

# DA339A Laboration L5

## Syfte

Laborationen syftar till att ge förståelse för hur kod kan skrivas för att endast vissa delar körs med hjälp av selektioner. Detta förstärks även genom övningar med sanningstabeller. Slutligen innehåller laborationen ett antal obligatoriska uppgifter som ska redovisas.

## Följande tas upp i laborationen:

- Selektioner
- Sanningstabeller
- Förstärkning av kunskaper från Laboration 3 och 4.

## 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 (enkla 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-hjälptillfälle som helst som inte är i workshop-format.

## Genomförande av redovisning

Laborationen redovisas muntligt och genom uppvisande av källkod, 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 (Gitbash/terminal) 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 börjar 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

Bygger på information från samtliga föreläsningar till och med F5 samt kurslitteratur.

## Uppgifter

I Laboration 3 introducerades uppgiftsbeskrivningar med följande struktur: problembeskrivning, information, gör så här och tips. Denna laboration återgår till detta format för att främja eget analytiskt och kritiskt tänkande. Instruktioner och vissa tips på vad som ska göras kommer att ges, men försök att själv tolka och strukturera uppgiften på samma sätt som i föregående laboration. Specifika tips för mer utmanande moment kommer att markeras tydligt. Om det inte går att komma vidare bör man alltid fråga kurskamrater eller labbassistenter innan facit används.

Om filen inte går att köra och följande fel uppstår:

*"Error: Could not find or load main class "klassnamn" Caused by: java.lang.NoClassDefFoundError:".*

beror detta ofta på att ett paket har lagts till högst upp i koden. I sådana fall måste kommandot för att exekvera ange paketnamnet, exempelvis: *java paketnamn.filnamn*. Alternativ ta bort den pakettilldelningen i början av källkoden.

### Uppgift 1 – Avgör storlek på ett tal

I denna uppgift ska ett enkelt program med namnet *AskNumber* skapas. Programmet *frågar efter ett heltal* och *sparar* värdet i en *lämplig variabel*. Om heltalet är *större än 100* skrivs 'Talet är större än 100' ut. *Annars* skrivs 'Talet är högst 100' ut."

### Uppgift 2 – Avgöra om någon är vuxen

I denna uppgift ska programmet *AskAge* skrivas. Programmet ska *fråga användaren* efter dennes ålder och spara den i en *lämplig variabel*. Därefter avgör programmet om personen är barn eller vuxen. Om åldern är *17 år eller yngre* skrivs "Du är ett barn.", *annars* skrivs "Du är vuxen."

### Uppgift 3 – Räkna längd på ord

I denna uppgift ska programmet *WordLength* skapas. Programmet *ber användaren skriva ett ord* och *sparar detta i en lämplig variabel*. Därefter *räknar programmet hur långt det angivna ordet är* och sparar detta i en *lämplig variabel*. Slutligen *skrivs resultatet ut till användaren i formen av "Ordet x är y tecken långt!"*.

Tips: Använd den inbyggda metoden `length()` i strängklassen för att avgöra längden på ordet.

## Uppgift 4 – Selektioner

### Uppgift 4a

I denna uppgift ska programmet *Selections* skapas, programmets uppgift är att med hjälp av flertal selektioner avgöra ifall ett nummer en användare skickar in uppfyller ett eller flera av de definierade kraven.

Anta att vi får in ett tal, till exempel 73. Programmet behöver då först kunna avgöra att talet *är större än 67*. Om talet är *exakt 3, 6 eller 7* så ska programmet känna igen just dessa exakta värden. Skulle talet hamna någonstans *mellan 25 och 50*, till exempel 42, ska även det identifieras. På samma sätt måste tal *mellan 1 och 4 eller mellan 7 och 9* hanteras som särskilda intervall. Slutligen ska programmet dessutom kunna avgöra om talet är *mindre än noll eller större än tio*, vilket innebär att både negativa tal som  $-2$  och positiva tal som 15 omfattas. *Lämpliga meddelanden bör skrivas ut* för de respektive resultaten.

### Uppgift 4b

En liten utmaning i denna övning är att skapa ytterligare ett program, *Cakes*, vilket tillhandahålls av ett bageri. Bageriet får ofta frågor om deras tårter är jämnt delbara, specifikt med 3. Dessutom får de frågor om tårtorna inte är delbara med 4. Eftersom personalen är vidskeplig vill de även veta om det kommer in nummer där entalssiffran eller hundratalssiffran är 7. Därför har personalen önskat ett verktyg som kan ta emot input från användaren och jämföra detta med de olika kraven. *Lämpliga meddelanden bör skrivas ut* för de respektive resultaten.

Tips: Använd modulo.

## Uppgift 5 – Veckodagar

Denna uppgift fortsätter övningarna med selektioner och går ut på att ta emot indata från en användare för att sedan översätta detta till instruktioner som programmet ska köra.

