

2 Eclipse – en handledning

Inledning

Här beskrivs Eclipse, den programutvecklingsmiljö som utnyttjas i programmeringskurserna. Mera information finns på:

http://www.eclipse.org	Eclipse hemsida. Enorma mängder information.
http://help.eclipse.org	Hjälpssidor för Eclipse, senaste versionen. Dessa hjälpsidor finns också lokalt i Eclipse-systemet, under Help Contents i Help-menyn.
http://www.eclipse.org/downloads/	Nedladdning av Eclipse för olika datorer. Välj "Eclipse IDE for Java Developers".

Det finns också många handledningar som är betydligt utförligare än denna, till exempel på www.vogella.com/articles/Eclipse/article.html.

Att utveckla Javaprogram

Ett Javaprogram består av en eller flera klasser som lagras i filer med tillägget `.java`. Dessa filer innehåller "vanlig text" (bokstäver, siffror och andra tecken). Innan man kan exekvera (köra) programmet måste filerna kompileras (översättas) till ett annat format (bytekod) som datorn "förstår". Bytekoden lagras i filer med tillägget `.class`. Översättningen görs av en kompilator som heter `javac`, och programmet körs av en Javatolk som heter `java`.

Allt detta kan man hantera manuellt. Då editerar man `.java`-filerna med Emacs eller en annan texteditor, kompilerar dem med `javac` och kör programmet med `java`. Det kan se ut så här:

<code>% emacs HelloWorld.java &</code>	Starta en editor, skriv följande programtext:
	<pre>public class HelloWorld { public static void main(String[] args) { System.out.println("Hello, world!"); } }</pre>
<code>% javac HelloWorld.java</code>	Kompilera programmet
<code>% ls</code>	Lista innehållet i katalogen
<code>HelloWorld.class HelloWorld.java</code>	Två filer: kompilerad kod och källkod
<code>% java HelloWorld</code>	Kör programmet
<code>Hello, world!</code>	
<code>%</code>	

Det är inget fel med att utveckla program på detta sätt, men det har en del nackdelar: stora program består av många klasser och det kan vara svårt att hålla reda på dem, man måste komma ihåg att kompilera varje fil som man ändrar, när kompilatorn hittar fel i ett program måste man i editorn hitta motsvarande rad i filen för att kunna rätta till felet, om man vill följa exekveringen av ett program måste man ha en separat "debugger" (felsökningsprogram).

Ett alternativt sätt att arbeta är att använda en integrerad utvecklingsmiljö (Integrated Development Environment, IDE). Det är ett program med ett grafiskt användargränssnitt där editor,

kompiator och debugger är integrerade. Man startar IDE-programmet och hanterar sedan alla moment i programutvecklingen inifrån denna miljö.

Programutvecklingen består fortfarande av editering, kompilering och programkörning, men det blir enklare att utföra dessa moment — de utförs av IDE-programmet genom att man klickar på knappar eller väljer ur menyer.

Det finns många olika IDE-program, mer eller mindre avancerade (Eclipse, IDEA, JBuilder, Netbeans, JCreator, ...). I programmeringskurserna utnyttjas Eclipse.

Eclipse

Översikt

Eclipse är ett avancerat IDE-program som erbjuder all funktionalitet som behövs för utveckling av Javaprogram. Det finns också ett stort antal "plugins" som utökar Eclipse med nya möjligheter (utveckling av webbprogram, databasprogram, diagramritning, andra programspråk än Java, osv). Programmet är gratis.

Eftersom Eclipse har så många möjligheter är det inte helt enkelt att lära sig att använda fullt ut. Man måste sätta sig in i en hel del terminologi och lära sig hitta bland alla menyer och kommandon. Men grunderna är inte alltför komplicerade, och de beskrivs här.

