Data och Informationsteknik / Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Magnus Myréen, men mycket av materialet är från Erland Holmströms kurs 2013

Göteborg 2015-08-31

Lab 1: Första labben (2p)

Inledning:

Detta material beskriver den lab som görs i LV 1 + början på LV 2. Det är en okomplicerad labb men det är en hel del att läsa. Hoppa inte över läsandet!

Redovisning: **Redovisas senast tisdag LV2 18.00**.

Handledare är på plats på varje labtillfälle.

För alla labbar gäller:

Alla labbar har instruktioner som skall läsas på hemsidan. Du får poängavdrag om du inte följer dessa.

*Labba kan du vanligtvis göra när du vill och var du vill så länge du lämnar in i tid – du måste alltså normalt **inte** gå på handledning utan kan sitta t.ex. hemma.*

Enbart om du vill ha handledning på labben (eller om labben skall redovisas i labsal) måste du gå på ett handledningspass i labsalarna. Men ett gott råd är att inte börja för sent – Fire stängs automatiskt när tiden är ute.

Samtliga uppgifter skall naturligtvis provköras (där det går) så du vet att de fungerar som du tänkt. Normalt skall du redovisa alla uppgifter. Ibland, som i den här labben, är det bara vissa uppgifter. I så fall anges det vid respektive uppgift om den skall redovisas. Observera att även om vissa deluppgifter inte behöver redovisas så skall dom göras dvs du ansvarar för att du lär dig stoffet dom behandlar.

Man måste programmera mycket för att lära sig programmera. Labkursen är *inte* ett parallellt spår i kursen utan *tentan kommer att innehålla liknande uppgifter som de på labbarna*.

Anvisningar:

En kurs i programmering innehåller mycket labbar. Labbandet kan kännas tungt ibland, speciellt om du försöker labba utan att vara väl förberedd. Men du skall ju nu ägna rätt mycket tid åt programmeringen ett tag. Några kan kanske lägga mindre, många måste lägga mer. Använd tiden väl nu i början, det tenderar att bli mer ont om tid senare. Och lägg en hel del "lästid" före labbandet.

Förberedelse som normalt görs **innan labtillfället:**

- *Hitta en labkompis* som är i samma ”kunskapsgrupp” som du är. Grupperna får du på föreläsning. Bara i undantagsfall kan man labba ensam. I så fall måste man kontakta kursansvarig eftersom Fire inte accepterar ensamgrupper (Du kan alltså inte lämna in labben om du är ensam i din grupp om inte examinator gett klartecken). Det finns inget som säger att man måste labba med samma labkompis alla labbar utan du kan ha olika på varje lab. Se bara till att ”joina” rätt grupp i Fire och skapa inte nya grupper för att byta labkompis.

- *Läs igenom hela labben* och fundera över om du behöver läsa/repetera något. Om du finner att du behöver det så är det bara att sätta igång med läsandet.

För den här labben:

- *Skumma igenom boken* och bekanta dig med dess upplägg. Tänk på att du bara kan ha ett första möte med en bok en gång – slarva inte bort det.

- *Läs igenom:* bokens kap 1. Det finns också en kort introduktion till Unix på hemsidan (under “Länkar”, längst nere på sidan) som du bör läsa. Har du använt kommandofönster i Windows så är det inget nytt, annars blir det lite nyheter som du måste behärska.

Syfte med lab 1:

- att bekanta dig med boken och dess upplägg samt få en första introduktion till ämnet.

- att kort introducera dig till Unix (en ingenjör måste behärska Unix).

- att repetera hur man skriver, kompilerar och kör ett Java program och rättar enkla fel.

- att lära dig reglerna för när labbarna blir godkända

- att lära dig hur man skickar in sin lab för rättning med det datoriserade systemet Fire.

- att visa dig på var du själv kan hitta svar på frågor.

Uppgifter del ”inledning” - Registrering i Fire - inlämningssystemet för labbar.

Den här första uppgiften redovisas genom att du senare skickar in lab 1 till Fire under del D. Det är viktigt att du följer instruktionerna som ges annars får du retur på labben och poängavdrag, det viktigaste med den här labben är att lära dig redovisa korrekt i Fire så vi slipper problem med det sedan. Du måste registrera dig i ett webbaserat system för labhantering som heter “Fire” för att kunna redovisa

labbarna. Alla labbar skickas sedan in elektroniskt och man får också returer elektroniskt. Både du och lab-kompisen måste vara närvarande vid labbandet.

- Hur ni gör när ni *registrerar er* i inlämningssystemet (görs bara en gång) finns beskrivet på hemsidan under ***Laborationerna***.

Uppgifter del A - Unix.

Redovisas genom att du svarar på frågorna märkta med "**(S)**", spara svaren i en fil.

