

Laboration 2: Introduktion - programutveckling med java

Laborationens syfte är att du ska skriva, kompilera och exekvera några program skrivna i programmeringsspråket java. Laborationen består av tre delar:

I del 1 av laborationen kommer du använda program som finns på datorn och program som levereras med JDK (Java development kit) när du jobbar med ett par enkla program.

I del 2 får du stifta bekantskap med utvecklingsverktyget Eclipse. Eclipse förenklar många delar av programutvecklingen och vi kommer att använda Eclipse på kursen.

I del 3 ska du utveckla ett antal program på egen hand.

Laborationen är anpassad för datoralarna här på Malmö högskola.

För att del 1 av laborationen ska kunna genomföras på din egen dator måste ett JDK – Java Development Kit – vara installerat på datorn. Börja med att hämta hem dokumentet JSE.pdf från kurssidan och följ instruktionerna. *setpath.bat* ska du inte köra på egen dator.

Del 1 förutsätter att du jobbar under operativsystemet Windows (Kommandotolken, Utforskaren, Anteckningar). Du kan använda motsvarande verktyg under andra operativsystem

För att del 2 och del 3 ska kunna genomföras på din egen dator måste Eclipse vara installerat på datorn. Börja med att hämta hem och installera JDK (se ovan). Sedan laddar du hem Eclipse från www.eclipse.org/downloads, välj *Eclipse IDE for Java Developers*. Sedan packar du upp katalogen *eclipse* ur arkivfilen du hämtat hem. Med filen *eclipse.exe* startar du programmet.

Del 1

Uppgift 1a

Att stifta bekantskap med några olika program som kan användas vid programmering med Java.

Program som används i del 1 av laborationen

Windows Explorer	Bl.a.för att flytta, kopiera, ta bort eller lägga till filer och kataloger. Och för att se innehållet i kataloger.
NotePad	Används som editor för att jobba med källkod (dvs vanlig text). T.ex. skriva och spara källkod.
Command Prompt	För att köra javac.exe och java.exe.
javac.exe	För att översätta (kompilera) en källkodsfil till bytekodsfil.
java.exe	För att exekvera en bytekodsfil.

Editor

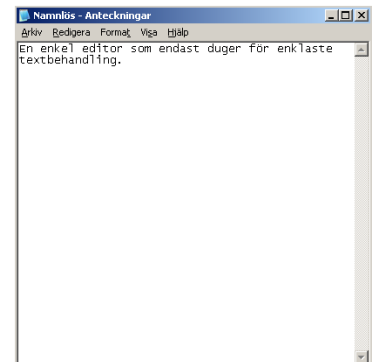
En *editor* är ett program i vilket man kan jobba med text. Här på datavetenskapen använder vi editorn för att skriva de instruktioner som ska utföras i programmet. Det är viktigt att du följer reglerna för hur instruktionerna ska skrivas. I java är det dessutom viktigt att skilja på stor och liten bokstav. Instruktionerna som programmet består av kallas för källkod. När du sparar instruktionerna i en fil på hårddisken (t.ex. Lab3Exercise1.java nedan) så kallas filen för *källkodsfil*..


I laborationen ska du använda programmet *Notepad* (Anteckningar) som editor. Starta *NotePad* genom att välja **Start – All Programs – Accessories - NotePad**.

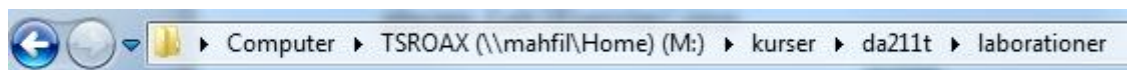


Start

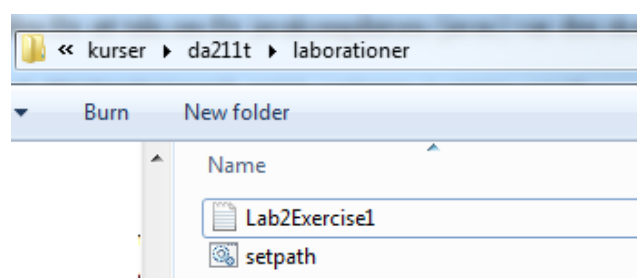
Ibland skriver man ett program från grunden och ibland jobbar man vidare med ett program som redan är skrivet. Just nu ska du jobba med ett redan färdigt program. På kurssidan hittar du programmet *Lab2Exercise1.java*. Men innan du hämtar hem programmet så ska du skapa en katalog på enhet M i vilken du ska placera *Lab2Exercise1.java*.



Starta *Windows Explorer* (Klicka på ) och skapa underkatalogen *kurser* på enhet M. Skapa därefter katalogen *da211t* i katalogen *kurser*. Slutligen ska du skapa katalogen *laborationer* i katalogen *da211t*.

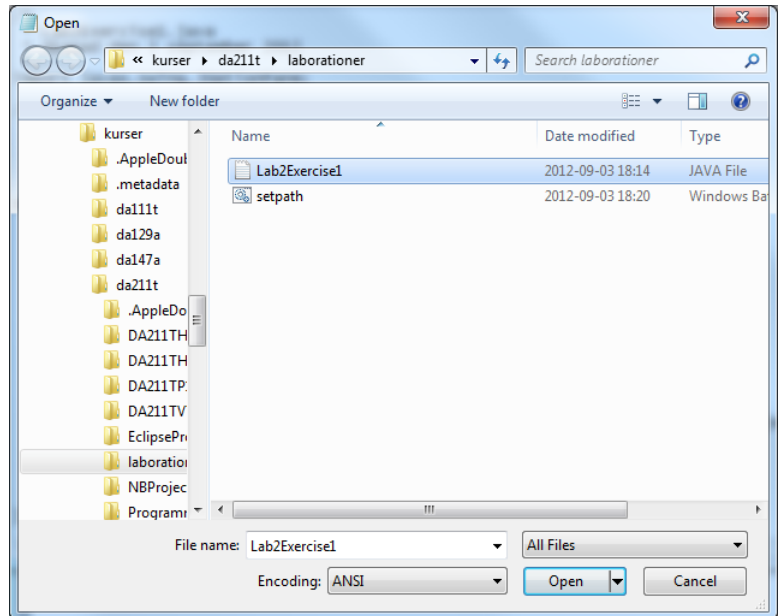


Hämta filerna *Lab2Exercise1.java* och *setpath.bat* från hemsidan och placera filerna i katalogen *laborationer*.