Några termer måste man känna till:

Projekt	(project) Ett antal <i>.java</i> - och <i>.class</i> -filer som hör ihop, till exempel alla filer som behövs för en laboration eller en inlämningsuppgift. Motsvaras av en katalog i filsystemet.
Arbetsområde	(workspace) Det område där man sparar de projekt som man skapar. Motsvaras av en katalog i filsystemet (projekten är underkataloger till denna katalog).
Perspektiv	(perspective) Utseendet hos Eclipsefönstret. Ett perspektiv har ett antal fönster och knappar för kommandon.

Det finns massor av möjligheter att ställa in Eclipse så att det fungerar på olika sätt (Preferences i Window-menyn). Varje inställningssida har en knapp Restore Defaults som ställer tillbaka alla inställningar till de ursprungliga.

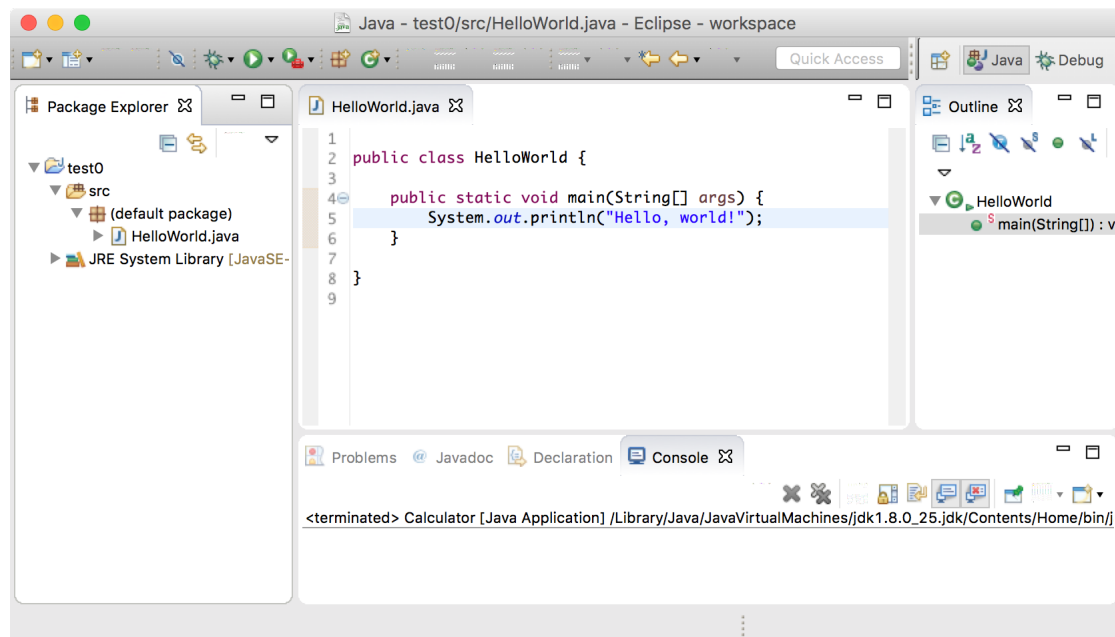
Användning av Eclipse

Vi förutsätter i detta avsnitt att Eclipse har installerats (hur man gör skiljer sig något mellan olika datorer; se separata anvisningar). Vi förutsätter också att vi har skapat ett arbetsområde med ett projekt *test0* som innehåller en fil *HelloWorld.java* och en kompilerad fil *HelloWorld.class*. Hur man skapar projekt och filer visar vi i nästa avsnitt. När man startar Eclipse får man ett fönster som visas i figur 1 (under Linux, det kan se något annorlunda ut på andra datorer):

Om det inte står Java i fönstrets titelrad så är fel perspektiv aktiverat. Byt då perspektiv med Open Perspective i Window-menyn. I projektvyn (längst till vänster under Package Explorer) ser man de projekt man har skapat. När man klickar på pilen vid ett projekt så öppnas projektet så att man ser innehållet. *.java*-filerna finns i katalogen *src* under (*default package*), *.class*-filerna finns i en katalog *bin* som inte visas i projektvyn.