Första uppgiften är att bekanta sig med Unix och att skapa en bibliotheksstruktur för labbarna. Du måste inte göra labbarna på skolans datorer men måste göra den här delen även om du använder egen dator, på den egna datorn i det fallet.

Du använder ditt vanliga CTH-konto när du labbar. Det finns både för och nackdelar med det. En nackdel är att alla kan läsa dina filer om du inte gör något åt det. Och gör du något åt det så kan inte labkompisen läsa dina filer. Du kan använda GitHub.com och BitBucket.org för att hålla ordning på koden. På BitBucket.org kan du ha privata git repositories som endast din labkompis ser (*rekommenderas!*).

Du kanske inte tycker att det spelar någon roll eller att man av princip skall ha en öppen attityd och låta alla läsa "allt" men det finns ett problem. Om någon skulle kopiera din lab och lämna in den och vi upptäcker detta så är det svårt att avgöra vem som gjort originalet och vem som kopierat. Resultatet blir att bägge grupperna misstänks för fusk vilket kan upplevas som rätt obehagligt, även om man är oskyldig.

För att i möjligaste mån skydda dig mot detta skall du skapa ett underbibliotek som skall spärras för sökning för alla utom dig själv enligt nedan. Samtidigt blir det en liten övning i att hantera Unix. Det finns ett unix kommando som heter "**man**" som ger manualsidor för olika kommandon. Så "**man cd**"ger t.ex. en manualsida för kommandot "**cd**". Det är inte alltid lätt att läsa dessa sidor men försök.

A.1. **(S)** Det finns ett kommando som gör att du hamnar i ditt hembibliotek oavsett var du befinner dig i filsystemet. Hur ser kommandot ut? (tips **cd**) Se till att du står i ditt hembibliotek med hjälp av det.

A.2. Gör kommandot "**mkdir tda545**" för att skapa ett tomt underbibliotek.

A.3. **(S)** Vilket kommando används för att få en utskrift som den nedan? Det kommandot genererar nedanstående utskrift i ett av mina bibliotek? Vad betyder de olika kolumnerna? Ge kommandot i ditt hembibliotek och se vad ditt underbibliotek **tda545** har för rättigheter. Är det som det skall dvs ägaren får göra allt medan andra inte får göra något alls? Kan du inte förklara någon kolumn så säg det, hoppa inte bara över svaret för då får du retur.

```
total 3
drwxr-x--- 2 myreen myreen   2 Aug 28 14:13 annat
-rw-r----- 1 myreen myreen 426 Aug 28 14:17 HelloWorld.class
-rw-r----- 1 myreen myreen 224 Aug 28 14:17 HelloWorld.java
```

A.4. **(S)** Troligen är inte rättigheterna rätt, så ge kommandot "**`chmod go-rwx tda545`**". Vilka läs och skrivrättigheter har biblioteket **`tda545`** nu? Vad gör kommandot? (dvs beskriv det)

A.5. **(S)** Flytta dig till **`tda545`** biblioteket och skapa en fil där, enklast kan man göra det med kommandot "**`touch test`**" om filen skall heta "**`test`**". Vilka läs och skrivrättigheter är satta på den filen? Om biblioteket är spärrat för alla utom dig själv så är det OK att filerna i biblioteket är läsbara för alla.

A.6. Alla filer till respektive lab (lab1, lab2, ...) skall ligga i ett eget underbibliotek i biblioteket **`tda545`**. Skapa alltså underbibliotek som du döper till ***`firegruppnummer`*`.lab1`** (och senare till nästa lab ***`firegruppnummer`*`.lab2`** osv) där ***`firegruppnummer`*** är det Firegruppnummer som ni tänker använda för labben (det är det gemensamma labgruppsnumret i Fire ni skall använda).

A.7. Flytta dig till biblioteket ***`firegruppnummer`*`.lab1`** och skapa en **`README`** fil med svaren på dessa och eventuella frågor i följande uppgifter. **`README`** filen är en av filerna som skall redovisas. Skriv **`README`** filen tex med Emacs eller Vim. Exakt vad det skall finnas i **`README`** filen hittar du på kursens hemsidan under *Information om labbarna*. När du skall köra **`zip`**-kommandot för att packa filerna till Fire så skall du sedan stå i biblioteket **`tda545`**, inte i ***`firegruppnummer`*`.lab1`** biblioteken. Om du är i ***`firegruppnummer`*`.lab1`** biblioteket, kör kommandot **`cd ..`** och sedan

```
zip -r firegruppnummer.lab1.zip firegruppnummer.lab1
```

för att göra en zip-fil av ***`firegruppnummer`*`.lab1`** biblioteket. Det är ***`firegruppnummer`*`.lab1.zip`**-filen som vi kallar för zip-filen. Det är den som ni ska skicka in via Fire systemet, som ni hittar via kursens hemsida.