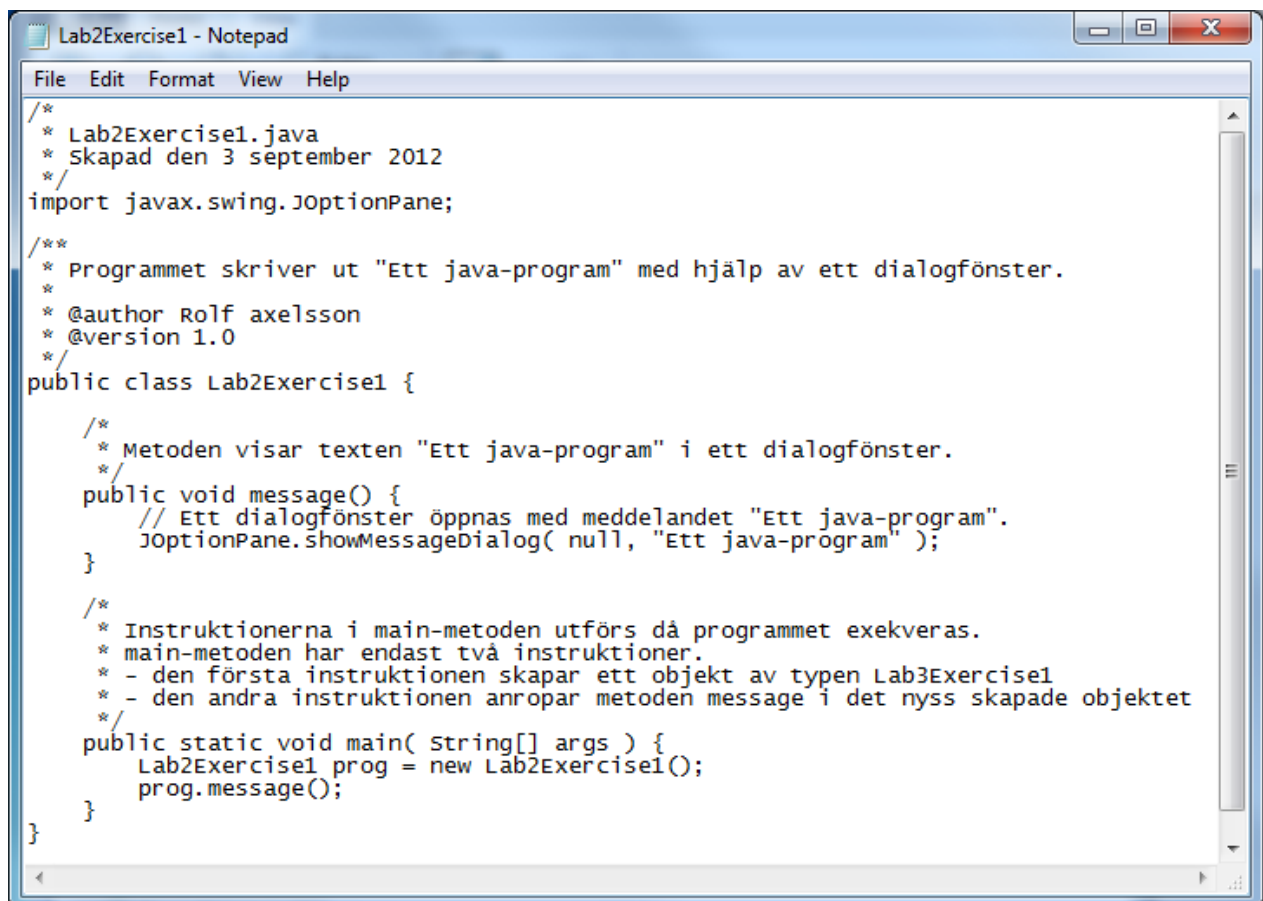


Gå till **NotePad** och öppna Lab2Exercise1.java.

Du måste ändra *Filformat* till *All files* för att Lab2Exercise1.java ska synas.



Nu ska filen visa sig i **NotePad**:



```
/*
 * Lab2Exercise1.java
 * Skapad den 3 september 2012
 */
import javax.swing.JOptionPane;

/**
 * Programmet skriver ut "Ett java-program" med hjälp av ett dialogfönster.
 *
 * @author Rolf axelsson
 * @version 1.0
 */
public class Lab2Exercise1 {

    /**
     * Metoden visar texten "Ett java-program" i ett dialogfönster.
     */
    public void message() {
        // Ett dialogfönster öppnas med meddelandet "Ett java-program".
        JOptionPane.showMessageDialog( null, "Ett java-program" );
    }

    /**
     * Instruktionerna i main-metoden utförs då programmet exekveras.
     * - den första instruktionen skapar ett objekt av typen Lab3Exercise1
     * - den andra instruktionen anropar metoden message i det nyss skapade objektet
     */
    public static void main( String[] args ) {
        Lab2Exercise1 prog = new Lab2Exercise1();
        prog.message();
    }
}
```

Ändra inte i filen just nu.

Kompilator

När du skrivit in och sparat ett program som en textfil (tänk dig att du har skrivit Lab2Exercise1.java) är det dags att *översätta* programmets instruktioner till ett format som kan utföras av datorn. Denna översättning kallas för att *kompilera* källkoden. Det är ett speciellt program som utför kompileringen, nämligen **javac.exe**. Programmet ingår i JDK. Programmet kallas för en *kompilator*.

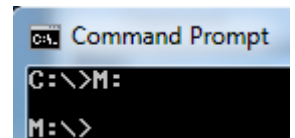
Programmet **javac.exe** använder du från ett annat program, nämligen **Command Prompt** (Kommandotoken) . Starta **Command Prompt** genom att välja **Start – All Programs – Accessories - Command Prompt**.

Nu öppnas ett fönster i vilket en enhet på datorn anges. I nedanstående fönster är det enhet M.



Nu gäller det att navigera till katalogen som innehåller Lab2Exercise1.java.

