2D1310 Programmeringsteknik för Bio1 och I1
Laborationer läsåret 2003/2004

Fyll i ditt namn och personnummer med bläck eller motsvarande. OBS: Om inte denna kvittenssida tas med vid redovisningen får du ingen kvittens (resultatet rapporteras dock in i rapporteringssystemet på Nada).

Kursledare är Erik Fransén, erikf@nada.kth.se.

Namn ..... Personnr .....

Laborationer

Laboration 2	Godkänt den ..... (2004−02−04)	Kvitteras	Timmar .....
Laboration 3	Godkänt den ..... (2004−02−11)	Kvitteras	Timmar .....
Laboration 4	Godkänt den ..... (2004−02−18)	Kvitteras	Timmar .....
Laboration 5	Godkänt den ..... (2004−02−24)	Kvitteras	Timmar .....

J-del

Spec	Godkänt den ..... (2004−03−05)	Kvitteras	Timmar .....
Granskning	Godkänt den ..... (2004−05−14)	Kvitteras	Timmar .....
Redovisning	Godkänt den ..... (2004−05−14)	Kvitteras	Timmar .....

Laboration 1: Introduktion till datormiljön

Nyckelord: *katalog, katalogträd, rot, fil, källkod, kompilering, avlusning, program, exekvering, tolkning, prototypprogrammering, textbaserat användargränssnitt, inmatning, utmatning, paket.*

Mål: *Att du efter laborationen ska klara dig på egen hand i datorsalarna, ha registrerat dig på kursen och förstå uppbyggnaden av ett enkelt Javaprogram. Att du ska förstå innebörden av varje nyckelord och förberedelsefrågor samt kunna ge beskrivande exempel.*

Föreberedelse: *Skaffa användarnamn och lösenord och hitta en jämnbra labbkompis.*

Sammanfattning av labben

I den här labben ska du lära dig om operativsystemet **UNIX** och redigeringsprogrammet **Emacs**. Du ska också skriva ett enkelt Javaprogram och registrera dig på labbkursen. Efter den här laborationen ska du klara dig på egen hand i datorsalarna. Betydligt mer information om hur man hanterar datorerna finns i häftet *Introduktion till Unixanvändning* som finns under Nadas datorsystems hemsida (se kurshemsidans information om datorsalar) eller kan köpas från Nadas studentexpedition.

Inloggning

Sätt dig vid en dator tillsammans med din labbkompis och logga in. Skärmen kommer att se ut ungefär så här:

Det finns nu ett eller flera fönster på skärmen. I vårt exempel är det stora fönstret ett terminalfönster. När du skriver i det fönstret ger du kommandon till kommandotolken. Kom ihåg att kommandotolken skiljer mellan stora och små bokstäver. Terminalfönstret motsvarar ett DOS-fönster på en PC.

Längst ned finns frontpanelen. Den används för att starta vissa program, skriva ut filer m m. Läs gärna mer om den i Unixhäftet (säljs på Nadas studentexpedition).

Filer och kataloger

Med en filhanterare kan du utforska datorns och nätverkets *kataloger*. En katalogs innehåll kan bestå av andra kataloger och *filer*. Kataloger som ligger i en annan katalog kallas ibland *underkataloger*.

Kataloger med underkataloger bildar en grenliknande struktur där varje gren är en katalog. Denna grenstruktur kallar man ofta *katalogträd*. Till skillnad från verkliga träd brukar man dock låta katalogträd breda ut sig nedåt och till höger istället för rakt upp. En bieffekt av detta är att katalogträdets *rot* hamnar längst upp till vänster samt att man pratar om att ”gå ned” i en katalog (eller underkatalog).

En fullständig beskrivning av den plats en fil ligger på brukar kallas *sökväg* och kan till exempel se ut på följande sätt:

```
/info/prgt/exempelfiler/labbar/exempel

/home/a/u1xxxxxx/exempel
```

Notera att katalognamn och filnamn separeras av tecknet ’/’. Ett annat speciellt tecken är ’~’ (uttalas ”tilde”) som är en förkortning av den inloggades hemkatalog vilken i verkligheten ligger någonstans under katalogen `/home` i katalogträdet (`/home/...`). Hemkatalogen är den katalog du hamnar i när du loggar in och i vilken du har rätt att spara filer och kataloger.