### Uppgift 5a

I denna uppgift ska programmet *WeekDays* skapas. Programmet ska, *beroende på indata från användaren*, skriva ut *vilken veckodag som avses*. Måndag motsvarar 1 och de följande veckodagarna följer i ordning som vanligt.

### Uppgift 5b

Programmet *WeekDays* ska nu uppdateras. I stället för att enbart skriva ut en dag *ska alla dagar fram till helgen skrivas ut*. Anges en siffra vilket motsvarar *en helgdag skrivs enbart* den helgdagen ut. Om användaren till exempel anger siffran som motsvarar onsdag ska programmet skriva ut onsdag, torsdag och fredag.

Tips: använd ett så kallat switch-case.

**Uppgift 5c**

Slutligen ska ett nytt program med namnet Cure skapas. Programmet ska i stället för veckodagar skriva ut motsvarande rader ur The Cures "Friday I'm in Love". Om användaren anger 1 som indata ska måndag till söndag skrivas ut, och om användaren anger 4 ska torsdag till söndag skrivas ut, och så vidare. Hela texten finns nedan.

*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*

Observera att tisdag och onsdag skrivs på samma rad men ska inte vara en del av samma villkor i switch-satsen.

**Uppgift 6 – Besparing**

Att spara pengar kan vara svårt, framför allt när man inte riktigt vet hur länge man behöver spara för att nå en viss summa. Skriv därför programmet *Savings* och låt användaren bestämma hur mycket som ska sparas varje vecka och under hur många veckor. Eftersom programmet används av en generös bank får sparare möjlighet till bonus. Om den totala sparade summan når minst 5000 kr får användaren 100 kr i bonus tillsammans med meddelandet "*Du är en duktig sparare, du får en bonus på xkr!*". Om sparandet inte når 5000 kr men är minst 2500 kr får användaren 50 kr i bonus och samma meddelande. Om sparandet inte når 2500 kr får användaren endast meddelandet "*Du är ej bra på att spara!*". Oavsett om bonus ges eller inte ska programmet i slutändan skriva ut det totala beloppet inklusive eventuell bonus, tillsammans med det aktuella meddelandet. Använd lämpliga variabler för de respektive beloppen.

## Uppgift 7 – Fortsättning störst tal

Denna uppgift är en fortsättning på Uppgift 1. Programmet *ThreeCompare* ska skapas och läsa in tre tal från användaren. Programmet ska sedan avgöra om ett av talen är störst, om alla tre talen är lika stora, eller om två av talen är lika och större än det tredje talet.

Tips: Använd nedan tabell för att testa alla möjliga utfall.

Testfall	a	b	c	Förväntad resultat
Alla lika	2	2	2	Talen är lika stora.
Två första lika och störst	6	6	3	De två första talen är lika och störst.
Första och tredje lika och störst	7	5	7	Första och tredje talet är lika och störst.
Två sista lika och störst	5	9	9	Andra och sista talet är lika och störst.
Första talet störst	9	2	5	Det första talet är störst.
Andra talet störst	15	20	0	Det andra talet är störst.
Tredje talet störst	1	5	11	Det tredje talet är störst.

## Uppgift 8 – Tandläkarklinik

### Uppgift 8a

Denna uppgift bygger på uppgift 7 i Laboration 4. Ett program med namnet *DentalClinic* ska skapas utifrån den lösning som gjordes i Laboration 4. Programmets uppgift är att avgöra vilken kostnad som ska läggas till totalen. Variablerna som innehåller de olika behandlingarna representerar konstanta kostnader. Val av behandling sker via indata från användaren.

Tips: Ingen iteration behöver användas i denna uppgift; det räcker att kunna ange en behandling.

### Uppgift 8b

Denna uppgift fortsätter på lösningen i uppgift 8a. Nu ska det *läggas till en selektion* vilket avgör ifall kostnaden som debiteras *uppnår taket* för ett *konstant* avdrag å 10%. Det bör även finnas en variabel vilket innehåller den *mängd som behövs* för att nå upp till avdraget.

Om inget avdrag sker ska programmet tala om detta för användaren.

## Uppgift 9 – Sanningstabell

Sanningstabeller är mycket användbara inom programmering där logik används. Om till exempel  $a$ ,  $b$  och  $c$  är variabler i uppgiften, vet vi att antalet möjliga kombinationer är:

$$2^n \text{ där } n \text{ är antalet variabler.}$$

$$\text{Vilket i detta fall är } 2^3 = 8.$$

Detta innebär att tabellen ska innehålla åtta rader. Ett effektivt sätt att bygga tabellen är att börja med den första variabeln  $a$ , där hälften av raderna sätts till sant och hälften till falskt, och därefter följa samma mönster för de övriga variablerna. På så sätt täcks alla möjliga kombinationer av värden för variablerna. Exempel:

a	b	c
True	True	True
True	True	False
True	False	True
True	False	False
False	True	True
False	True	False
False	False	True
False	False	False

### Uppgift 9a

Använd en sanningstabell för att avgöra om uttrycket  $(!a || !b)$  är likvärdigt med  $(!(a \& \& b))$ . Ditt svar ska visa hela uppställningen av sanningstabellen, det räcker alltså inte att enbart visa om uttrycken är likvärdiga utan hur detta bevisas.