Om enhet C är aktuell i ditt fönster ska du ändra till enhet M genom att skriva M: följt av RETURN (dvs . knappen med ↵)

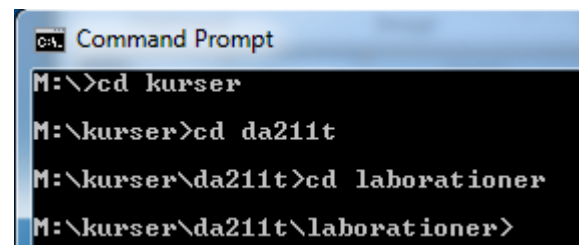


Navigera till katalogen *laborationer* (M:\kurser\da211t\laborationer) genom att skriva:

- `cd kurser` följt av RETURN
- `cd da211t` följt av RETURN
- `cd laborationer` följt av RETURN

eller genom att skriva

- `cd kurser\da211t\laborationer` följt av RETURN



Kompilera Lab2Exercise1.java

Det är nödvändigt att ange var programmet javac.exe finns på datorn innan det ska användas. Denna inställning gör batch-filen setpath.bat. Exekvera filen genom att skriva setpath.bat och tryck på RETURN.

```
M:\kurser\da211t\laborationer>setpath.bat
```

Nu ska du **kompilera** Lab2Exercise1.java. Den aktuella katalogen ska vara *laborationer*, dvs. den katalog i vilken du placerat Lab2Exercise1.java. Kompileringen sker genom att skriva instruktionen:

- `javac Lab2Exercise1.java` följt av RETURN.

```
M:\kurser\da211t\laborationer>javac Lab2Exercise1.java
```

OBS!

Det är viktigt att du skriver källkodsfilens namn korrekt. Java gör t.ex. skillnad på liten och stor bokstav.

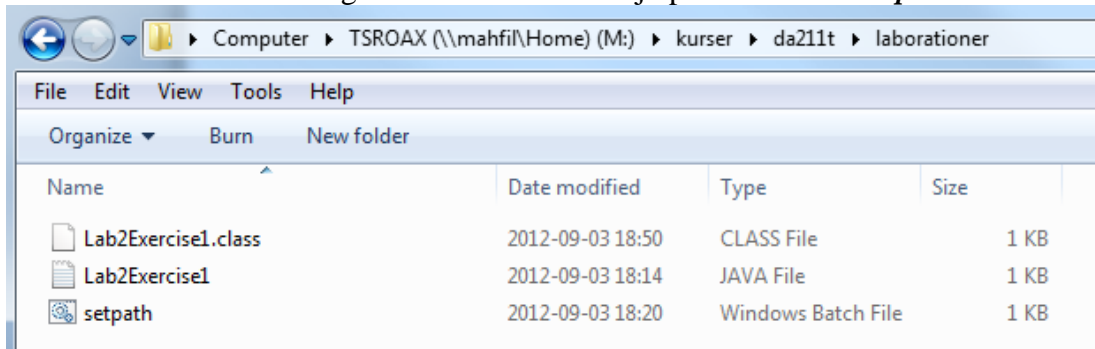
Nu startas programmet *javaw.exe* och filen som ska kompileras är *Lab2Exercise1.java*.

Resultatet av kompileringen är en fil som slutar med ".class", i det här fallet.

Lab2Exercise1.class. Klassen placeras i samma katalog som källkodsfilen. Den nya filen innehåller *bytekod* och kallas oftast för *bytekodsfil* eller *klassfil*.

Kontrollera att *Lab2Exercise1.class* har skapats. Detta kan du göra genom att:

- Studera innehållet i katalogen *laborationer* med hjälp av **Windows Explorer**.



- Studera innehållet i aktuell katalog i **Command Prompt**. Skriv *dir* och tryck på RETURN.

```
M:\kurser\da211t\laborationer>dir
Volume in drive M has no label.
Volume Serial Number is 0401-95FD

Directory of M:\kurser\da211t\laborationer

2012-09-03  18:50    <DIR>          -
2011-10-21  21:26    <DIR>          -
2012-09-03  18:14             966 Lab2Exercise1.java
2012-09-03  18:50             490 Lab2Exercise1.class
2012-09-03  18:20             72 setpath.bat
                3 File(s)              1 528 bytes
                2 Dir(s)  285 684 441 088 bytes free
```

Exekvera programmet

Java kräver ett speciellt program, en virtuell maskin, för att programmet du skrivit ska exekveras. Detta program heter *java.exe* och ingår i JDK.

Den aktuella katalogen ska vara *laborationer* (den katalog i vilken *Lab3Exercise1.class* lagras). Exekvering sker genom att skriva instruktionen:

- *java Lab2Exercise1* följt av RETURN.

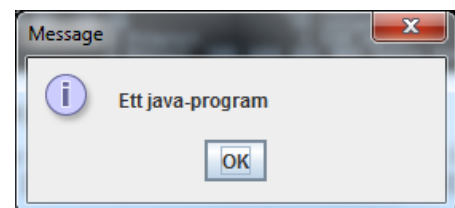
```
M:\kurser\da211t\laborationer>java Lab2Exercise1
```

OBS!

Det är viktigt att du skriver bytekodfilens namn korrekt!. Java gör t.ex. skillnad på liten och stor bokstav.

Om allt är som det ska kommer ett fönster visa sig på skärmen.

När du klickar på OK avslutas programmet. Då är den enda instruktionen i metoden meddelande utförd:



```
JOptionPane.showMessageDialog( null, "Ett java-program" );
```

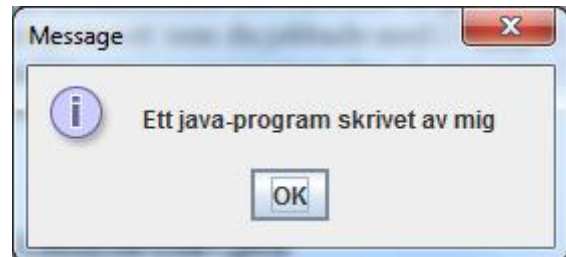
Uppgift 1b

Det är väldigt vanligt att ett program inte fungerar som tänkt vid första provkörningen. Om så är fallet måste man öppna sin källkodsfil och ändra i programmet. Därefter kompilera källkodsfilen på nytt för att slutligen exekvera programmet på nytt.

De steg som man ska ta är alltså samma som i uppgift 1:

1. Öppna källkodsfilen och gör nödvändiga ändringar. Spara sedan ändringarna.
2. Kompilera källkodsfilen. Om kompileringen inte går bra gå tillbaka till punkt 1.
3. Exekvera bytekodsfilen. Om resultatet inte är bra så gå tillbaka till punkt 1.