Starta filhanteraren genom att klicka på filhanterarens ikon (en låda fylld med mappar) i frontpanelen. Filhanteraren kommer då att visa vilka filer du har i din hemkatalog. Du kan skapa nya kataloger med menykommandot `File` → `New Folder...`. Skapa en katalog som du kallar för `prgibio04`. Gå ner i `prgibio04` genom att dubbelklicka på filens ikon. Gå upp igen till din hemkatalog genom att dubbelklicka på ikonen `.. (go up)`.

Terminalfönstret

Vissa kommandon skriver man i terminalfönstret (allt ovanstående går också att göra med skrivna kommandon). Till exempel kan du lista alla filer i en katalog med kommandot `ls` (list). Vill du lista alla underkataloger och filer i kurskatalogen skriver du `ls /info/prgt/exempelfiler`.

Det du framför allt behöver veta är hur du flyttar dig mellan olika kataloger i terminalfönstret. För att gå ner i katalogen `prgibio04` skriver du `cd prgibio04`, vilket utläses *change directory to prgibio04*. För att gå upp en nivå skriver du `cd ..` (glöm inte mellanslaget före punkterna).

Det finns genvägar för att slippa skriva så mycket i terminalfönstret; till exempel behöver du sällan skriva ut hela filnamnet. Tryck på TAB eller  (längst till vänster) så fylls kommandon, filnamn och katalognamn i om de är entydiga (detta kallas ”TAB completion” på engelska). Vill du få upp föregående kommando räcker det att trycka på uppåtpiltangenten som sitter nere till höger på tangentbordet. Trycker du flera gånger bläddrar du successivt tillbaka bland gamla kommandon. Vill du veta mer om något kommando kan du ta fram ett manualblad med kommandot `man kommando`, till exempel `man more` för att få reda på mer om kommandot `more`.

Användbara kommandon

Terminalfönstret har en mängd kommandon och nedan finns exempel på de mest användbara för denna kurs.

Funktion	**Kommandoexempel**
Lista innehåll i aktuell katalog	`ls`
Lista innehåll i aktuell katalog, visa även gömda filer	`ls -a`
Byt aktuell katalog till hemkatalogen `~`	`cd`
Byt aktuell katalog till underkatalogen `lab1`	`cd lab1`
Byt aktuell katalog till "katalogen ovanför"	`cd ..`
Skapa en underkatalog som heter `lab1`	`mkdir lab1`
Kopiera filen `exempel` till din `lab1`-katalog	`cp /info/prgt/exempelfiler/labbar/exempel ~/prgibio04/lab1/`
Kopiera filen `.emacs` till hemkatalogen	`cp /info/prgt/exempelfiler/.emacs ~/.`
Ta bort filen `exempel` **Obs!** Filen försvinner för alltid	`rm exempel`
Ta bort katalogen `lab2` och alla underkataloger **Obs!** Innehållet försvinner för alltid	`rm -r lab2`
Kompilera Javakällkod	`javac Programmet.java`
Exekvera Javaprogram	`java Programmet`
Skriv ut filen `exempel` till skrivaren	`print exempel`

Registrering på kursen

Nu när du provat några kommandon är det dags att du registrerar dig på kursen. Det gör du genom att använda kommandona `course` och `res`. Börja med att använda kommandot

```
res checkin prgibio04
```

och följ instruktionerna. Du har nu registrerat dig på kursen så att vi kan föra in dina resultat. **Obs!** Detta är inte samma registrering som ditt kansli gör.

Skriv sedan

```
course join prgibio04
```

Kommandot `course join` gör att du på ett smidigt sätt får tillgång till kursinformation. Du som jobbar ensam hoppar nu fram till nästa avsnitt "Emacs". Ni som jobbar i grupp fortsätter med att utföra följande kommando:

`course include prgibio04 u1xxxxxx` *(labbkompisens kontonamn)*

Genom detta kommando kan nu båda jobba i katalogen `prgibio04`. Katalogen finns på det konto du just nu använder och heter `prgibio04` (testa med kommandot `ls` eller titta i filhanteraren). Alla kataloger som någon av er skapar under `prgibio04` är gemensamma för labbgruppen.

