DA339A Laboration L5

Syfte

Den här laborationens syfte är att öva på att använda selektion som verktyg vid problemlösning och att skriva kod som innehåller selektioner. I detta ingår att kunna beskriva en lösning med pseudokod, rita ett aktivitetsdiagram och kunna redogöra för villkors utformande med hjälp av sanningstabell samt att implementera lösningar med selektion i kod och exekvera denna.

Redovisning

En delmängd av uppgifterna i den här laborationen ska redovisas för godkänd laboration L5. Godkänd laboration L5 krävs för godkänt betyg G på provkod 2007 Laborationer och workshoppar del 1.

Vilka uppgifter som ska kunna redovisas är noterat efter uppgiftens rubrik. Även om inte alla uppgifter ska kunna redovisas för godkänd laboration L5 förväntas studenten för sin egen övnings skull arbeta med alla uppgifterna (enklare uppgifter som inte kräver redovisning innebär ofta en lösning på ett delproblem i de större uppgifter som ska redovisas).

Laboration 5 kan tidigast redovisas, på laboration, vid det angivna Kronox-schemat tillfället (kurstakt L5). Är man inte redo att redovisa vid detta tillfälle kan laborationer redovisas vid vilket senare labb-hjälp-tillfälle som helst som inte är i workshop-format.

Genomförande av redovisning

Laborationen redovisas muntligt och genom uppvisande av käll-kod, kompilering och exekvering av kod. Studenten ska kunna visa upp sin källkod på dator för lärare/assistent som examinerar och kunna visa hur hen kompilerar och exekverar koden på dator. Studenten ska också kunna svara tillfredställande på frågor om sin kod och de lösningar hen implementerat samt kunna demonstrera förmåga att göra mindre ändringar i koden direkt, kompilera och exekvera igen (exempelvis ändra indata eller något gränsvärde). Studenten ska även kunna visa upp andra lösningar som krävs i laborationen exempelvis diagram eller sanningstabeller.

Redovisningen är en examination och är fokuserad på att göra en bedömning av den kod studenten producerat. Det finns inte tid vid redovisningen att ge utförligare kommentarer om de uppgifter som inte väljs ut att demonstreras vid redovisningen. Det finns vid denna redovisning inte heller tid att studenten beskriver sin hela tankeprocess eller förklarar hela programmet. Den lärare eller assistent som studenten redovisar för kommer att ställa frågor och be studenten demonstrera vissa saker. Diskussioner utöver detta finns det inte utrymme för vid redovisningen.

Önskar studenten återkoppling i större utsträckning på en lösning ges detta som vanlig hjälp vid laborationer.

Förberedelser för redovisning

Studenten förväntas vara förberedd när redovisningen startar och ha följande förberett innan redovisningen startar:

- Ha kommandotolk startad och redo att kompilera och exekvera de uppgifter som kan redovisas (det vill säga ha navigerat till den katalog du har dina filer i innan din redovisningstid startar).
- Ha källkodsfiler öppnade och tillgängliga i editor för att kunna visa upp koden och kunna göra ändringar i denna vid förfrågan.
- Ha diagram eller tabeller redo att redovisas innan redovisningen början samt vara så pass klara och renskrivna så att det inte tar för lång tid att gå igenom.

Generella krav vid redovisning

För godkänd redovisning krävs:

- Väl formaterad källkod.
- Källkod ska gå att kompilera och exekvera med ett resultat som efterfrågas för respektive uppgift via kommandotolk.
- Att uppgifterna är lösta på rimligt sätt oavsett om det är i kod eller annan form av dokumentation av lösning som efterfrågas.
- Studenten ska muntligen kunna förklara sina lösningar och beskriva varför lösningen ger det efterfrågade resultatet.
- Studenten ska kunna visa förmåga att göra mindre ändringar i koden och kompilera, exekvera den ändrade koden med efterfrågat resultat via kommandotolk

Ingen skriftlig inlämning sker i redovisningen. Studenten ska alltid vara beredd på att kunna visa legitimation vid redovisning.