### Uppgift 9b

Använd en sanningstabell för att avgöra om uttrycket  $(!a || b)$  är likvärdigt med  $(a \& \& !b)$ . Ditt svar ska visa hela uppställningen av sanningstabellen, det räcker alltså inte att enbart visa om uttrycken är likvärdiga utan hur detta bevisas.

### Uppgift 9c

Använd en sanningstabell för att avgöra om uttrycket  $(!(a || b) || c)$  är likvärdigt med  $((!a \& \& !b) || c)$ . Ditt svar ska visa hela uppställningen av sanningstabellen, det går alltså inte att enbart visa om uttrycken är likvärdiga utan hur detta bevisas.

## Uppgift 10 – Negering

Skapa ett program med namnet Negation. Programmet ska låta användaren mata in ett sant eller falskt-värde (true eller false). Därefter ska programmet beräkna negationen av värdet och skriva ut resultatet. Om användaren till exempel anger true ska programmet skriva ut false, och om användaren anger false ska programmet skriva ut true.

Tips: använd ! vid negering.'

## Uppgift 11 – XOR REDOVISAS

Uttrycket XOR är en förkortning för "Exclusively or". Det innebär följande:

- $a \text{ XOR } b$  är sant enbart om a eller b är sant
- $a \text{ XOR } b$  är falskt om både a och b är sanna
- $a \text{ XOR } b$  är falska om både a och b är falska

För att bygga ett uttryck för XOR används ofta AND (&&) och OR (||) tillsammans, eftersom XOR inte alltid finns som en egen operator i språket.

Uppgiften är att använda en sanningstabell för att avgöra om uttrycket  $((a \parallel b) \&\& !(a \&\& b))$  motsvarar en XOR-operation eller inte. Vid redovisning ska det kunna förklaras genom att läsa av sanningstabellen om uttrycket är logiskt ekvivalent med XOR.



## Uppgift 12 – Affär REDOVISAS

Ett företag har ansökt om att få ett program byggt i Java för deras affär. De har begränsad kunskap om programmering och kravställning, men de har ritat ett aktivitetsdiagram som visar hur de önskar att programmet ska fungera. Programmet ska utföra tre olika beräkningar enligt diagrammet: rabatt för VIP-kunder, vanlig rabatt och ingen rabatt.

Kunderna har även gett ett exempel på hur de vill att användaren ska se sitt kvitto:

Antal objekt som du vill köpa är mer än 10.

Du är VIP och får VIP reducerad frakt!

Kostnaden blir 180.0 SEK



Figur 1: Kundens skiss på programmet.

Observera att oavsett vilket belopp användaren får, ska kvittot anpassas därefter. Exemplet ovan är endast ett exempel från kunden.

## Uppgift 13 – Av-nästling REDOVISAS

Selektioner kan vara mycket kraftfulla, men ibland svåra att skriva korrekt. Det händer lätt att ett krav hanteras i flera selektioner, vilket är onödigt och bör undvikas eftersom det ökar kodens komplexitet och försvårar felsökning.

I denna uppgift visas en nästlad selektion, och uppgiften är att förkorta och förenkla den nästlade selektionen i programmet Nestling.

```
3      int i = 0;
4      String s = "s";
5
6      if (i == 0 && s == "s") {
7          System.out.println(x:"0");
8      } else if(i == 0 && s == "r") {
9          System.out.println(x:"0");
10     } else if(i == 1 && s == "s") {
11         System.out.println(x:"1");
12     } else if(i == 1 && s == "r") {
13         System.out.println(x:"1");
14     } else if(i == 2 && s == "s") {
15         System.out.println(x:"2");
16     } else if(i == 2 && s == "r") {
17         System.out.println(x:"2");
18     } else {
19         System.out.println(x:"3");
20     }
21 }
22 }
```

Figur 2: Nästlade selektioner.

## Facit

### Uppgift 1 – Avgör storlek på ett tal

```

1  import java.util.Scanner;
2
3  public class AskNumber {
4      Run | Debug
5      public static void main(String[] args) {
6          Scanner sc = new Scanner(System.in);
7          System.out.print(s:"Ange ett heltal: ");
8          int askedInteger = sc.nextInt();
9          if(askedInteger > 100) {
10             System.out.println(x:"Talet är större än 100");
11         } else {
12             System.out.println(x:"Talet är högst 100");
13         }
14     }
15 }

```

Figur 3: Lösningförslag för uppgift 1.