När du följer nedanstående instruktioner så gör du en mindre ändring i programmet som du jobbade med i Uppgift 1a. Du ändrar instruktionen i message-metoden så att texten "Ett java-program skrivet av mig!" visas i fönstret.



1. Ändra i Lab2Exercise1.java

Öppna Lab2Exercise1.java i *NotePad* om filen inte redan är öppen. Ändra texten i message-metoden till:

```
JOptionPane.showMessageDialog( null, "Ett java-program skrivet av mig" );
```

När du gjort ändringen ska du **spara filen**. Kompilatorn använder ju den sparade versionen.

2. Kompilera Lab2Exercise1.java

Gå till *Kommandotolken* och skriv in kompileringskommandot på nytt:

- `javac Lab2Exercise1.java` följt av *RETURN*.

```
M:\kurser\da211t\laborationer>javac Lab2Exercise1.java
```

3. Exekvera Lab2Exercise1

Skriv in exekveringskommandot på nytt:

- `java Lab2Exercise1` följt av *RETURN*.

```
M:\kurser\da211t\laborationer>java Lab2Exercise1
```

Del 2

Nu ska du använda verktyget Eclipse för att jobba med java-program. Du ska bekanta dig med Eclipse programmeringsmiljö, dvs att med hjälp av Eclipse

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.

Förberedelse

Skapa katalogen *EclipseProjects*. I katalogen M:\kurser\da211t

TSROAX (\\mahfil\Home) (M:) ▶ kurser ▶ da211t ▶ EclipseProjects ▶

Starta Eclipse

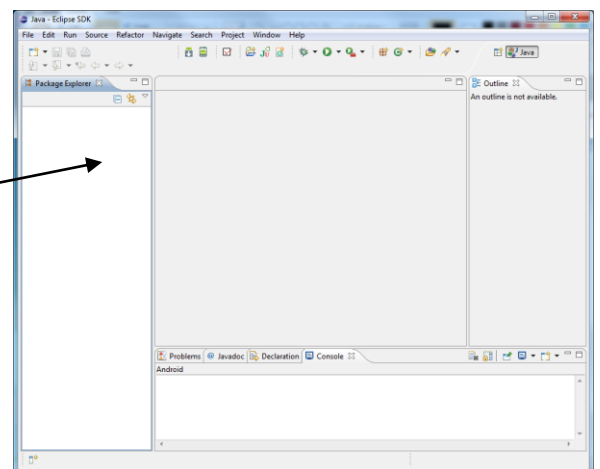
- Klicka på **Start – All Programs – Eclipse Indigo - eclipse**.
- Nu visar sig en dialog, **Workspace Launcher**, i vilken du ska ange den katalog där du vill att Eclipse-projekt (dina program) ska sparas.
Skriv in: *M:\kurser\da211t\EclipseProjects* eller klicka på Browse, leta upp *EclipseProjects*, markera katalogen och klicka på OK.

Efter en liten stund ska ett fönster liknande figuren till höger synas. Stäng välkomstfönstret med X efter Welcome (sitter på fliken).



Nu visar sig fönstret till höger och det är dags att börja använda Eclipse.

Projekt-fönster



1. Skapa ett nytt projekt

I Eclipse använder man sig av projekt. Projektet hjälper till att hålla ordning på filer, paket och mycket annat som kan ingå i utvecklingen av program.

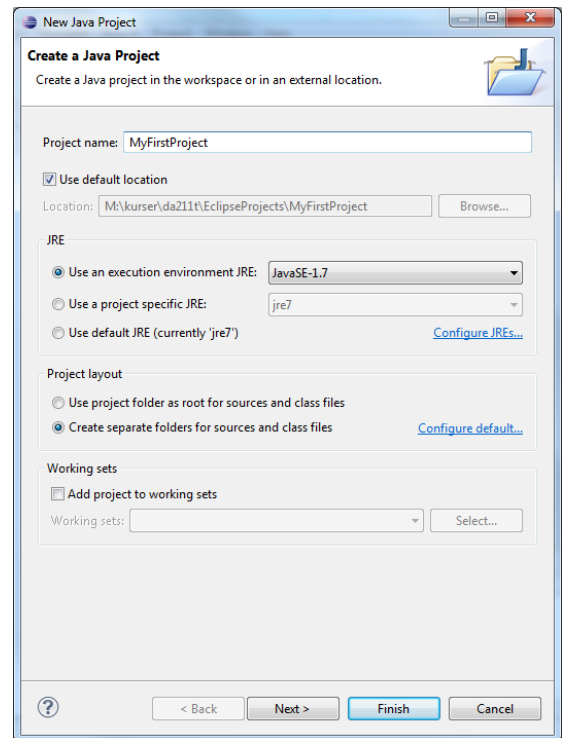
- Välj **File – New – Java Project**

Nu visas ett dialog-fönster - **New Java Project**.

Överst ska du ge projektet ett namn. Kalla det för **MyFirstProject**.

Övriga inställningar kan du låta vara oförändrade.

Klicka på **Finish**.

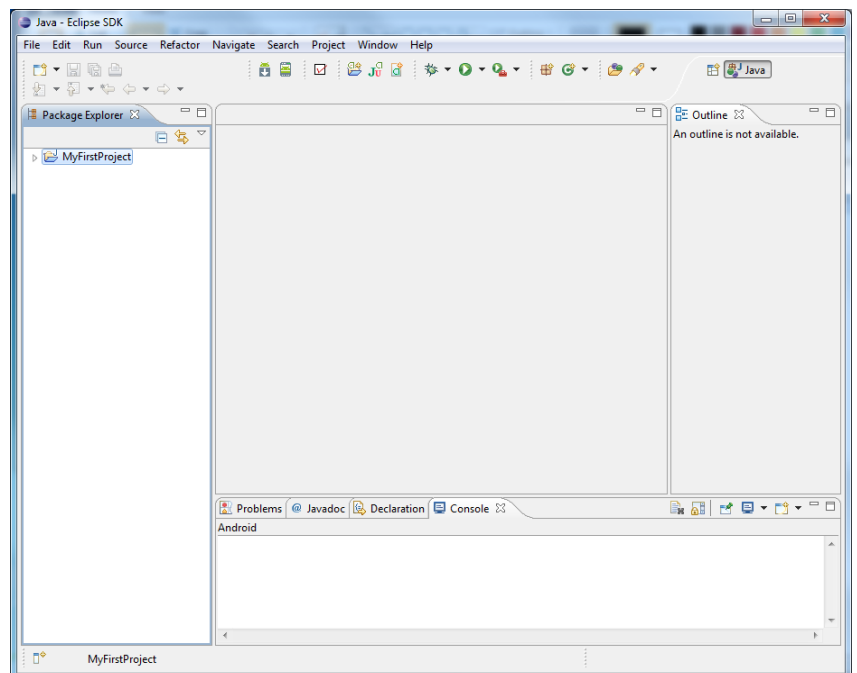
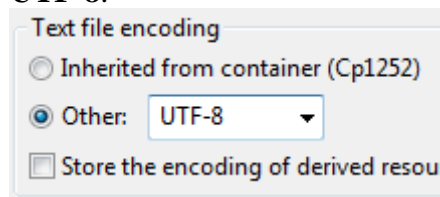