Förberedelser

Den här laborationen är direkt kopplad till det innehåll som finns i föreläsning F4 till och med F6. Vissa av uppgifterna i laboration L5 förbereddes i L4 som lösningar i pseudokod och/eller aktivitetsdiagram.

För laboration L5 förutsätts det att du även behärska tidigare genomgånget innehåll på kursen.

Du behöver ladda ner filen *da339a_L5_filer_ht24.zip* och packa upp denna inför laborationen då den innehåller filer som du ska utgå ifrån för vissa av uppgifterna.

Uppgifter

Uppgift 1a

Utgå från koden i filen *AskNumber.java*, och skriv ett program som:

1. Frågar efter ett heltal
2. Om heltalet är större än 100 skriver ut "Talet är större än 100"

Uppgift 1b

Utgå från din lösning i Uppgift 1a och utöka ditt program så att programmet:

1. Frågar efter ett heltal
2. Om heltalet är större än 100 skriver ut "Talet är större än 100" annars ska programmet skriva ut "Talet är högst 100".

Uppgift 2

Skapa en fil med namnet *AskAge.java*. I filen ska följande kod finnas:

```
import java.util.Scanner;
public class AskAge{
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    //Här ska du skriva mer kod själv
  }
}
```

Observera att namnet på filen är samma som det står efter `public class` på rad 2 ovan. Detta gäller i alla filer du sett hittills och du kommer på sikt att förstå varför det måste vara så här. Nyttja detta nu för att göra din egen fil med källkod från början.

I koden ovan ska du nu modifiera lösningen från Uppgift 1b så att programmet:

1. Frågar efter användarens ålder i år och läser in denna som ett heltal
2. Om åldern är mindre eller lika med 17 ska programmet skriva ut "Du är ett barn" annars ska programmet skriva ut "Du är vuxen"

Uppgift 3

Utgå från filen *AskName.java*. I den här filen så läser vi in en sträng från användaren istället för ett heltal. Vi kan sedan ta reda på hur många tecken den här strängen innehåller genom att skriva `variabelnsNamn.length()` detta ger oss längden på strängen - hur många tecken där finns som ett heltal. Vi behöver spara svaret i en variabel som kan lagra ett heltal:

```
int sparaLängd = strängVariabelnsNamn.length();
```

Vi använder detta i filen *AskNamne.java* för att ta reda på längden på den sträng som användaren skriver in. Titta på hur detta ser ut i filen *AskName.java*.

Experimentera med detta och skriv ut den textrad användaren skrev in och längden på strängen. Hur kan du mixa att skriva ut en text och en variabel? Titta i filen *Ticket.java* från L3 om du inte kommer ihåg detta.

Skriv en selektion som:

- om namnet användaren anger är mindre än 5 tecken långt skriver ut "Ett kort namn med *X* bokstäver" där *X* är antalet tecken i strängen
- annars skriver ut "Ett långt namn med *X* bokstäver" där *X* är antalet tecken i strängen

Uppgift 4a

Antag att du har följande kod:

```
int number = input.nextInt();
boolean result = //här ska stå något villkor
```

Hur ska villkoret se ut för att värdet på variabeln *result* ska bli **true** då:

- number är större än 67
- number är exakt 3, 6 eller 7
- number är i intervallet 25-50
- number tillhör något av intervallen 1-4 eller 7-9
- number är negativt eller större än 10

Skriv villkoren i vanlig text först och sedan med operatorer som du gjort i kod. I nästa uppgift ska du implementera kod som använder dem.

Uppgift 4b

Utgå från den kod som finns i filen *Conditions.java*. Utöka koden med ett antal selektioner som visar om variabeln number är:

- större än 67
- exakt 3, 6 eller 7
- är i intervallet 25-50
- tillhör något av intervallen 1-4 eller 7-9
- är negativt eller större än 10

Lösningen ska ha formen:

```
if(villkor 1){
  skriv ut vad som stämmer för number
}

if(villkor 2){
  skriv ut vad som stämmer för number
}
…
```

Uppgift 4c

Uppgiften är lite mer utmanande tankemässigt men kräver inte mer avancerad kod. Du kan nyttja matematiska resonemang och operatorn modulus % för att lösa dessa problem.