Logga sedan ut genom att trycka på EXIT-knappen i frontpanelen. Logga in på det andra kontot i gruppen. Upprepa kommandona:

```
res checkin prgibio04
course join prgibio04
```

men byt ut det tidigare `course include` mot

`course labdir prgibio04 u1xxxxxx` *(labbkompisens kontonamn)*

Glöm inte att byta ut labbkompisens namn. Tag fram filhanteraren. Det ska nu finnas en så kallad länk till den katalog, `prgibio04`, som du skapade tidigare.

Webbläsare och kursinformation

Programmet *Netscape* är en webbläsare som startas genom att trycka ner höger musknapp i bakgrunden på skärmen. Du får då upp en meny där du väljer Netscape. (Alternativt kan du använda kommandot `netscape &` i ett terminalfönster.) Du får först upp en sida med licensinformation. Tryck på `Accept`. Klicka en gång i det fönster du nu får upp. Skriv in följande adress i Netscape: `http://www.nada.kth.se/kurser/kth/2D1310/`. Klicka på knappen med ett Bio och I. Du ska nu få upp en sida med kursinformation för programmeringsteknikkursen för Bio1 och I1. Denna sida är viktig! Här kommer vi att lägga ut information under kursens gång, så vänj dig att alltid titta på sidan när du loggar in.

För att slippa klicka dig fram till sidan varje gång ska du spara adressen till sidan med ett bokmärke. Du gör det genom att välja menykommandot `Bookmarks` → `Add bookmark`.

Redigeringsprogrammet Emacs

För att skapa och ändra filer används ett redigeringsprogram. Emacs är ett kraftfullt redigeringsprogram som har flertalet finesser om man skall skriva *Javakällkod*. En av de viktigaste är *indentering*. Med indentering menas att texten i Javakällkoden skjuts in en bit beroende på vilken del av programmet som texten utgör. Personliga inställningar för Emacs kan göras i filen `.emacs` som kopierades tidigare.

Du startar Emacs genom att trycka ner höger musknapp i bakgrunden på skärmen. Du får då upp en meny där du väljer Emacs.

Innan du fortsätter med Emacs ska du skapa en ny katalog i `prgibio04` och kalla den `lab1`. Kontrollera att katalogen `lab1` syns. Skapa sedan ytterligare en katalog i `prgibio04` för varje laboration (`lab2`, `lab3`, ..., `lab5`) så blir det enklare för dig att hålla reda på filerna under kursens gång. Kopiera sedan filen `Maze.java` från kurshemsidan till din katalog `lab1`.

Filer och buffertar

Två viktiga begrepp i Emacs är *fil* och *buffert* (”file”, ”buffer”). En fil är något som finns sparad på en hårddisk. Om datorn slås av kommer en fil finnas kvar och kan utnyttjas när datorn startas igen. En buffert är något som används

tillfälligt under tiden du skriver. Ändringarna i en buffert finns inte automatiskt kvar om datorn slås av. **Det är alltså viktigt** att spara ändringar i en buffert till en fil lite då och då. Detta gäller speciellt om filen ska utnyttjas till något annat (t ex kompilering).

Öppna en existerande eller en ny fil

Öppna filen `exempel` som du kopierat tidigare. Man kan öppna en fil via menyn (`Files` → `Open File`) eller genom tangentkombinationer (`C-x C-f`, d v s håll ned kontrolltangenten, tryck på `x`, håll ned kontrolltangenten, tryck på `f`). Längst ned i Emacs dyker då en rad med den aktuella sökvägen upp. Ersätt den aktuella sökvägen med `~/prgibio04/exempel` . Notera hur Emacs använder tecknet `/` för att separera katalognamn och filnamn, precis som i terminalfönstret. **Tips!** ”TAB completion” kan användas.

När hela sökvägen är inskriven trycker du på returtangenten för att bekräfta valet. Innehållet i filen `exempel` ska nu dyka upp i en av Emacs buffertar och börja enligt:

```
Prinsessan av Babylonien

Det var en mörk vinterafton i den lilla stugan i Skrolycka.
Kattrinna, hustrun i gården, satt och spann, och katten låg i hennes
knä och spann, han också, så gott han kunde.  Mannen, Jan Andersson,
satt vid spisen och värmde sig med ryggen mot elden.  Han hade hela
dagen gått och huggit ved i Erik i Fallas skog, så att ingen kunde
begära, att han skulle ta sig före något arbete nu, när han var hemma.
Inte en gång Kattrinna hade något att anmärka på att han nu inte
gjorde annat än lekte och pratade med deras lilla flicka, som den här
vintern gick på sitt femte år.
...
```

Observera att om den valda filen inte existerar, så kommer Emacs öppna en ny fil med det valda namnet. Med andra ord används med fördel `Files` → `Open File` eller `C-x C-f` både för att öppna en befintlig fil och för att skapa en ny!

