Laboration 1 – introduktion

Mål: Under denna laboration ska du lära dig vad ett datorprogram är och se exempel på enkla program. Du ska också lära dig att editera, kompilera och exekvera enkla Java-program med hjälp av programutvecklingsverktyget Eclipse. Du ska också prova att använda Eclipse debugger.

Förberedelseuppgifter

- Läs i en av läroböckerna: **antingen** Downey & Mayfield, kapitel 1–2, **eller** Holm, kapitel 1.
- Läs avsnittet Bakgrund.
- Läs igenom anvisningarna om Eclipse (finns i kurskompendiet).

Bakgrund

Ett datorprogram är ett antal rader text som beskriver lösningen till ett problem. Vi börjar med att titta på ett mycket enkelt problem: att från datorns tangentbord läsa in två tal och beräkna och skriva ut summan av talen. Lösningen kan se ut så här i Java:

```
1  import java.util.Scanner;
2
3  public class Calculator {
4      public static void main(String[] args) {
5          System.out.println("Skriv två tal");
6          Scanner scan = new Scanner(System.in);
7          double nbr1 = scan.nextDouble();
8          double nbr2 = scan.nextDouble();
9          double sum = nbr1 + nbr2;
10         System.out.println("Summan av talen är " + sum);
11     }
12 }
```

Detta är nästan samma program som beskrivs i Eclipse-handledningen, men talen som summeras läses från tangentbordet. Du behöver inte förstå detaljerna i programmet nu, men vi vill redan nu visa ett program som gör någonting intressantare än bara skriver ut ”Hello, world!”. Allt som behövs för att du ska förstå programmet kommer vi att gå igenom under de första veckorna av kursen. Förklaring av de viktigaste delarna av programmet:

- Rad 1: en inläsningsklass `Scanner` importeras från ett ”paket” med namnet `java.util`. Detta paket och många andra ingår i Javas standardbibliotek.
- Rad 3: `public class Calculator` talar om att programmet, klassen, heter `Calculator`.
- Rad 4: `public static void main(String[] args)` talar om att det som vi skriver är ett ”huvudprogram”. Varje Javaprogram måste innehålla en klass med en `main`-metod. Exekveringen börjar och slutar i `main`-metoden.
- Rad 5: texten `Skriv två tal` skrivs ut på skärmen (`println` betyder ”print line”). När man använder Eclipse så hamnar utskrifter i konsolfönstret.
- Rad 6: inläsningsobjektet `scan` skapas. Vi kommer senare att gå igenom ordentligt vad detta betyder — nu räcker det att du vet att man alltid måste skapa ett sådant objekt när man vill läsa från tangentbordet.
- Rad 7: en variabel `nbr1` som kan innehålla ett reellt tal (`double`-tal) deklareras. Det betyder att man skapar plats för variabeln i datorns minne. Man tilldelar också `nbr1` ett talvärde som man läser in från tangentbordet med `nextDouble()`.
- Rad 8: samma sak med variabeln `nbr2`.

- Rad 9: variabeln `sum` deklareras. Talen `nbr1` och `nbr2` adderas och summan lagras i `sum`.
- Rad 10: resultatet (innehållet i variabeln `sum`) skrivs ut.
- Rad 11: slut på `main`-metoden.
- Rad 12: slut på klassen.

Uppgifter

1. Logga in på datorn.

2. Under laborationerna kommer du att använda en hel del färdigskrivna eller nästan färdigskrivna program. Nu ska du skapa ett Eclipse-arbetsområde (workspace) som innehåller alla dessa filer.

   1. Öppna ett terminalfönster.
   2. Skriv följande kommandon:[1]

      ```
      wget cs.lth.se/edaa55-ht2/ws
      unzip ws
      ```

      Det första kommandot laddar ner en komprimerad zip-fil med ditt workspace. Det andra kommandot packar upp filens innehåll.
   3. Nu har du fått en katalog *pt-workspace* i din hemkatalog. Katalogen innehåller underkatalogerna *cs_pt*, *Lab01*, *Lab02*, *Lab03*, ... Kontrollera innehållet i *pt-workspace* med följande kommando:

      ```
      ls pt-workspace
      ```

      I *pt-workspace* finns också en katalog *.metadata*. När du senare startar Eclipse skapas i din hemkatalog en katalog *.eclipse*. Dessa kataloger syns inte när du gör `ls`, eftersom deras namn inleds med punkt. Rör *inte* dessa kataloger — de används av Eclipse för att spara viktig information, och om de inte finns eller har fel innehåll så går det inte att starta Eclipse.
   4. Den nedladdade filen behövs inte mera. Tag bort den med följande kommando:

      ```
      rm ws
      ```

3. Nu ska du starta Eclipse. Ge följande kommando:

   ```
   eclipse &
   ```

   Efter en stund får du en dialogruta där Eclipse frågar efter vilket arbetsområde som du vill använda. Ändra förslaget `workspace` till `pt-workspace`. Om du, efter att ha valt katalog och klickat ok, får en varningsruta om att workspacet byggdes med en annan version av Eclipse så är det ingen fara. Det är bara att gå vidare förbi detta.