Hur ska villkoret se ut för att värdet på variabeln *result* ska bli **true** då

- number är delbart med 3
- number är ej delbart med 4
- entalsiffran i number är 7
- hundratalssiffran är 7

Uppgift 5a

Utgår från koden i java-filen *Week.java* som skriver ut namnet på den dag matat in. Till exempel om du matar i 1 så skrivs måndag ut matar du in 3 så skrivs onsdag ut.

Modifiera nu programmet på så att det skriver ut alla dagar tills helgen. Till exempel om användaren anger 3 så skrivs onsdag, torsdag och fredag ut men inte lördag och söndag.

Ledning: Använd en switch-sats och starta inte veckan på måndag.

Uppgift 5b

Ändra i programmet från 5a så att det svarar med motsvarande rader för veckodagarna från The Cures ”Friday I’m in love” nedan och också skriver ut rader för lördag och söndag. Observera att tisdag och onsdag ger samma rad/svar. Om vi anger 1 får vi då hela veckan:

Monday you can fall apart
Tuesday, Wednesday break my heart
Oh, Thursday doesn't even start
It's Friday, I'm in love
Saturday, wait
And Sunday always comes too late

Uppgift 6

Utgå från koden i java-filen *Savings.java*. Just nu beräknar programmet hur mycket du skulle spara utifrån hur många veckor du sparar en viss summa och skriver ut den summa du uppnår med det sparandet. Du ska göra ett tillägg i programmet som ger duktiga sparare bonus innan den totala summan man sparat skrivs ut.

Detta sker enligt följande regler:

- om sparkapitalet är minst 5000kr ska spararen ges berömmet ”Du är en duktig sparare!” samt få 100kr i bonus
- annars om sparkapitalet är minst 2500kr så ska samma beröm som ovan ges och spararen för 50kr i bonus
- annars ges spararen kritik för bristande sparförmåga: ”Du är ej bra på att spara!”

Därefter meddelas sparresultatet.

Uppgift 7

Utgå från koden i java-filen *MaxNumber.java*. Den här koden visar en lösning för att avgöra vilket av två heltal som är störst eller om de är lika stora. Anpassa koden för att fungera med tre inlästa heltal utifrån din lösning i L4 Uppgift 2. Din lösning ska ange vilket av de tre talen som är störst, eller om alla tre talen är lika stora, eller vilka två av talen som är lika och större än det sista talet. Du behöver inte implementera någon felhantering utan förutsätt att användaren anger ett heltal.

Uppgift 8a

Utgå från uppgift 7 i L4 och skriv kod för den del av lösningen som avgöra vilken kostnad som ska läggas till. Utgå från koden i filen *DentistReception.java*. Använd konstanterna som finns deklarerade i koden när du ska beräkna kostnaden.

Programmet behöver inte innehålla någon iteration utan det är tillräckligt att ange en behandling.

Uppgift 8b

Utgår från din lösning på Uppgift 8a och din lösning för L4 Uppgift 7 och lägg till en selektion efter att kostnaden sats som avgör om det ska ske något avdrag på 10% eller inte. Använd en variabel eller konstant för att på förhand ange hur stort avdrag i procent som kan ske och en variabel eller konstant för att ange vad gränsen för avdrag går.

Om inget avdrag sker så ska programmet tala om det för användaren.

Variera vad värdet på variabeln **cost** sätts till i början av programmet för att kontrollera olika utfall.

Programmet behöver inte innehålla någon iteration utan det är tillräckligt att ange en behandling.

Uppgift 9

Använd en sanningstabell för att avgöra om uttrycket **(!a||!b)** är likvärdigt med **(!(a&&b))**. Ditt svar ska visa hela uppställningen i sanningstabellen och svara på om uttrycken är likvärdiga eller inte och hur detta utläses ur tabellen.

Uppgift 10

Använd en sanningstabell för att avgöra om uttrycket **(!a||b)** är likvärdigt med **(a&&!b)**. Ditt svar ska visa hela uppställningen i sanningstabellen och svara på om uttrycken är likvärdiga eller inte och hur detta utläses ur tabellen.