Området i mitten är en texteditor. När man dubbelklickar på en fil i ett projekt så laddas filen in i editorn. Man kan editera flera filer samtidigt i olika flikar.

I editorn arbetar man som i alla editorer: man skriver och tar bort text som vanligt, markerar text genom att dra med musen, klipper ut och klistrar in med CONTROL-X och CONTROL-V, och så vidare. Eclipse-editorn känner till hur Javaprogram ska formateras, så den gör automatiskt indragningar där de ska vara, stoppar in en högerparentes av rätt slag när man skriver en




Figur 1: Eclipse-fönster, Java

vänsterparentes, och så vidare. Man kan till och med formatera om en hel fil så att programmet följer kodkonventionerna; det gör man med kommandot Format i Source-menyn.

Eclipse-editorn kontrollerar också programmet och kompilerar det under tiden man skriver. Om man skriver något som bryter mot Javas regler så markeras rader som innehåller fel med kryss till vänster om raden. Man får en förklaring av felet genom att hålla musmarkören över krysset.

Området under texteditorn har flera flikar. I Console visas utskrifter från program som körs, och där skriver man också indata till programmet. I Problems visas felmeddelanden som finns i filer som man sparar.

Man kör ett program genom att högerklicka på den fil som innehåller main-metoden i projekt vyn och välja Run As > Java Application. (Eller genom att välja Run As > Java Application i Run-menyn eller genom att klicka på Run-ikonen  i verktygsraden.)

Om man råkar stänga till exempel editorfönstret utan att vilja det eller ställt till något annat med Javaperspektivet, så kan man återställa perspektivet till utgångsläget med kommandot Reset Perspective i Window-menyn.

Skapa projekt och klass


Nu visar vi steg för steg hur man skapar ett nytt projekt med en klass och hur man kör programmet. Projektet ska heta *test1* och innehålla en klass *Calculator* med följande utseende:

```
/** Calculator beräknar summan av två tal */
public class Calculator {
    public static void main(String[] args) {
        double nbr1 = 5.5;
        double nbr2 = 11.3;
        double sum = nbr1 + nbr2;
        System.out.println("Summan av talen är " + sum);
    }
}
```

Detta är inte ett vettigt program, eftersom det bara kan beräkna summan av talen 5.5 och 11.3. Om man vill beräkna summan av två andra tal måste man ändra i programmet och kompilera

om det. Något mera användbart skulle programmet bli om det läste in talen från tangentbordet, men vi gör det så enkelt som möjligt nu.

Gör följande:

1. Skapa projektet *test1*. Välj File > New > Java Project, skriv namnet på projektet, klicka på Finish.
2. Skapa en fil *Calculator.java*. Markera projektet *test1* i projektvyn, klicka på New Java Class-ikonen  i verktygsraden, skriv namnet på klassen (Calculator), klicka på Finish.
3. Öppna projektet, *src*-katalogen och (*default package*) genom att klicka på pilarna. Dubbelklicka på *Calculator.java* för att ladda in filen i editorn. Som du ser så har Eclipse redan fyllt i klassnamnet och de parenteser som alltid ska finnas. Komplettera klassen med *main*-metoden som visas ovan.
4. Spara filen.
5. Kör programmet genom att högerklicka på filen och välja Run As > Java Application. Utskriften som görs med `System.out.println` hamnar i konsolfönstret (om utskriften inte blir Summan av talen är 16.8 så har du skrivit fel i programmet).

En del program behöver utnyttja klasser som inte finns i det aktuella projektet. (Egentligen gör alla program det, de använder Javas standardklasser, men dessa är alltid tillgängliga och man behöver inte göra något för att programmen ska komma åt dem.) Man måste för projektet ange var dessa klasser finns. Det gör man genom att högerklicka på projektet och välja Build Path > Configure Build Path.