(En hel del av detta, tex att biblioteket skall heta "***`firegruppnummer`*`.lab1`**", är för att underlätta vår rättning)

Uppgifter del B - Java.

Vi skall nu öva på att tyda fel i ett Java program. Redovisas genom att du svarar på frågorna märkta med "**(S)**".

Öppna din editor t.ex. med (om du använder emacs men du får använda vilken editor som helst. Vet du inte vad du skall använda rekommenderar jag dock emacs tillsvidare)

```
emacs HelloWorld.java
```

och skapa filen nedan. Du behöver inte förstå vad koden gör i detalj men tänk på att små eller stora bokstäver är viktiga och att filen måste heta samma sak som klassen (**`HelloWorld`**) med tillägget "**`.java`**".

Indentera också rätt (indragningen av vissa rader) och se till att du inte blandar mellanslag och tabbar. Se not i slutet.

```
public class HelloWorld {
    //  A program to display the message
    //  "Hello World!" on standard output
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Spara sedan filen annars kan inte Java-kompilatorn "se" innehållet i filen (i själva verket är detta ett vanligt fel man gör, att ändra i filen och glömma spara ändringarna). *Du skall också lägga till en kommentar med era namn och gruppnummer först i filen (skall stå i alla filer) och en rad som innehåller vilken editor ni använder (emacs, notepad, eclipse, …)*

Kompilera sedan programmet med Unix-kommandot javac:

```
javac HelloWorld.java
```

Kompileringen bör gå bra om du skrivit av riktigt (annars får du rätta felen och kompilera om) och du kan nu *köra* programmet med

```
java HelloWorld
```

På skärmen skall det komma upp

```
Hello World!
```

Syntaxfel och logiska fel

När du kompilerar dina Java program kommer det att förekomma så kallade *syntaxfel*. Dessa är i någon mening "enkla" för kompilatorn kan alltid hitta dem (men den förstår inte alltid vad som är fel och den "pekar" inte alltid ut rätt placering av felet). Du kommer också att få *logiska fel* dvs ett fel som gör att programmet "fungerar" men ger fel resultat - mycket svårare! Vi kommer att prata mer om fel på föreläsningarna men pröva att ändra i `HelloWorld.java` filen, "låtsas" att du skrev fel från början och inför följande syntaxfel. Ändra rad 3 till (glöm inte spara!)

```
/ "Hello World!" on standard output
```

dvs tag bort en '**/**', och kompilera om filen. Resultatet bör bli

```
    javac HelloWorld.java
    HelloWorld.java:3: illegal start of type
             / "Hello World!" on standard output
^ HelloWorld.java:6: <identifier> expected