Uppgift 11

Använd en sanningstabell för att avgöra om uttrycket **(!(a || b) || c)** är likvärdigt med **((!a && ! b) || c).** Ditt svar ska visa hela uppställningen i sanningstabellen och svara på om uttrycken är likvärdiga eller inte.

Uppgift 12

I denna uppgift ska vi testa vad som händer när man vänder på uttrycket lika (==) till olika (!=). Skapa en ny java-fil med namnet *uppgift12.java*. Deklarera två int variabler (till exempel total och sum) och ge dem båda värdet 20. Därefter kör ett test (if-sats) som kollar om variablerna är lika med varandra, om det är sant ska meddelandet ”total equals sum” skrivas ut. Kompilera och kör koden.

Ändra nu värdet på en av int variablerna till 10 i stället för 20, kompilera och kör programmet. Om allt gått bra så kommer inget att skrivas ut i Git Bash (på grund av att de inte är lika).

Ändra värdet på variabeln som har värdet 10 tillbaka till 20. Skriv programkod som kollar om de två int värdena är olika (!=), om det är sant ska meddelandet ”total not equal to sum” skrivas ut. Kompilera och kör koden.

Som i testet med lika (==) kommer inget att skrivas ut. Ändra värdet på en av int variablerna till 10 i stället för 20 igen, kompilera och kör programmet. Nu kommer meddelandet ”total not equal sum” skrivas ut.

Lösningsförslag finns i slutet av instruktionerna (Notera att varje test körs enskilt, kommentera bort de testerna som du inte vill testa innan kompilering).

Uppgift 13

I denna uppgift ska vi göra några tester där vi kommer att jämföra heltal med char och att det kan fungera om man vet vilket tecken (till exempel ’A’) motsvarar i Unicode/ASCII. Länken nedan innehåller en tabell med Unicode/ASCII nummer:
https://www.javatpoint.com/java-ascii-table

Börja med att ladda ner java-filen *UnicodeTest.java* från kurssidan (finns i zip-filen som heter Uppgift13.zip). Programmet kan ta emot en input från användaren med hjälp av en scanner och konverterar den till en char som vi därefter använder i de tre testerna nedan:

- Test 1 – Kolla om input från användaren finns i intervallet A – Z, decimal värde intervall 65 – 90 (kommer redan med i java-filen).
- Test 2 – Kolla om input finns inom intervallet a – z.
- Test 3 – Kolla om input finns inom intervallet 0 – 9.

Kod exempel från *UnicodeTest.java* för Test 1 (Notera i koden att vi gör två tester för A – Z):

```
1  import java.util.Scanner;
2
3  public class UnicodeTest
4  {
5    public static void main(String[] args)
6    {
7
8      System.out.println("Testa vilket intervall input ligger i");
9      System.out.println("A-Z, Decimal värde intervall: 65 - 90");
10     System.out.println("Ange input: ");
11
12     Scanner scannerInput = new Scanner(System.in);
13     char input = scannerInput.next().charAt(0);
14
15     // Testar med tecken
16     if ((input >= 'A') && (input <= 'Z')) {
17       System.out.println("Ja innanför intervallet A-Z");
18     }
19
20     // Testar med decimal värdet
21     if ((input >= 65) && (input <= 90)) {
22       System.out.println("Ja innanför intervallet A-Z");
23     }
```

Med koden i bilden som en bas, utöka testerna så att ni också kör Test 2 och Test 3 i koden för både char delen samt decimal värde delen.

Lösningsförslag finns i slutet av instruktionerna.

Uppgift 14 REDOVISAS

Uttrycket XOR innebär att:

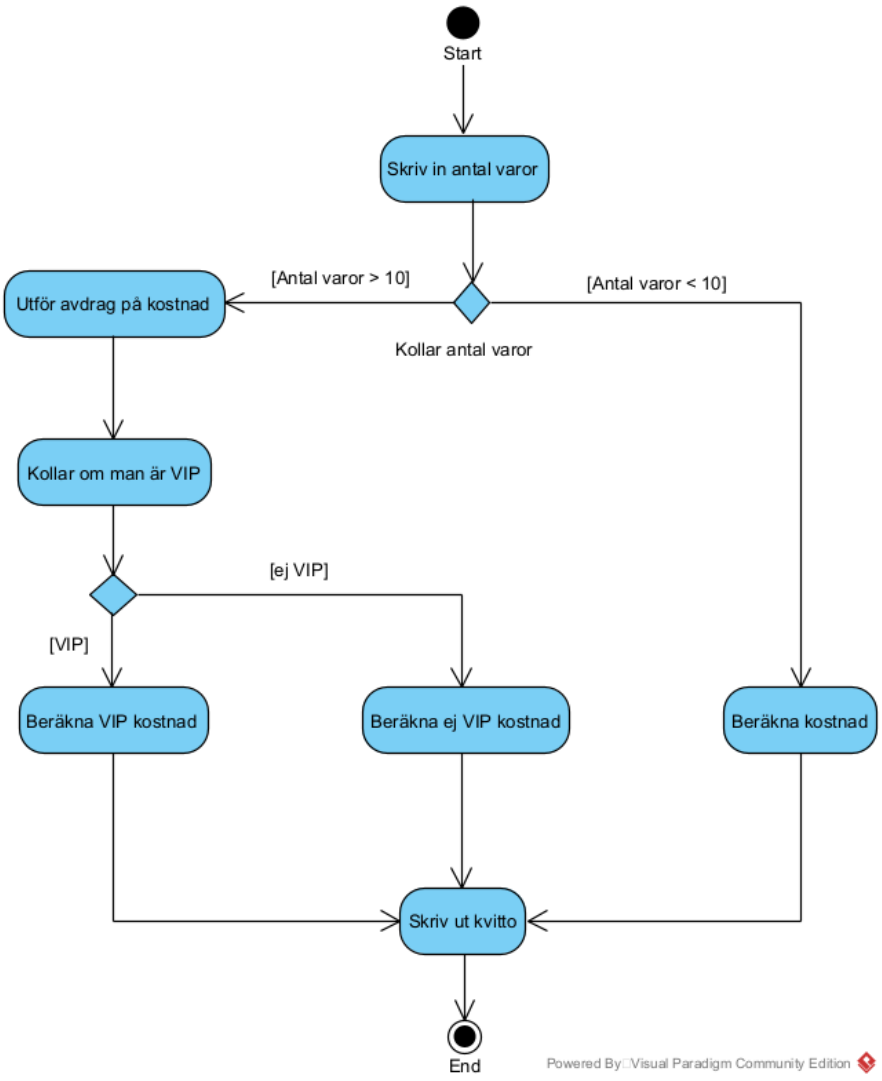
- (a XOR b) är sant om a eller b är sant
- (a XOR b) är falskt om både a och b är sanna
- (a XOR b) är falskt om både a och b är falska

Alla programspråk har inte en specifik operator för XOR som man har för AND och OR. Man kan dock skapa ett sammansatt uttryck med AND och OR som motsvarar XOR.

Använd en sanningstabell för att visa om uttrycket ((a || b) && !(a&&b)) motsvarar XOR eller inte. Förklara hur du utläser ur sanningstabellen om uttrycken XOR och ((a || b) && !(a&&b)) är likvärdiga eller inte.

Uppgift 15a REDOVISAS

Med hjälp av aktivitetsdiagrammet nedan, skriv ett Java-program som implementerar funktionaliteten som visas i diagrammet. Ett meddelande ska finnas som säger om kunden har mer eller mindre än 10 varor och om kunden ska få reducerad kostnad. Det ska också vara ett meddelande om kunden är en VIP medlem eller inte. Beroende på vilken av de tre olika beräkningsaktiviteterna man hamnar i ska olika uträkningar ske (De tre beräkningarna måste vara unika/olika/skilja sig från varandra, det spelar ingen roll hur kostnaden beräknas i övrigt). Det sista som ska hända är att den totala kostnaden efter beräkningarna ska skrivas ut i form av ett kvitto.