Sedan finns det två fall:

- Klasserna som ska utnyttjas finns i ett annat projekt i samma arbetsområde. Klicka på Projects > Add... och markera projektet som innehåller klasserna.
- Klasserna som ska utnyttjas finns i en biblioteksfil (*.jar*-fil, Java Archive) i ett annat projekt. Klicka på Libraries > Add JARs... och välj rätt *.jar*-fil. Om *.jar*-filen finns utanför arbetsområdet klickar man i stället på Add External JARs... och letar upp filen.

Hitta fel i program

Fel i program är av två slag: dels "språkfel" som kompilatorn hittar (man glömmer ett semikolon, glömmer att deklarerar en variabel, stavar fel till ett ord och liknande fel), dels "logiska fel" som medför att ett program ger felaktigt resultat när det exekveras.

Som exempel tar vi programmet som beräknar summan av två tal från föregående avsnitt. Satsen som beräknar summan och sparar den i variabeln *sum* ser ut så här:

```
double sum = nbr1 + nbr2;
```

Om vi i stället hade skrivit `dubbel sum = nbr1 + nummer2;` hade vi gjort två språkfel: typen för reella tal heter *double*, inte *dubbel*, och variabeln där värdet av det andra talet finns heter *nbr2*, inte *nummer2*. Kompilatorn markerar alla fel av denna typ, och de är för det mesta enkla att korrigera (det kan vara svårt att tolka felmeddelandena i början, men det lär man sig efterhand).

Logiska fel är svårare att hitta. Antag att vi hade skrivit satsen på följande sätt:

```
double sum = nbr1 - nbr2;
```

Detta är en helt korrekt sats och programmet kan kompileras och köras utan problem, men resultatet blir inte korrekt (man får utskriften `Summan av talen är -5.8`).





Det här programmet är så litet och felet är så uppenbart att man kan hitta felet genom att bara titta på programmet. Men i allmänhet är det inte så enkelt: "riktiga" program är mycket större och felet kan vara mycket mera komplicerat.

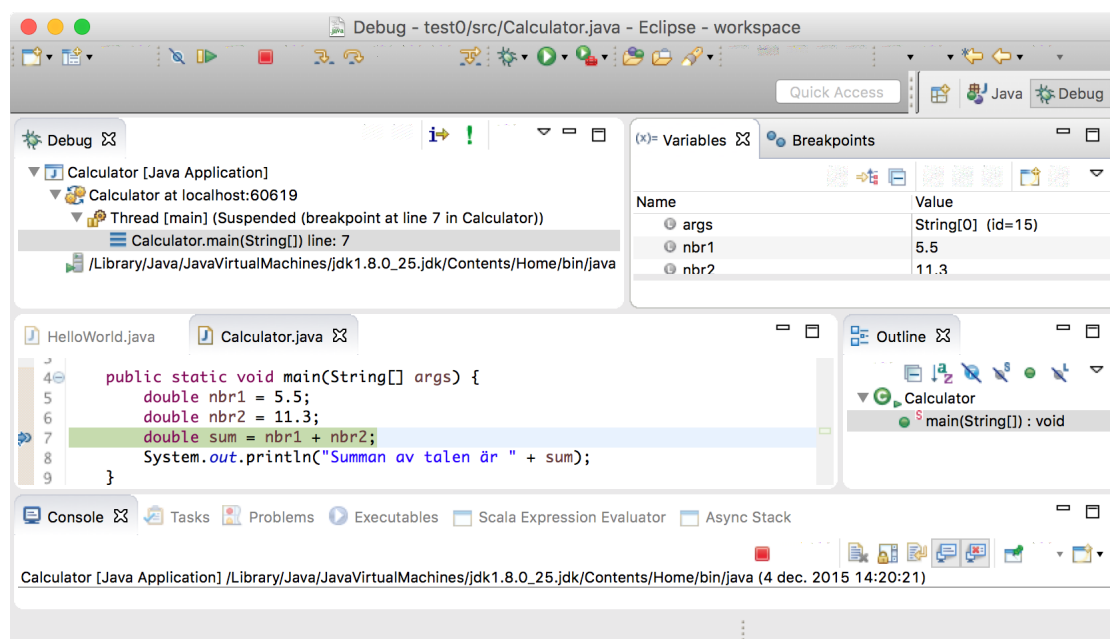
För att hitta logiska fel behöver man skaffa sig information om vad som händer under exekveringen av programmet. Ett sätt att göra det är att lägga in utskrifter av viktiga variabelvärden på lämpliga ställen i programmet (det är ju egentligen det vi gjort i ovanstående program, där vi omedelbart efter summeringen skriver ut summan). Detta sätt är dock svårhanterligt och tidsödande.