}

^ 2 errors
```

Kompilatorn hittar felets plats men misstar sig på vilken typ av fel det blir, "`illegal start of type`", och sedan spårar kompilatorn ur och hittar fler fel som egentligen inte finns (**`<identifier> expected`**), så kallade följdfel. Detta är vanligt, dvs om man får 200 fel så behöver man inte misströsta utan man rättar några i början, så många som verkar vettiga, och sedan kompilerar man om igen.

Ni måste också vara medvetna om att kompilatorn gör flera "pass" dvs den går igenom koden flera gånger i sin jakt på olika fel. Det kan betyda att den bara redovisar ett fel och när du rättat det så kommer det 20 till vid nästa kompilering! Se lite mer om olika fel i föreläsningarna.

Istället för kombinationen kommandofönster och Emacs kan man använda ett sk. IDE (Integrated Development Environment) tex Eclipse (för den avancerade användaren) eller drJava (mer för nybörjaren). Det kan du ladda ner och installera på din dator hemma om du vill (gå till http://www.eclipse.org/ respektive http://drjava.org/). Det finns 3 alternativ av nerladdning för drJava, en version för windows, en för OS X och en jar fil som funkar för Unix. Anledningen till att vi inte använder Eclipse i kursen är att det är en viss inlärningströskel att använda ett sådant program och man klarar sig långt med en editor och ett vanligt terminal fönster.

B.8. **(S)** Hur ser felutskrifterna ut när du kompilerar/kör efter att du lagt in följande fel i **`HelloWorld.java`** programmet? (ett i taget) (klipp och klistra in i redovisningsfilen men se till att tabbar kommer rätt). Redovisa också *hur bra dom pekar ut felet och om du förstår dom*. *Försök förklara dem*. Programmering handlar mycket om att förstå kompilatorns felutskrifter.

a) Tag bort den första parentesen ("{").
b) Tag bort den sista parentesen ("}").
c) Tag bort ordet "**`public`**" för klassen. Vad händer om man tar bort det från metoden **`main`** istället?
d) Tag bort ordet "**`static`**".
e) Stava fel på ordet static tex till "**`satic`**".

Uppgifter del C.

Var hittar man information? (Ingen redovisning behövs men det är viktigt!)

Hemsidan är en källa. Här finns naturligtvis information om kursen, handledare osv men även mycket mer. Under "Links" finns också länkar till

- Java APIn,
- till en "style guide" (*Java style guide (or code conventions)*) dvs hur man bör formatera sin Java kod (*och dessa instruktioner skall ni följa någorlunda*),
- länk till material från div datorintrokurser och mycket mer.

Sun (numera Oracle) har massor med information om Java men är man nybörjare så har man inte så stor nytta av den, man "går vilse" helt enkelt. Bokens register är ett bra sökställe som många verkar glömma bort.

Man skall inte heller underskatta sökmotorerna men var lite försiktig med att kopiera kod till dina labbar. Kan du hitta koden så kan jag...

Att göra: Gå till hemsidan och bekanta dig med den samt undersök en del av länkarna.

Uppgifter del D.

Att redovisa med Fire. Du skall redovisa lösningarna från tidigare övningar i **`README`** filen och skicka in källkoden (den korrekta) för ditt lilla program från del B ovan. Tänk på att kontrollera om det är en blandning av tabbar/mellanslag i början på raderna i filen genom att läsa under "*Tidigare års fel*" nedan.

Vi skall nu se på hur man redovisar sina labbar i Fire. Använd din bläddrare för att ta dig till kursens hemsida och klicka på länken ***Laborationer***. Följ instruktionerna som finns på den sidan.

För labbarna i den här kursen skall det alltid finnas en **`README`** fil och källkod.

Fråga: spelar det någon roll om man inte använder de namn som anges i labuppgifterna? (t.ex. hur bibliotek och filer skall namnges eller hur variabler/metoder/klasser skall namnges)

Egentligen inte men att vissa saker skall heta ett visst givet namn är vanligen enbart för att rättningen skall underlättas. Det blir så otroligt mycket enklare att rätta 40 labbar (som är ungefär vad varje rättare har) om samma saker heter samma sak hos alla grupper. Därför är vi lite petiga med att ni använder de namn vi föreslår.

Tidigare års vanliga fel i lab 1 (Läs detta avsnitt så slipper du någon eller några returer...) Fel som alltid ger retur utan vidare genomläsning:

- Ni har inte gjort någon `zip`-fil eller inte skapat en mappstruktur och kört

`zip`-kommandot därifrån - dvs inte lämnat in på korrekt sätt. Det är viktigt att läsa innantill och se vad man skall göra och hur det skall göras.

Fel som ger retur men vi fortsätter läsa igenom labben:

- Inga onödiga filer får skickas in tex tilde filer eller .class filer. Står tydligt i instruktionerna. Man kan antingen lägga det man vill skicka i en speciell mapp eller också får man städa innan man skickar in till Fire. (Akta att du inte tar bort de filer du just skapat! Att använda GitHub eller BitBucket hjälper här, på det sättet finns filerna ännu på deras servern, ifall du I misstag tog bort dem.)

- Uppg A.3: ofullständigt svar, alla kolumner skall beskrivas. Återigen - alla frågor skall besvaras och det måste ske med fullständiga svar. Kan man inte svaret så kan man åtminstone skriva att man inte kan. (Det är bättre än att inte svara alls.)

- Namn och grupp saknas i någon fil. Skall finnas med i alla filer.

- Indenteringen är felaktig (utan att ni gör följande punkt)

- (OBS **Väldigt vanligt fel**) Ni blandar tabbar och mellanslag i Java filen. Det innebär att indenteringen blir felaktig om inte vi har exakt samma inställning som ni i våra editorer (och det är kanske inte så sannolikt) De flesta editorer kan indentera rätt åt dig - lär dig hur. På hemsidan finns instruktioner om hur man kan ”se” dessa annars osynliga tecken, klicka på ”*Unix kommandon för att ta reda på vad en fil innehåller*” under fliken ”*Links*”. Och en instruktion för hur man fixar det i Emacs, klicka på ”*Att få emacs att inte blanda tabbar och mellanslag*” också under fliken ”Links”.

Vanliga fel som kanske ger retur beroende på hur segt det blir att läsa och beroende på hur många fel det är (dvs ett fel kanske man lipper undan med):

- Om du sitter vid en dator som inte klarar att överföra åäö ordentligt så bör du skriva på engelska. Ett exempel på hur det kan se ut hos mej: *prog lÃ¤s- och kÃ¶rrÃ¤ttigheter.

- Använd ungefär max 80 tecken på en rad i filerna. Detta är lite lurigt för du kan mycket väl arbeta med en editor som bryter raderna åt dig (sk softwrap) och då ser det ok ut hos dig. Men hos mig är det oläsligt med rader på 200 tecken. Detta kan hända även åt andra hållet dvs i det jag skriver till dig. Meddela mig gärna det i så fall.