Redigering och användbara kommandon

Så fort du redigerar lite i en buffert i Emacs så kommer buffertens innehåll inte stämma överens med den sparade filens innehåll. Detta visas i Emacs genom att markeringen `**` visas till vänster om filnamnet längst ned. Sparas bufferten i en fil så försvinner markeringen.

Börja med att spara den aktuella bufferten i en fil med namnet `exempel.txt`. Att spara till en fil med annat namn görs med `Files` → `Save Buffer As` eller `C-x C-w`. **Var noggrann** med STORA och små bokstäver!

Ändra någonting i bufferten. Notera hur markeringen `**` dyker upp. Spara bufferten (`Files` → `Save Buffer` eller `C-x C-s`) och notera hur markeringen `**` försvinner.

Du har nu använt några av de vanligaste kommandona i Emacs. Det finns otroligt många fler, men för denna kurs kan nedanstående kommandon vara bra att komma ihåg. Med `C-` avses kontrolltangenten nedtryckt, med `M-` avses ”meta”-tangenten nedtryckt. På en UNIX-dator fungerar både ”diamant”-tangenten (bredvid mellanslagstangenten) och `Esc`-tangenten (uppe till vänster)

som metatangent. Trycker man `M-x` kan man skriva in namnet på ett emacs-kommando och köra det. Den som vill lära sig mer om emacs kan läsa Emacs tutorial som nås genom att trycka `F1 F1 t`.

Funktion	**Menyval**	**Kommando**
Avbryt		`C-g`
Öppna befintlig fil/ skapa ny fil	`Files` → `Open File...`	`C-x C-f`
Spara buffert i fil	`Files` → `Save Buffer`	`C-x C-s`
Spara i fil under annat namn	`Files` → `Save Buffer As...`	`C-x C-w`
Stäng buffert	`Files` → `Kill Current Buffer`	`C-x k`
Ångra	`Edit` → `Undo`	`C-_`
Klipp ut från markören till slutet av raden (kan upprepas)		`C-k`
Start av markering		`C-mellanslag`
Slut av markering (kopiera)	`Edit` → `Copy`	`M-w`
Slut av markering (klipp ut)	`Edit` → `Cut`	`C-w`
Klistra in	`Edit` → `Paste`	`C-y`
Indentera Javakällkod på en rad		`TAB`
Skriv ut aktuell buffert	`Tools` → `Print` → `Print Buffer`	
Sök	`Search` → `Search...`	`C-s`
Sök och byt	`Search` → `Query Replace...`	`M-%`
Indentera hela bufferten		`M-x indent-buffer`

Skrivaren

För att få en pappersutskrift av en fil drar du filikonen från filhanteraren till skrivarsymbolen på frontpanelen. I dialogfönstret fyller du i skrivarens namn och trycker sedan på print. Skrivarna på Nada heter oftast samma som salen de är placerade i. Kommandot från terminalfönstret är `print`. *Spara papper! Skriv inte ut i onödan!*

Att köra Javaprogram

Om du inte redan gjort det, börja med att kopiera filen `Maze.java` från kurshemsidan till din egen `lab1`-katalog. Alla Javaprogram måste ha ett namn som slutar med `.java` för att Javakompilatorn ska veta att det är en källkodsfil. Javakällkoden till ett program måste först kompileras till s k Java-byte-kod för att senare kunna *exekveras* (köras). Att kompilera Javakällkod görs med kommandot `javac`, där `c`:et står för ”compiler”. Försök nu kompilera programmet i terminalfönstret:

```
javac Maze.java
```

(Om datorn påstår att det inte finns någon sådan fil så befinner du dig inte i samma katalog som den du kopierade filen till. Flytta i så fall antingen filen, eller byt katalog i terminalfönstret.)

