillfället är vi bara intresserade av **Bok.html**. Resterande html-filer kan du ta bort.

OBS! Om du inte hittar Bok.html kan det bero på att klassen är placerad i ett paket. Filen Bok.java på kurssidan innehåller satsen

```
package ovning.exempel;
```

Det innebär att klassen **Bok** finns i paketet **exempel** vilket i sin tur ligger i paketet **ovning**. javadoc.exe skapar samma katalogmönster som anges i package-satsen. Det innebär att du i detta fall måste titta i katalogen C:\java\java2sdk\doc\ovning\exempel..

Dubbeltklicka Bok.html i Utforskaren och studera resultatet. Som du ser är dokumentet uppdelat i tre delar:

- information om klassen. Det handlar om paket, arv och implementerade interface (inga i Bok).
- sammanfattningar om attribut, konstruktorer och metoder. Om klassen innehåller publika attribut sammanfattas de under Fields.
- Mer detaljerad information om attribut, konstruktorer och metoder. Eftersom vi inte skrivit några javadoc-kommentarer så är det ännu inga detaljer.

Nu är det dags att lägga till lite kommentarer i Bok.java. javadoc-kommentarer startar alltid med /** och avslutas med */.

Information om klassen/gränssnittet

Information om klassen/gränssnittet skrivs efter package- och import-satser men före class/interface-definitionen. Ett par javadoc-taggar som kan användas här är:

@author – Författare av klassen. Visar sig endast om –author används när javadoc.exe körs

@version – Version av klassen. Visar sig endast om –version används när javadoc.exe körs

@see – Hänvisning till metoder i klassen eller till andra klasser.

Lägg till följande kommentarer i Bok.java (lägg märke till att man kan lägga in HTML-taggar):

```
package ovning.exempel;  
import extra.*;  
/**  
 * Klassen Bok innehåller information om en bok. Viktiga attribut är  
 * bokens titel och isbn-nummer.  
 * <p>  
 * @version 1.0, 13/9-2006  
 * @author Rolf Axelsson  
 * @see java.lang.String  
 */  
  
public class Bok {  
    :
```

Generera sedan javadoc-dokumentation med

```
javadoc Bok.java -author -version -d doc
```

```
C:\java\java2sdk>javadoc Bok.java -author -version -d doc  
Loading source file Bok.java...  
Constructing javadoc information...
```

När du gjort detta så kan du dubbelklicka Bok.html på nytt (eller göra refresh på det gamla). Nu ser det ut ungefär som nedanstående figur.

ovning.exempel

Class Bok

java.lang.Object
└─ovning.exempel.Bok

public class **Bok**
extends java.lang.Object

Klassen Bok innehåller information om en bok. Viktiga attribut är bokens titel och isbn-nummer.

Version:
1.0, 13/9-2006

Author:
Rolf Axelsson

See Also:
String

Information om konstruktörer och metoder

Javadoc-kommentarer till konstruktörer och metoder (och attribut) skrivs direkt ovanför definitionerna. Speciella javadoc-taggar man bör använda är:

- @param – Parametrar till konstruktorn/metoden
- @return – Returvärde från metod
- @see - Hänvisning

Lägg till följande kommentarer i Bok.java:

```
public class Bok {
    private String titel;
    private String isbn;

    /**
     * Konstruerar och initialiserar en bok utan titel och isbn-nummer.
     */
    public Bok() {
        this("", "");
    }

    /**
     * Konstruerar och initialiserar en bok med angiven titel och
     * ISBN-nummer.
     * @param titel Bokens titel.
     * @param isbn Bokens ISBN-nummer som en sträng.
     */
    public Bok(String titel, String isbn) {
        this.titel = titel;
        this.isbn = isbn;
    }

    /**
     * Returnerar en boks titel.
     * @return Bokens titel.
     */
    public String getTitel() {
        return titel;
    }

    public String getIsbn() {
        return isbn;
    }

    /**
     * Anger bokens titel. Bok-objektet kommer att hålla en referens till
     * parametern <code>titel</code>.
     * @param titel Bokens titel.
     */
    public void setTitel(String titel) {
        this.titel = titel;
    }

    public void setIsbn(String isbn) {
        this.isbn = isbn;
    }

    public String toString() {
        return titel+", ISBN: "+isbn;
    }
}
```

Spara ändringarna och kör javadoc igen. Dubbelklicka sedan på Bok.html. Följande förändringar kan du lägga märke till:

- Under **Constructor summary** och **Method summary** har det förts in information. Det är den första meningen i javadoc-kommentaren som visas här.