   `&`-tecknet betyder att Eclipse ska startas ”i bakgrunden”, alltså som ett fristående program. Det medför att man inte låser terminalfönstret utan kan arbeta med andra saker i det samtidigt som Eclipse kör. Alternativt kan du starta Eclipse genom att välja Eclipse (den senaste versionen) från Gnome-menyn Applications > Programming.

[1] Dessa kommandon gäller för LTH:s Linuxdatorer. Om du använder en egen Windows-dator eller Mac finns instruktioner på kurshemsidan för hur du kommer igång. Om du ändå vill ladda hem workspacet på detta sätt med en Mac byter du ut det första kommandot (`wget`) mot `curl -OL cs.lth.se/edaa55-ht2/ws`

I Eclipse-fönstret finns i projektvyn (”Package Explorer”, längst till vänster) projekten *cs_pt*, *Lab01*, *Lab02*, *Lab03*, ... — det är de projektkataloger som finns i *pt-workspace*. Filerna med namn som börjar på *Lab* innehåller färdigskrivna program som utnyttjas under laborationerna. *cs_pt* innehåller en biblioteksfil (*.jar*-fil) med klasser som utnyttjas. I samma projekt finns källkoden (*.java*-filerna) till dessa klasser — de behövs inte för laborationerna, men en del brukar vara intresserade av att titta på dem.

4. Du ska nu ladda in filen *Calculator.java* i en editor så att du kan ändra den. Man måste klicka en hel del för att öppna en fil:
   a) Öppna projektet *Lab01* genom att klicka på plustecknet (eller pilen) bredvid projektet.
   b) Öppna katalogen *src* genom att klicka på plustecknet.
   c) Öppna paketet *(default package)* genom att klicka på plustecknet.
   d) Öppna filen *Calculator.java* genom att dubbelklicka på filnamnet. Filen öppnas i en editorflik.

5. Kör programmet: markera *Calculator.java* i projektvyn, högerklicka och välj Run As > Java Application. I konsolfönstret skrivs texten `Skriv två tal`. Klicka i konsolfönstret, skriv två tal och tryck på RETURN. Observera: när man skriver reella tal ska man *vid inläsning* använda decimalkomma. När man skriver reella tal i programkod använder man decimalpunkt.

   Man kan köra det senaste programmet en gång till genom att klicka på Run-ikonen i verktygsraden.

6. Ändra `main`-metoden i klassen `Calculator` så att fyra rader skrivs ut: talens summa, skillnad, produkt och kvot. Exempel på utskrift när talen 24 och 10 har lästs in:

   ```
   Summan av talen är 34.0
   Skillnaden mellan talen är 14.0
   Produkten av talen är 240.0
   Kvoten mellan talen är 2.4
   ```

   Du ska alltså efter utskriften av summan lägga in rader där talens skillnad, produkt och kvot beräknas och skrivs ut. Subtraktion anger man med tecknet `-`, multiplikation med `*`, division med `/`.

   Under tiden du skriver så kommer du att märka att Eclipse hela tiden kontrollerar så att allt du skrivit är korrekt. Fel markeras med kryss till vänster om raden. Om du håller musmarkören över ett kryss så visas en förklaring av felet. Om man sparar en fil som innehåller fel så visas felmeddelandena också i Problem-fliken längst ner.

7. Spara filen. Filen kompileras automatiskt när den sparas.

   När *.java*-filen kompileras skapas en ny fil med samma namn som klassen men med tillägget *.class*. Denna fil innehåller programmet översatt till byte-kod och används när programmet exekveras.

   Man ser inte `.class`-filerna inuti Eclipse, men de lagras i arbetsområdet precis som `.java`-filerna. *.java*-filerna finns i en katalog *src* under respektive projektkatalog, *.class*-filerna finns i en katalog *bin*.

8. Kör programmet och kontrollera att utskriften är korrekt. Rätta programmet om den inte är det.

9. Du ska nu använda Eclipse debugger för att följa exekveringen av programmet. Normalt använder man debuggern för att hitta fel i program, men här är avsikten bara att du ska se hur programmet exekverar rad för rad och att du ska bekanta dig med debuggerkommandona.