Om filen `Maze.java` innehåller rätt skriven Javakällkod kommer inga fel uppträda och nya filer som slutar med `.class` skapas. Eftersom det finns ett fel i programmet får du ett fel liknande:

```
Maze.java:68: cannot resolve symbol
symbol  : variable brown
location: class java.awt.Color
    Color[] color = {Color.brown,
                             ^
1 error
```

Det betyder att det finns ett fel på rad 68 i filen `Maze.java`. Felet beror på att det inte finns någon fördefinierad färg med namnet `Color.brown` i Java. Öppna filen i Emacs och rätta till detta genom att byta ut färgen mot `Color.black`. Glöm inte att spara. Försök sedan kompilera programmet på nytt. Nu ska det gå bra! Det har nu skapats ett antal nya filer som alla slutar med `.class`. Dessa filer innehåller det körbara programmet. Varje fil motsvarar en del (klass) av programmet. Filen `Maze.class` innehåller den klass där programmet startar.

Det kompilerade programmet (`Maze.class`) kan exekveras med en *Java-tolk* som översätter (tolkar) Java-byte-koden till den kod som datorn förstår (*maskinkod*). Exekvera (kör) nu programmet med hjälp av Javatolken:

`java Maze` *(Obs!* `.class` *skrivs inte ut.)*

Titta nu efter vilka filer du har. Förutom `Maze.java`, som innehåller källkoden, och `class`-filerna, som innehåller exekverbar kod, finns det säkert också filer med ett ˜-tecken efter. Detta är backup-kopior som Emacs skapar; de innehåller tidigare versioner av motsvarande filer utan ˜. Det finns ingen anledning att spara sådana filer. Du kan städa bort dem innan du loggar ut. Om du vill kan du också städa bort `.class`-filerna.

Första Javauppgiften

Du ska skriva ett program som heter `Hej.java` som frågar efter ditt namn och din ålder och som svarar med ditt namn och hur gammal du kommer att vara när du tar examen på KTH (om du tar ytterligare 4 år på dig). Gör så här:

1. Skapa en fil med namn `Hej.java`. Det gör du genom att öppna en fil i Emacs med namnet `Hej.java` i katalogen `lab1`. Om det inte redan finns en fil med det namnet så skapas en tom fil med namn `Hej.java`.

2. Skriv av följande program precis som det står men med ditt eget namn på andra raden:

```
// Hej.java
// Första programmet av <ditt namn>

import java.io.*;

class Hej {
    public static void main(String[] args) throws IOException {
        BufferedReader stdin = new BufferedReader
            (new InputStreamReader(System.in));
```

```
            System.out.print("Vad heter du? ");
            String namn;
            namn = stdin.readLine();
            System.out.println("Hej " + namn + "!");
            // Lägg till fråga om användarens ålder här.
            // Lägg till inläsning av användarens ålder här.
            // Lägg till beräkningar här.
            // Lägg till den slutliga utskriften här.
        }
    }
```

3. Kompilera programmet. Om du har skrivit fel kommer du att få kompileringsfel. Rätta till felen och prova igen. När du inte har några fel testkör du programmet. **Obs!** Kompilera programmet efter varje ändring och rätta till eventuella fel. Det blir mycket lättare att hitta fel då.

4. Lägg till en rad som frågar efter användarens ålder.

5. Lägg till en variabel för strängen som kommer innehålla användarens ålder (`String åldertext`).

6. Lägg till en rad som läser in åldern som en sträng till `åldertext`.

7. Lägg till en variabel som kommer innehålla användarens ålder men som ett heltal (`int åldertal`).

8. Skriv den kod som omvandlar `åldertext` till en `int` (med hjälp av `Integer.parseInt(åldertext)`) och lägger resultatet i variabeln `åldertal`.

9. Lägg till en variabel som ska innehålla åldern på användaren när hon/han tar examen på KTH (`int examensålder`).

10. Beräkna hur gammal användaren kommer att vara då den tar examen och lägg resultatet i variabeln `examensålder`.

11. Lägg till en rad som skriver ut hur gammal användaren kommer att vara vid examen.

12. Kompilera programmet igen. Rätta till eventuella fel tills man kan kompilera och köra programmet. Testkör programmet.

13. Se till att programmet är snyggt indentera genom att använda `M-x indent-buffer` i emacs. Skriv därefter ut programmet på skrivaren.

14. Se till att du förstår programmet.

Redovisning

Om du vill kan du redovisa laborationen för en handledare. Från och med nästa laboration får du bonuspoäng till datorprovet om du redovisar i tid. Kontrollera att dina resultat har blivit rapporterade med kommandot.

```
res show prgibio04
```

Vid redovisningen under J-delen får du visa upp ditt program och dina minnesbilder samt svara på de frågor handledaren ställer.