Ett exempel på hur ett kvitto kan se ut är:

Antal objekt som du vill köpa är mer än 10.
Du är VIP och får VIP reducerad frakt!
Kostnaden blir 180.0 SEK

Uppgift 15b REDOVISAS

Skriv om de nästlade if-satserna till sammansatta uttryck i villkor.

```
int i = 1;
String s = "s";

if (i == 0 && s == "s") {
    System.out.println("0");
}
else if (i == 0 && s == "r") {
    System.out.println("0");
}
else if (i == 1 && s == "s") {
    System.out.println("1");
}
else if (i == 1 && s == "r") {
    System.out.println("1");
}
else if (i == 2 && s == "s") {
    System.out.println("2");
}
else if (i == 2 && s == "r") {
    System.out.println("2");
}
else {
    System.out.println("3");
}
```

Lösningar

Uppgift 9

a	b	!a	!b	**!a || !b**	a&&b	**!(a&&b)**
T	T	F	F	**F**	T	**F**
T	F	F	T	**T**	F	**T**
F	T	T	F	**T**	F	**T**
F	F	T	T	**T**	F	**T**

Kolumnerna för de två uttrycken (!a||!b) och (!(a&&b)) är samma alltså är dessa likvärdiga.

Uppgift 10

a	b	!a	!b	**!a||b**	**a&&!b**
T	T	F	F	**T**	**F**
T	F	F	T	**F**	**T**
F	T	T	F	**T**	**F**
F	F	T	T	**T**	**F**

Uttrycken är INTE likvärdiga då deras kolumner är olika.

Uppgift 11

a	b	c	!a	!b	a||b	!(a||b)	**!(a||b)||c**	!a&&!b	**(!a&&!b)||c**
S	S	S	F	F	S	F	**S**	F	**S**
S	S	F	F	F	S	F	**F**	F	**F**
S	F	S	F	S	S	F	**S**	F	**S**
S	F	F	F	S	S	F	**F**	F	**F**
F	S	S	S	F	S	F	**S**	F	**S**
F	S	F	S	F	S	F	**F**	F	**F**
F	F	S	S	S	F	S	**S**	S	**S**
F	F	F	S	S	F	S	**S**	S	**S**

Kolumnerna är lika för de två uttrycken – alltså är dessa likvärdiga.

Uppgift 12

```java
public class uppgift12
{
  public static void main(String[] args)
  {
    // == operator
    int total = 20; // total samma som sum, 20
    int sum = 20;
    if (total == sum){
      System.out.println("total equals sum");
    }

    int total = 10; // Ändra total till 10
    int sum = 20;
    if (total == sum){
      System.out.println("total equals sum");
    }

    // != operator
    int total = 20; // total samma som sum, 20
    int sum = 20;
    if (total != sum){
      System.out.println("total not equal to sum");
    }

    int total = 10; // ändra total till 10
    int sum = 20;
    if (total != sum){
      System.out.println("total not equal to sum");
    }
  } // Slutet av main()-metoden
} // Slutet av uppgift12 klassen
```

Uppgift 13

```java
import java.util.Scanner;

public class UnicodeTest
{
  public static void main(String[] args)
  {
    System.out.println("Testa vilket intervall input ligger i");
    System.out.println("A-Z, Decimal värde intervall: 65 - 90");
    System.out.println("a-z, Decimal värde intervall: 97 - 122");
    System.out.println("0-9, Decimal värde intervall: 48 - 57");
    System.out.println("Ange input: ");

    Scanner scannerInput = new Scanner(System.in);
    char input = scannerInput.next().charAt(0);
    // Testar med tecken
    if ((input >= 'A') && (input <= 'Z')) {
      System.out.println("Ja innanför intervallet A-Z");
    }else if ((input >= 'a') && (input <= 'z')) {
      System.out.println("Ja innanför intervallet a-z");
    }else if ((input >= '0') && (input <= '9')) {
      System.out.println("Ja innanför intervallet 0-9");
    }
    // Testar med decimal värdet
    if ((input >= 65) && (input <= 90)) {
      System.out.println("Ja innanför intervallet A-Z");
    }else if ((input >= 97) && (input <= 122)) {
      System.out.println("Ja innanför intervallet a-z");
    }else if ((input >= 48) && (input <= 57)) {
      System.out.println("Ja innanför intervallet 0-9");
    }
  }
}
```