Nu kommer fönstret att se ut så här:

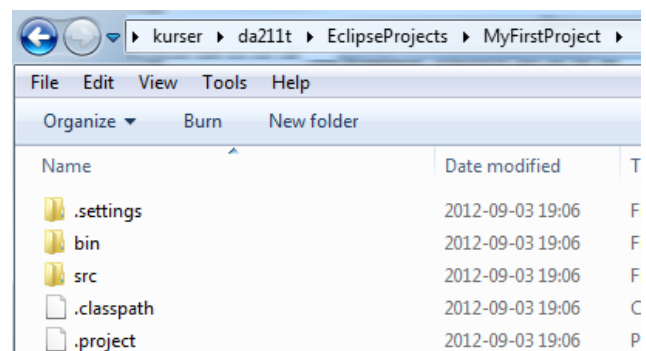
Om du tittar i Projektfönstret så ser du projektet **MyFirstProject**.

Högerklicka **MyFirstProject** i projektfönstret och välj **Properties**.

Ändra **Text file encoding** till **UTF-8**.



I **Windows Explorer** kan du se att ett antal filer och kataloger skapats, bl.a. projekt-katalogen *MyFirstProject* och underkatalogerna *src* och *bin*.



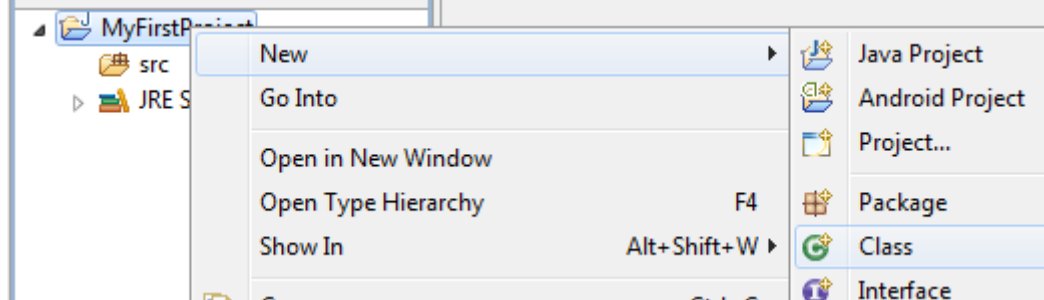
2. Skriva in källkod

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

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

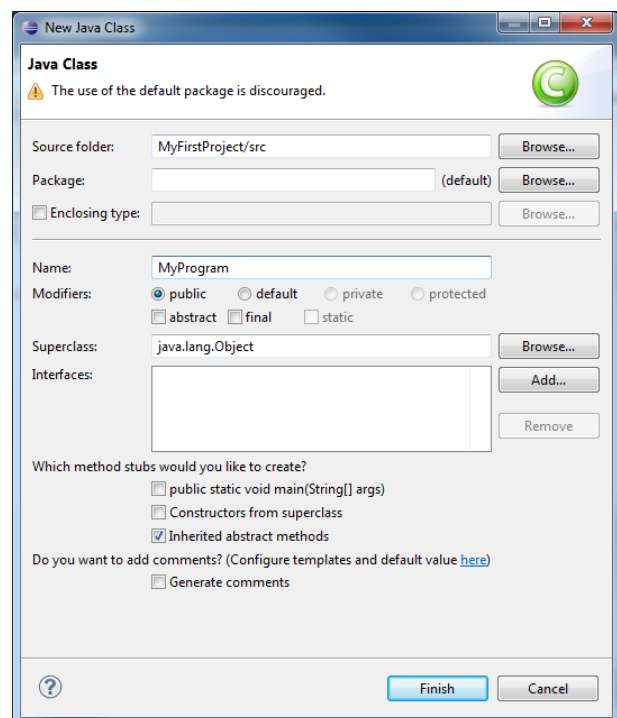
- Högerklicka **MyFirstProject** eller **src** i projektfönstret. Välj sedan **New – Class** i popup-menyn.



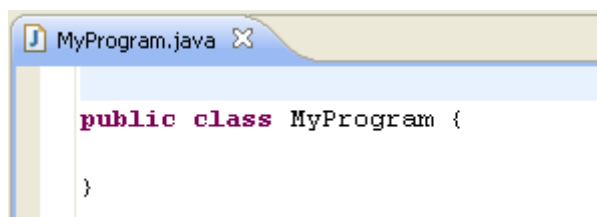
I dialogfönstret **New Java Class** ska du ange lite information, bl.a

- om klassen ska vara i något paket (package). Lämna tomt denna gång.
- klassens namn (ska alltid börja med stor bokstav).

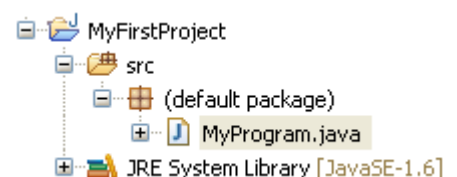
Ange att klassens namn ska vara **MyProgram** och klicka sedan på **Finish**.



Nu skapas en källkodsfil med skalet till klassen **MyProgram**.



Om du tittar i projektkatalogen (*M:\kurser\da211t\EclipseProjects\MyFirstProject*) så finner du filen *MyProgram.java* i *src*-katalogen. *MyProgram.java* är en vanlig textfil i vilken du kommer att lägga till kommentarer och kod.