Tänk på att laborationerna tar betydligt längre tid än den schemalagda datorsalstiden, så förbered dig och påbörja laborationen innan det schemalagda laborationstillfället! Du har tillgång till Nadas terminaler dygnet runt, men salarna kan vara bokade vissa tider. Behöver du hjälp, se kurshemsidan.

Byt lösenord!

Det lösenord du fick med kontot är säkert svårt att komma ihåg, dessutom kan någon ha sett det. Du ska därför byta lösenord. När du fick ditt konto på Nada fick du häftet *Ansvarsförbindelse för datoranvändning på Nada*. Där finns det information och tips om lösenord. Läs igenom det.

Byte av lösenord gör du i terminalfönstret med kommandot `passwd`. Skriv `passwd` och tryck returtangenten. Det kommer upp instruktioner på skärmen som berättar vad du ska göra.

Logga ut

För att logga ut trycker man på EXIT-knappen.

Kontroll nästa laboration

- ☐ Jag har registrerat mig på kursen med kommandor res.
- ☐ Jag förstår innebörden av alla nyckelord och kan visa användbara exempel.
- ☐ Jag vet hur jag hittar information på kurshemsidan.
- ☐ Jag vet hur man skapar, öppnar och sparar filer samt kan använda Emacs.
- ☐ Jag kan skriva ett enkelt Javaprogram.
- ☐ Jag förstår vilka delar gör vad i Hej.java, speciellt hur utskrifter från och inläsning till programmet fungerar.
- ☐ Jag kan deklarera en variabel och tilldela variabler olika värden.
- ☐ Jag kan kompilera och provköra Javaprogram.
- ☐ Jag kan läsa kompileringsmeddelanden och rätta fel i ett Javaprogram.

Laboration 2

Instuderingsfrågorna nedan är indelade efter föreläsningarna. Praktikfrågorna kräver ett program som lösning. Varje vecka ska du dels skriva ett program enligt anvisningarna nedan, dels ska du göra instuderingsfrågorna.

Gör följande

Skriv ett program som frågar efter ett tal och om talet är ett jämnt tal så ska talet delas med 2 och resultatet skrivas ut och annars programmet ska meddela att talet är inte delbart med 2.

```
datorn> java Jämnt.java
datorn> javac Jämnt
Mata in ett tal: 3
Talet 3 är inte delbart med 2
datorn>
```

Skriv ett program som frågar efter ett tal och använder en for-slinga för att skriva multiplikationstabellen för talet. Ett exempelkörning kan se så här ut:

```
datorn> java Multiplikation.java
datorn> javac Multiplikation
Mata in ett tal: 3
3 6 9 12 15
18 21 24 27 30
datorn>
```

Praktikfrågor

2.1. Vad är det för skillnad på print() och println()?

2.2. Vad får man för felmeddelande om man försöker använda en variabel som man inte har deklarerat?

2.3. Gör så att programmet skriver ut följande om man matar in talet 3.

```
30 27 24 11 18
15 12 9 6 3
```

2.4. Gör så att programmet skriver ut följande om man matar in talet 3.

```
3 9 15 21 27
```

2.5. Gör så att programmet skriver ut följande om man matar in talet 3.

```
6 12 18 24 30
```

2.6 Gör punkt 2.1 till 2.5 men den här gången ska du använda dig av en `while`-sats.

Teorifrågor

2.7. Vad är det för tre saker som kännetecknar en variabel och vad har de för betydelse?

fortsätning följer på nästa sida

2.8. Redogör de olika delar av en `for` -sats

```
for (A;B;C){
    D
}
E
```

i vilken ordning kommer `for` -satsen att exekveras t.ex: DABC DAB DAB DABC...E eller?

Laboration 3

Instuderingsfrågorna nedan är indelade efter föreläsningarna. Praktikfrågorna kräver ett program som lösning. Varje vecka ska du dels skriva ett program enligt anvisningarna nedan, dels ska du göra instuderingsfrågorna.