Constructor Summary	
Bok ()	Konstruerar och initialiserar en bok utan titel och isbn-nummer.
Bok (java.lang.String titel, java.lang.String isbn)	Konstruerar och initialiserar en bok med angiven titel och ISBN-nummer.

Method Summary	
java.lang.String	getIsbn ()
java.lang.String	getTitel () Returnerar en boks titel.
void	setIsbn (java.lang.String isbn)
void	setTitel (java.lang.String titel) Änkar bokens titel.
java.lang.String	toString ()

- Under **Constructor Detail** och **Method Detail** står den javadoc-kommenterade texten. Lägg speciellt märke till hur @param och @return fungerar. Du kan också lägga märke till HTML-taggen i kommentaren till setTitle.

Constructor Detail

Bok

```
public Bok()
```

Konstruerar och initialiserar en bok utan titel och isbn-nummer.

Bok

```
public Bok(java.lang.String titel,
           java.lang.String isbn)
```

Konstruerar och initialiserar en bok med angiven titel och ISBN-nummer.

Parameters:

titel - Bokens titel

isbn - Bokens ISBN-nummer som en sträng.

Method Detail

getTitel

```
public java.lang.String getTitel()
```

Returnerar en boks titel.

Returns:

Bokens titel.

Använda jar.exe

På distanskursen ska vi använda jar.exe för att förpacka inlämningsuppgifterna i en fil, en s.k. jar-fil (Java Archive File). Denna fil ska innehålla allt material som ska lämnas in, även samtliga class-filer i inlämningsuppgiften. Det är flera fördelar med detta:

- Allt material är samlat i en fil ur vilken jag kan packa upp valda delar. Dessutom är filerna som ingår komprimerade.
- Du testkör class-filerna som förpackats i jar-filen. Fungerar programmet i hemmet så fungerar det på min dator. Men det gäller att alla class-filer tagits med, förutom Javas standardklasser. De finns redan representerade på dator (i filen rt.jar – leta upp den).

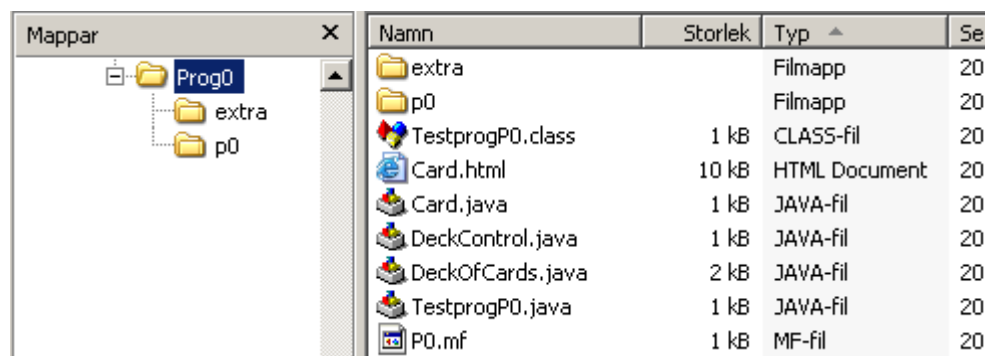
Så här gör du:

I nedanstående exempel används ett antal filer. Du finner dessa i AnvandaJar.zip på kurssidan. Du kan hämta hem dem och genomföra nedanstående instruktioner.

Vi tänker oss att inlämningen ska innehålla följande filer/dokument:

- **TestprogP0.java** – Testprogram som du skrivit (innehåller main-metod i vilken exekveringen av programmet startar). Klassen är inte placerad i något paket. I TestprogP1 används följande klasser vilka inte är standardklasser i Java:
 - * **Output** från det egna paketet **extra**.
- **Card.java, DeckOfCards.java, DeckControl.java** – klasser som du skrivit. Båda klasserna är placerade i paketet **p0**.
- **Card.html** dokumentationsfil av klassen Card.
- **class-filer** så att TestprogP0 är körbart. Det innebär att alla klassfiler som inte är standardklasser i J2SE 1.5.0 ska finnas i jar-filen. I det här fallet innebär det
 - * **TestprogP0.class**
 - * **Card.class, DeckOfCards.class** och **DeckControl.class**. Båda ska vara placerade i en katalog med namnet **p0**.
 - * **Output.class**. class-filen ska vara placerade i en katalog med namnet **extra**.
- För att jar-filen ska bli körbar ska den innehålla en **manifestfil**. Manifestfilen ska innehålla en rad med text och under den raden en tomrad (måste vara en tomrad, annars blir inte jar-filen körbar). I detta exempel heter manifestfilen **P0.mf**. Så här ska P0.mf se ut:
Main-Class: TestprogP0

Samla ovanstående filer i en katalog på hårddisken, t.ex. **C:\java\Prog0**.



Namn	Storlek	Typ	Se
extra		Filmapp	20
p0		Filmapp	20
TestprogP0.class	1 kB	CLASS-fil	20
Card.html	10 kB	HTML Document	20
Card.java	1 kB	JAVA-fil	20
DeckControl.java	1 kB	JAVA-fil	20
DeckOfCards.java	2 kB	JAVA-fil	20
TestprogP0.java	1 kB	JAVA-fil	20
P0.mf	1 kB	MF-fil	20

The screenshot shows a file explorer window titled 'Mapper' on the left and a table on the right. The file explorer displays a folder structure: 'Prog0' containing 'extra' and 'p0'. The 'p0' folder is selected. The table on the right lists files with columns: 'Namn', 'Storlek', 'Typ', and 'Se'. It contains three entries: 'Card.class' (1 kB, CLASS-fil), 'DeckControl.class' (1 kB, CLASS-fil), and 'DeckOfCards.class' (2 kB, CLASS-fil).

Namn	Storlek	Typ	Se
Card.class	1 kB	CLASS-fil	20
DeckControl.class	1 kB	CLASS-fil	20
DeckOfCards.class	2 kB	CLASS-fil	20

The screenshot shows a file explorer window titled 'Mapper' on the left and a table on the right. The file explorer displays a folder structure: 'Prog0' containing 'extra' and 'p0'. The 'extra' folder is selected. The table on the right lists files with columns: 'Namn', 'Storlek', 'Typ', and 'Se'. It contains one entry: 'Output.class' (3 kB, CLASS-fil).

Namn	Storlek	Typ	Se
Output.class	3 kB	CLASS-fil	20

Gå till katalog C:\java\Prog0 i dos-fönstret och skriv:

```
jar cmvf P0.mf TestP0.jar *.*
```

Nu kommer jar.exe att tala om vilka filer som placeras i jar-filen och hur komprimerade de är.

```

C:\java\Prog0>jar cmvf P0.mf TestP0.jar *.*
extra manifestfil
l gger till: Card.html <in = 9403> <ut = 1956> <79% komprimerat>
l gger till: Card.java <in = 720> <ut = 311> <56% komprimerat>
l gger till: DeckControl.java <in = 738> <ut = 367> <50% komprimerat>
l gger till: DeckOfCards.java <in = 1076> <ut = 507> <52% komprimerat>
l gger till: extra/ <in = 0> <ut = 0> <0% lagrat>
l gger till: extra/Output.class <in = 2246> <ut = 1131> <49% komprimerat>
l gger till: p0/ <in = 0> <ut = 0> <0% lagrat>
l gger till: p0/Card.class <in = 497> <ut = 312> <37% komprimerat>
l gger till: p0/DeckControl.class <in = 814> <ut = 555> <31% komprimerat>
l gger till: p0/DeckOfCards.class <in = 1204> <ut = 762> <36% komprimerat>
l gger till: P0.mf <in = 24> <ut = 26> <-8% komprimerat>
l gger till: TestprogP0.class <in = 744> <ut = 475> <36% komprimerat>
l gger till: TestprogP0.java <in = 470> <ut = 267> <43% komprimerat>

C:\java\Prog0>

```

Testa slutligen att programmet  r k rbart med

```
java -jar TestP0.jar
```

K rresultatet ska bli en dialog med f ljande utseende. Klicka OK s  avslutas programmet



Ett alternativ till att starta programmet  r att dubbelklicka TestP0.jar. Testa  ven detta.