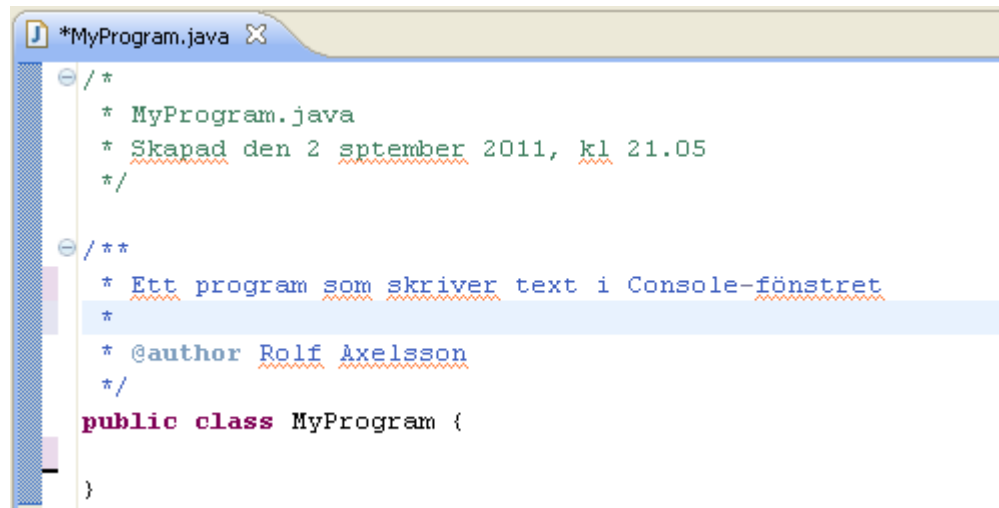


I Eclipse ser du källkodsfilen om du öppnar **MyFirstProject**, sedan **src** och slutligen (**default package**).

Skriva in källkoden

Du ska göra vissa tillägg i MyProgram.java:

- Lägg till kommentarer ovanför klassen. Kommentarer är dokumentation av java-programmet till för dig själv och andra som jobbar med programmet. Kommentarer tillhör inte programmet och påverkar inte programkörningen.



```
/*
 * MyProgram.java
 * Skapad den 2 september 2011, kl 21.05
 */

/**
 * Ett program som skriver text i Console-fönstret
 *
 * @author Rolf Axelsson
 */
public class MyProgram {
}
```

Den översta kommentaren är en s.k. flerradskommentar. En sådan kan du göra i Eclipse genom att skriva `/*` på en tom rad och sedan trycka på ENTER. Nu genereras automatiskt raderna

```
/*
 *
 */
```

och du kan börja skriva kommentaren. I Eclipse är flerradskommentarer gröna till färgen.

Flerradskommentaren

- börjar med (startmarkering) `/*`
- avslutas med (slutmarkering) `*/`
- allt mellan startmarkeringen och slutmarkeringen är kommentarer.

Kommentaren direkt ovanför klassen är en javadoc-kommentar. En sådan kan du göra i Eclipse genom att skriva `/**` på en tom rad och sedan trycka på ENTER. Nu genereras automatiskt raderna

```
/**
 *
 */
```

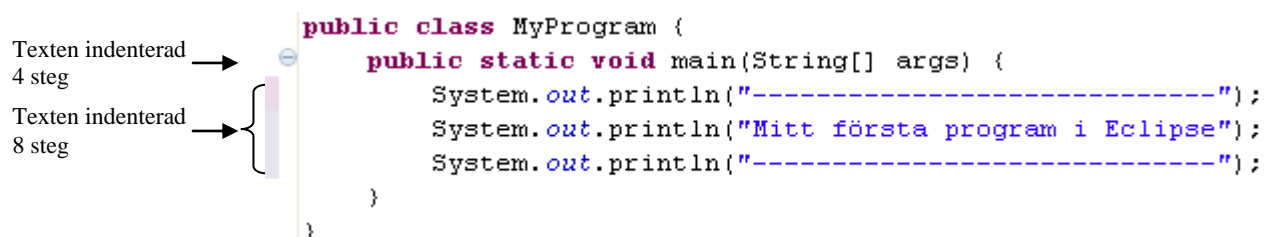
I Eclipse är javadoc-kommentarer blå till färgen. En javadoc-kommentar

- börjar med `/**`
- avslutas med `*/`
- allt mellan startmarkeringen och slutmarkeringen är kommentarer.

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

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

- Lägg till metoden **main** så att klassen MyProgram ser ut så här:



```
public class MyProgram {
    public static void main(String[] args) {
        System.out.println("-----");
        System.out.println("Mitt första program i Eclipse");
        System.out.println("-----");
    }
}
```

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 `main` och instruktionerna som tillhör `main` (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 Eclipse 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 klicka på disketten eller genom att välja **File – Save**.

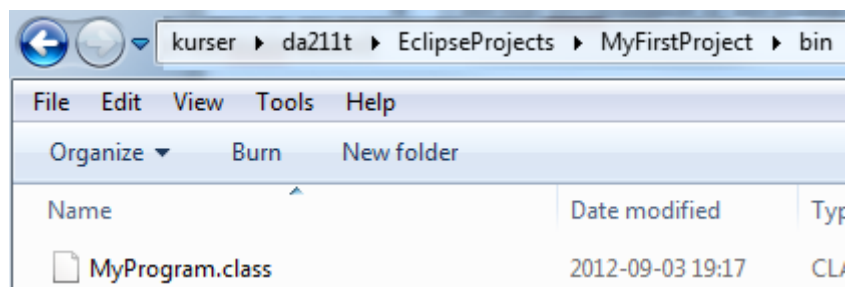
Ett par kommentarer till programmet:

- Klassens namn är alltid samma som filens namn (utom suffixet `.java`). I ovanstående exempel heter klassen `MyProgram` och filen `MyProgram.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 **main** som exekveras. I java startar alltid exekveringen i en `main`-metod.

3. Kompilera programmet

1. För att erhålla ett körbart program måste *källkoden* du skrivit översättas till *byte-kod*. Man säger att källkoden måste kompileras. I Del 1 körde du programmet **javac.exe** för att kompilera källkodsfilen.
2. Sedan exekverade du bytekodsfilen genom att köra programmet **java.exe**.

När du arbetar med **Eclipse** så kompilerar du källkoden varje gång du sparar källkodsfilen. Om du tittar i projektkatalogen så hittar du katalogen `bin`. I katalogen `bin` finner du filen `MyProgram.class`, dvs. filen med bytekod.