Gör följande

Skriv ett program som räknar ut BMI (vikt i kg / (längd i meter * längd i meter)), där en metod används för beräkningen. Experimentera därefter med detta för att besvara följande:

Praktikfrågor

3.1 Vad får man för felmeddelande om man har fel datatyp på en anropsparameter?

3.2 Vad får man för felmeddelande om man skriver anropsparametrarna i fel ordning?

3.3 Vad får man för felmeddelanden om man anropar en metod med fel returtyp (void resp datatyp), parameterordning, parameterantal resp parametertyp samt redovisa hur ovanstående fel uppstår.

3.4 Måste man välja samma variabelnamn för en förmellparameter i en metoddeklaration som variabelnamn i metodanropet?

Teorifrågor

3.5. Redogör för de olika delarna i metodhuvudet och deras funktion.

3.6. Vad är parametrar bra till?

3.7. Varför behöver man anropa metoder?

3.8. Varför behöver man metoder?

3.9. Vad är det för skillnad mellan anropsparametrar och formella parametrar?

3.10. Vilka parametertyper möjliggör att informationen anropsvariabeln håller reda på kan påverkas av metoden som man anropar? och för vilka datatyper gäller inte detta?

Laboration 4

Klasser, klassmetoder, Instanser, Instansmetoder. Praktikfrågorna kräver ett program som lösning.

Gör följande

Skriv en separat klass som hanterar en bil med instansvariablerna körsträcka, inköpsår, inköpskostnad, driftskostnad, märke och vikt och skriv instansmetoder för att beräkna årlig körsträcka samt kilometerkostnad.

Gör sedan följande

Lägg till kod så att du har minst tre instanser av bilklassen. Lägg sedan till en klassvariabel som representerar landsbokstaven och ge den värdet 'S'. (Eftersom vi antar att alla bilar är svenska bilar är det logiskt att ha denna variabel just som en klassvariabel.) Lägg därefter till en klassmetod som returnerar värdet av landsbokstaven, och anropa denna metod i ditt program. Ifall du via en referens till en av dina bilar ändrar landsbokstaven till 'N' (Norsk), vad händer då med landsbokstaven för de andra två bilarna? Experimentera därefter med programmet för att besvara följande:

Praktikfrågor

4.1. Visa hur man får felmeddelandet NoClassDefFoundError.

4.2. Hur får man en NullPointerException och varför uppstår det i praktiken?

4.3. Vad får man för felmeddelande när man felaktigt försöker anropa en instansmetod som om den vore en klassmetod och tvärtom?

4.4. Vad får man för felmeddelande när man felaktigt försöker nå en instansvariabel som om den vore en klassvariabel och tvärtom?

4.5. Visa fördelen med att kunna ha flera instanser av samma klass.

Teorifrågor

4.6. Vad är det för skillnad på en instans och en referens?

4.7. Vad är det för skillnad på en klass, en metod och en sats?

4.8. Vad innebär en instansiering?

4.9. Vad är det för skillnad på en instansvariabel och en lokal variabel?

4.10. Vad är det för skillnad på primitiva variabler och `String` (rita)?

4.11. Vad är en wrapperklass?

4.12. När och varför är det bra att rita upp minnet?

4.13. Vad är `this` och när existerar den (rita)?

4.14. Vad kännetecknar en klassvariabel respektive en instansvariabel?

4.15. Ge exempel på när man bör välja att implementera variabler och metoder som klassvariabler och -metoder respektive instansvariabler och -metoder.

4.16. Varför behöver man konstruktorer?

4.17. Hur ser strukturen ut när man deklarerar en klass?

4.18. Hur hanterar Java namnkollisioner? t.ex. då en instansvariabel har samma namn som en lokal variabel.

Laboration 5

Föreläsning 5: Vektorer. Praktikfrågorna kräver ett program som lösning.

Gör följande

Utgående från förra veckans program, lägg till ytterligare en klass som skapar och hanterar flera mbil-instanser i en array ([]-vektor). Gör samma sak där du använder klassen Vector istället för arrayen. experimentera för att besvara följande:

Praktikfrågor

5.1. Vad får man för felmeddelande om man försöker komma åt ett element utanför vektorns gränser?

5.2. Hur lägger man in element i en hakvektor respektive Vector?

5.3. Hur byter man ut element i en hakvektor respektive Vector?

5.4. Hur visar man att inläsningen till en vektor fungerar.

5.5. Hur anropar man en instansmetod i varje element i en hakvektor respektive Vector?

Teorifrågor

5.6. Vad är det för skillnad på hakvektorer och Vector?