Bättre är att utnyttja en *debugger* ("avlusare"). Då får man möjlighet att själv kontrollera exekveringen av programmet: man kan köra programmet rad för rad och man kan under tiden titta på hur variabelvärdena ändras. Man kan också sätta brytpunkter i programmet. När man sedan startar programmet så stoppas exekveringen när den når en brytpunkt.

Man sätter en brytpunkt på en rad genom att dubbelklicka till vänster om raden, och man tar bort en brytpunkt på samma sätt. När man har satt en brytpunkt kan man starta debuggern. Det gör man genom att högerklicka på filen och välja `Debug As > Java Application`. Eclipse byter till debug-perspektivet. Fönstrets nya utseende visas i figur 2.

I mitten finns editorfönstret. Den rad som står i tur att exekveras är markerad med en pil. Ovanför editorfönstret finns en vy med information om det körande programmet. I verktygsraden finns ikoner för kommandon. De viktigaste är:

-  Resume — fortsätt exekveringen till nästa brytpunkt eller till programmets slut.
-  Terminate — avsluta exekveringen av programmet.
-  Step over — exekvera en rad i programmet.
-  Step into — exekvera till nästa rad, gå in i metoder som anropas.



Figur 2: Eclipse-fönster, debug

Importera och exportera filer

När man arbetar med Eclipse har man normalt alla sina filer i projekt i arbetsområdet. Om man har filer någon annanstans kan man importera dem till ett existerande projekt. Det gör man genom att högerklicka på projektet, välja Import... > General > File System och leta upp filerna. Genom att i stället välja Export kan man exportera filer.

Man kan också importera och exportera hela projekt. Det gör man genom att högerklicka i projektvyn och välja Import > General > Archive File respektive genom att högerklicka på ett projekt och välja Export > General > Archive File. Ett arkiverat projekt sparas som en komprimerad .zip-fil.

Att skapa nya projekt (för egna övningar)

Du kan skapa egna projekt i Eclipse, där du kan öva på att skriva egna program. Om du lägger dina program i egna projekt hamnar de i en egen katalog i Eclipse, precis som laborationerna.

Hur du skapar ett nytt projekt ser du i avsnittet "Skapa projekt och klass" (som börjar på s. 11). Här beskrivs hur du gör kursens klasser (SimpleWindow, Square, m. fl.) tillgängliga för ditt nya projekt. Om du inte gör detta så kommer Eclipse att ge felmeddelanden då du försöker använda dessa (t. ex. i import-satser).

- Högerklicka på ditt nya projekt i vänsterspalten i Eclipse.
- En lång meny dyker upp, där du väljer Build Path > Configure Build Path... .
- I rutan som dyker upp väljer du fliken Libraries.
- Klicka på Add JARs...
- I listan över projekt öppnar du cs_pt och väljer filen cs_pt.jar. Klicka OK för att lämna projektlistan, därefter OK för att lämna Build Path-inställningarna.

Nu ska ditt nya projekt kunna utnyttja kursens klasser, precis som laborationerna.