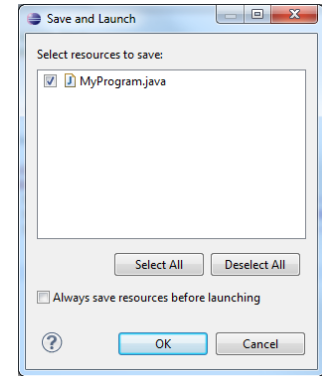
4. Köra programmet

När källkodsfilen är utan syntax-fel är det dags att exekvera (köra) programmet.

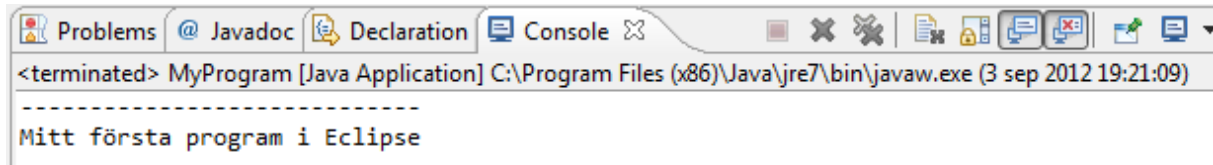
I Eclipse kör du ett program genom något av nedanstående alternativ

- Högerklicka på **MyProgram.java** i projekt-fönstret och välj **Run As – 1 Java Application**.
- Högerklicka på **MyProgram.java** i editor-fönstret och välj **Run As – 1 Java Application**.

Om den eller de källkodsfiler som ingår i programmet sparats sedan de senast ändrades så startar exekveringen direkt. Men om en eller flera av källkodsfilerna inte sparats så visar sig en dialog. Nu kan du välja om filerna ska sparas eller inte före exekveringen. De filer som du markerar kommer att kompileras före exekveringen medan de du inte markerar kommer att använda den senast kompillerade class-filen.

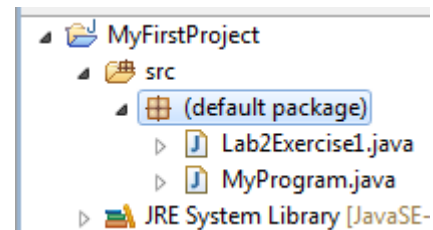


Resultatet av körningen är några utskrifter i Console-fönstret.



Placera en befintlig java-fil i ett Eclipse-projekt.

Navigera till katalogen *M:\kurser\da211t\laborationer* i **Windows Explorer** och kopiera *Lab2Exercise1.java* (markera filen och tryck på CTRL-C). Klicka sedan på Eclipse, markera default package och klistra in (CTRL-V)

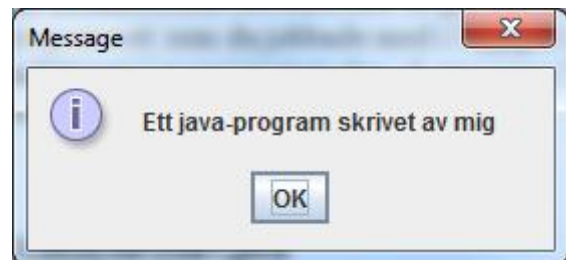


Om du klistrar in *Lab2Exercise1.java* direkt i src-katalogen via Windows Explorer så syns inte filen i Eclipse. Om du gjort detta ska du markera **src** och trycka på **F5 (Refresh)**. Nu ska filen dyka upp under **src**.

Dubbelklicka *Lab2Uppgift1.java* i projektfönstret. Nu öppnas textfilen i editor-fönstret. Tyvärr visas tecknen å, ä och ö på fel sätt. Det beror på att *Lab3Exercise1.java* sparades i Anteckningar med teckenkodningen ANSI medan Eclipse-projektet använder teckenkodningen UTF-8. Den senare teckenkodningen är dock säkrare när man byter filer mellan olika operativsystem. Därför använder vi den på kursen.

Rätta till felaktiga tecken så filen ser bra ut igen.

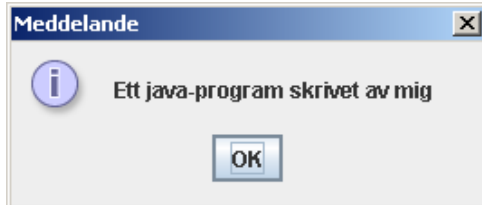
Exekvera filen genom att högerklicka på *Lab2Uppgift1.java* i projektfönstret och välja **Run As – 1 Java Application**. Du ska naturligtvis få samma körresultat som tidigare på laborationen.



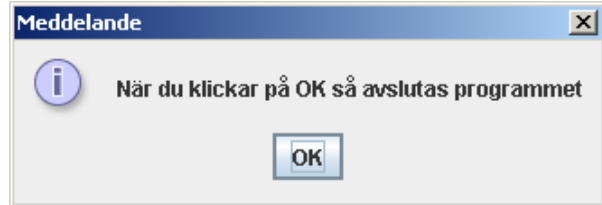
Uppgift 1c

I denna uppgift ska du modifiera **message**-metoden i klassen **Lab2Exercise1** så att två dialogfönster visas efter varandra.

Först:



Sedan:



För att programmet ska visa två dialogfönster efter varandra krävs två instruktioner i **message**-metoden. Först en för det första fönstret, sedan en för det andra fönstret.

```
JOptionPane.showMessageDialog( null, "Ett java-program skrivet av mig" );
JOptionPane.showMessageDialog( null, "När du klickar på OK så avslutas programmet" );
```

Exekvera programmet när du lagt till den sista instruktionen.

Nu ska du avsluta uppgiften med att uppdatera kommentarerna i klassen. Kommentarer påverkar ju inte exekveringen, men de måste vara korrekta.

- Kommentaren som beskriver metoden **message** ska ändras till:
 - * Metoden visar ett par meddelanden med hjälp av dialogfönster.
- Kommentaren ovanför den första instruktionen ska ändras till:
 - // Ett dialogfönster öppnas med meddelandet "Ett java-program skrivet av mig"
- En ny kommentar, som beskriver den andra instruktionen, ska läggas till precis över den andra instruktionen:
 - // Ett dialogfönster öppnas med meddelandet "När du klickar ..."