   - Sätt en brytpunkt på den första raden i `main`-metoden. (Läs avsnittet ”Hitta fel i program” i Eclipse-handledningen om du inte vet hur man gör.) Kör programmet under debuggern (Debug As > Java Application).
   - Svara ja på frågan om du vill byta till Debug-perspektivet.
   - Klicka på Step Over-ikonen några gånger. Notera att den rad i programmet som ska exekveras markeras i editorfönstret och att variabler som deklareras dyker upp i variabelvyn till höger. Variablernas aktuella värden visas i ett av eclipse-fönsterna, se till att du ser detta och kan identifiera hur variablerna får sina värden under programmets exekvering.
   - Sätt en brytpunkt på raden där du skriver kvoten mellan talen.
   - Klicka på Resume-ikonen för att köra programmet fram till brytpunkten.
   - Klicka på Resume igen för att köra programmet till slut.

   Byt tillbaka till Java-perspektivet genom att klicka på knappen Java längst till höger i verktygsfältet.

10. Följande program använder den färdiga klassen `SimpleWindow`:

```
import se.lth.cs.pt.window.SimpleWindow;

public class SimpleWindowExample {
    public static void main(String[] args) {
        SimpleWindow w = new SimpleWindow(500, 500, "Drawing Window");
        w.moveTo(100, 100);
        w.lineTo(150, 100);
    }
}
```

    Specifikationen av klassen `SimpleWindow` finns i appendix C i läroboken, eller online via länk från kurshemsidan. Förklaring av de viktigaste delarna av programmet:

    - På den första raden importeras den färdiga klassen SimpleWindow.
    - Raderna som börjar med `public` och de två sista raderna med `}` är till för att uppfylla Javas krav på hur ett program ska se ut.
    - På nästa rad deklareras en referensvariabel med namnet `w` och typen `SimpleWindow`. Därefter skapas ett `SimpleWindow`-objekt med storleken 500×500 pixlar och med titeln ”Drawing Window”. Referensvariabeln `w` tilldelas det nya fönsterobjektet.
    - Sedan flyttas fönstrets ”penna” till punkten 100, 100.
    - Slutligen ritas en linje.

    Programmet finns inte i arbetsområdet, utan du ska skriva det från början. Skapa en fil *SimpleWindowExample.java*:

    a) Markera projektet *Lab01* i projektvyn,

    b) Klicka på New Java Class-ikonen i verktygsraden.

    c) Skriv namnet på klassen (`SimpleWindowExample`).

    d) Klicka på Finish.

Dubbelklicka på *SimpleWindowExample.java* för att ladda in filen i editorn (om detta inte redan gjorts automatiskt). Som du ser har Eclipse redan fyllt i klassnamnet och de parenteser som alltid ska finnas. Komplettera klassen med `main`-metoden som visas ovan.

Spara filen, rätta eventuella fel och spara igen. Provkör programmet.

11. Ändra programmet genom att välja bland metoderna i klassen `SimpleWindow`. Till exempel kan du rita en kvadrat, ändra färg och linjebredd på pennan, rita fler linjer, skriva text, etc.

12. Öppna filen *LineDrawing.java*. I filen finns följande kod, komplettera programmet så att det fungerar. Raderna med kommentarer ska ersättas med riktig programkod.

```
public class LineDrawing {
   public static void main(String[] args) {
       SimpleWindow w = new SimpleWindow(500, 500, "LineDrawing");
       w.moveTo(0, 0);
       while (true) {
           // vänta tills användaren klickar på en musknapp
           // rita en linje till den punkt där användaren klickade
       }
   }
}
```

Ledtråd: `SimpleWindow` innehåller också metoder för att ta hand om musklick. Dessa metoder har följande beskrivning:

```
/** Väntar tills användaren har klickat på en musknapp */
void waitForMouseClick();

/** Tar reda på x-koordinaten för musens position
    vid senaste musklick */
int getMouseX();

/** Tar reda på y-koordinaten för musens position
    vid senaste musklick */
int getMouseY();
```

Fundera ut vad som händer i programmet. `while (true)` betyder att repetitionen ska fortsätta ”i oändlighet”. Man avbryter programmet genom att välja Quit i File-menyn i `SimpleWindow`-fönstret.

Spara filen, rätta eventuella fel och spara då igen. Provkör programmet.

Checklista

Exempel på vad du ska kunna efter laborationen:

- Starta Eclipse, öppna önskat arbetsområde (workspace) och öppna önskat projekt.
- Skapa .java-filer och skriva in enkla Java-program. (Med enkla program menas här ett program på några rader, som till exempel `Calculator`.)
- Kunna använda metoden `System.out.println` och förstå vad som händer när den används.
- Förstå vad det innebär att läsa in tal från tangentbordet.
- Spara .java-filer (kompilering sker då automatiskt).
- Exekvera program.

- Använda debuggern:
  - Sätta och ta bort brytpunkt.
  - Starta debuggern.
  - Köra fram till en brytpunkt med Resume.
  - Exekvera stegvis med Step over (och senare i kursen med Step Into).
  - Byta mellan debug-perspektivet och Java-perspektivet.