När du har gjort ovanstående ändringar ska programmet se ut som figuren nedan. Det viktiga är att du skrivit in den nya instruktionen exakt som den står. Det är också viktigt att de två sista kommentarerna börjar med `//`.

```
MyProgram.java  *Lab3Exercise1.java
import javax.swing.JOptionPane;

/**
 * Programmet visar ett par meddelanden med hjälp av dialogfönster.
 *
 * @author Rolf Axelsson
 * @version 1.0
 */
public class Lab3Exercise1 {

    /**
     * Metoden visar ett par meddelanden med hjälp av dialogfönster.
     */
    public void message() {
        // Ett dialogfönster öppnas med meddelandet "Ett java-program skrivet av mig"
        JOptionPane.showMessageDialog( null, "Ett java-program skrivet av mig" );
        // Ett dialogfönster öppnas med meddelandet "När du klickar på OK så avslutas programmet"
        JOptionPane.showMessageDialog( null, "När du klickar på OK så avslutas programmet" );
    }

    /**
     * Instruktionerna i main-metoden utförs då programmet exekveras.
     * main-metoden har endast två instruktioner.
     * - den första instruktionen skapar ett objekt av typen Lab3Exercise1
     * - den andra instruktionen anropar metoden message i det nyss skapade objektet
     */
    public static void main( String[] args ) {
        Lab3Exercise1 prog = new Lab3Exercise1();
        prog.message();
    }
}
```

Uppgift 2a

Hämta textfilen *Lab2Exercise2.java* från hemsidan och placera filen i **src**-katalogen.

- Exekvera *Lab2Exercise2*

Du får ett körresultat liknande figuren till höger. Programmet skriver ut några rader med tecken i Console-fönstret.

```
-    55
      311
      366
-----
```

För att skriva ut en rad med tecken i Console-fönstret kan man använd instruktionen **System.out.println("det som ska skrivas");**

Om du tittar i *Lab2Exercise2.java* ser du 4 instruktioner som ger utskrifter.

Ändra ordningen på instruktionerna så att programmet ger körresultatet till höger.

```
      366
-    55
-----
      311
```

Uppgift 2b

Om du ändrat på korrekt sätt i Uppgift 2a så ser instruktionerna i message-metoden och main-metoden ut så här:

```
public void calculation() {
    System.out.println("    366");           // A
    System.out.println("-    55");           // B
    System.out.println("-----");           // C
    System.out.println("    311");           // D
}

public static void main( String[] args ) {
    Lab2Exercise2 ex2 = new Lab2Exercise2(); // E
    ex2.calculation();                        // F
}
```

Varje instruktion följs av en bokstav som kommentar. I vilken ordning utförs instruktionerna?

Svara med hjälp av bokstäverna där den instruktion som utförs först ska vara först osv.

Ett exempel på ett *felaktigt* svar är: F – E – D – A – B – C

Svaret innebär bl.a. att instruktionen `ex2.calculation();` utförs först och instruktionen `System.out.println("-----");` sist.

Uppgift 2c

Lägg till två instruktioner i main-metoden:

```
public static void main( String[] args ) {
    Lab2Exercise2 ex2 = new Lab2Exercise2(); // E
    ex2.calculation();                        // F
    System.out.println("EN GÅNG TILL!");     // G
    ex2.calculation();                        // H
}
```

Exekvera programmet. Körresultatet blir som figuren till höger. Nu utförs vissa instruktioner flera gånger. Ange med bokstäverna den ordning som instruktionerna exekveras denna gång.

```
      366
-    55
-----
      311
EN GÅNG TILL!
      366
-    55
-----
      311
```

Del 3

I del 3 ska du ett skriva några program. Uppgifterna är hämtade ur Java Foundations, s 55-57. Jag har scannat in dem eftersom alla inte har hunnit skaffa boken.

Uppgift 3

```
PP 1.1    Enter, compile, and run the following application:

public class Test
{
    public static void main (String[] args)
    {
        System.out.println ("An Emergency Broadcast");
    }
}
```

Börja med att skapa klassen Test. På samma sätt som du skapade klassen MyProgram tidigare på laborationen. Skriv sedan main-metoden i klassen.

Uppgift 4

PP 1.2 Introduce the following errors, one at a time, to the program from the programming project 1.1. Record any error messages that the compiler produces. Fix the previous error each time before you introduce a new one. If no error messages are produced, explain why. Try to predict what will happen before you make each change.

- a. change Test to test
- b. change Emergency to emergency
- c. remove the first quotation mark in the string
- d. remove the last quotation mark in the string
- e. change main to man
- f. change println to bogus
- g. remove the semicolon at the end of the println statement
- h. remove the last brace in the program

Uppgift 5

PP 1.3 Write an application that prints, on separate lines, your name, your birthday, your hobbies, your favorite book, and your favorite movie. Label each piece of information in the output.

Uppgift 6

PP 1.4 Write an application that prints the phrase Knowledge is Power:

- a. on one line
- b. on three lines, one word per line, with the words centered relative to each other
- c. inside a box made up of the characters = and |

Uppgift 7

PP 1.5 Write an application that prints a list of four or five Web sites that you enjoy. Print both the site name and the URL.

Uppgift 8

PP 1.6 Write an application that prints the first few verses of a song (your choice). Label the chorus.

Uppgift 9

PP 1.7 Write an application that prints the following diamond shape. Don't print any unneeded characters. (That is, don't make any character string longer than it has to be.)

```
  *
 ***
*****
*****
*****
*****
*****
***
  *
```

Uppgift 10

PP 1.8 Write an application that displays your initials in large block letters. Make each large letter out of the corresponding regular character. For example:

JJJJJJJJJJJJJJ	AAAAA	LLLL
JJJJJJJJJJJJJJ	AAAAA	LLLL
JJJJ	AAA AAA	LLLL
JJJJ	AAA AAA	LLLL
JJJJ	AAAAA	LLLL
J JJJJ	AAAAA	LLLL
JJ JJJJ	AAA AAA	LLLL
JJJJJJJJJJ	AAA AAA	LLLLLLLLLLLLLL
JJJJJJJJ	AAA AAA	LLLLLLLLLLLLLL

Lösningar

Uppgift 2b

E-F-A-B-C-D

Uppgift 2c

E-F-A-B-C-D-G-H-A-B-C-D

Uppgift 3 - 10

Se filer i DA211TL2SolutionsHT12.zip