På rad 8 används en selektion som jämför om talet från användaren är större än 100. Om villkoret är sant skrivs 'Talet är större än 100' ut (rad 9). Om villkoret är falskt körs i stället else-satsen (rad 10) och texten 'Talet är högst 100' (rad 11) skrivs ut.

### Uppgift 2 – Fråga efter ålder

```

1  import java.util.Scanner;
2
3  public class AskAge {
4      Run | Debug
5      public static void main(String[] args) {
6          Scanner sc = new Scanner(System.in);
7          System.out.print(s:"Ange ett heltal: ");
8          int askedInteger = sc.nextInt();
9          int maxChildAge = 17;
10
11         if(askedInteger <= maxChildAge) {
12             System.out.println(x:"Du är ett barn.");
13         } else {
14             System.out.println(x:"Du är vuxen.");
15         }
16     }
17 }

```

Figur 4: Lösningförslag för uppgift 2

På rad 10 används en selektion för att avgöra om talet från användaren är mindre än eller lika med 17. Om så är fallet skrivs "Du är ett barn." ut på rad 11. Om selektionens resultat är falskt går programmet automatiskt in i else-satsen på rad 12 och skriver ut "Du är vuxen." på rad 13.

### Uppgift 3 – Räkna längd på ord

```
1  import java.util.Scanner;
2
3  public class WordLength {
4      Run | Debug
5      public static void main(String[] args) {
6          Scanner sc = new Scanner(System.in);
7          String word;
8          System.out.print(s:"Ange ett ord att räkna längden på.");
9          word = sc.nextLine();
10         int wordLength = word.length();
11         System.out.printf(format:"Ordet %s är %d långt!\n", word, wordLength);
12     }
```

Figur 5: Lösningförslag för uppgift 3.

På rad 9 beräknas längden på det ord användaren angivit. Detta görs genom att anropa den inbyggda metoden `length()` i strängklassen. Resultatet returneras och sparas i variabeln `wordLength`.

## Uppgift 4 – Selektioner

### Uppgift 4a

```

1  import java.util.Scanner;
2
3  public class Selections {
4      Run | Debug
5      public static void main(String[] args) {
6          Scanner sc = new Scanner(System.in);
7          System.out.println(x:"Ange ett tal: ");
8          int input = sc.nextInt();
9
10         if(input > 67){
11             System.out.println(x:"Talet är större än 67.");
12         }
13         if (input == 3 || input == 6 || input == 7) {
14             System.out.println(x:"Talet är 3, 6 eller 7.");
15         }
16         if (input >= 25 && input <= 50) {
17             System.out.println(x:"Talet är mellan 25 och 50.");
18         }
19         if ((input >= 1 && input <= 4) || (input >= 7 && input <= 9) ) {
20             System.out.println(x:"Talet är mellan 1 och 4 eller 7 och 9.");
21         }
22         if (input < 0 || input > 10) {
23             System.out.println(x:"Talet är mindre än 0 eller större än 10.");
24         }
25     }
26 }

```

Figur 6: Lösningsförslag för uppgift 4a.

De olika selektionerna är oberoende av varandra eftersom enskilda if-satser används. Detta innebär att flera villkor kan utvärderas som sanna samtidigt. Om användaren till exempel anger värdet 3 kommer både utskriften "Talet är 3, 6 eller 7." och "Talet är mellan 1 och 4 eller 7 och 9." att visas. Detta beror på att villkoren i koden på rad 12 respektive rad 18 båda uppfylls.

## Uppgift 4b

```

1  import java.util.Scanner;
2
3  public class Cakes {
4      Run | Debug
5      public static void main(String[] args) {
6          Scanner sc = new Scanner(System.in);
7          System.out.print(s:"Ange ett tal: ");
8          int input = sc.nextInt();
9
10         if (input % 3 == 0) {
11             System.out.println(x:"Tårtan är delbar med 3.");
12         }
13         if (input % 4 != 0) {
14             System.out.println(x:"Tårtan är inte delbar med 4.");
15         }
16         if (input % 10 == 7) {
17             System.out.println(x:"Entalssiffran är 7.");
18         }
19         if ((input/100) % 10 == 7) {
20             System.out.println(x:"Hundratalssiffran är 7.");
21         }
22     }
23 }

```

Figur 7: Lösningförslag för uppgift 4b.

$\text{input} \% 3 == 0$  (rad 9)

Modulo 3 ger resten när talet delas med 3. Om resten är 0 betyder det att talet är jämnt delbart med 3. Exempel:  $9 \% 3 = 0$ ,  $10 \% 3 = 1$ .

$\text{input} \% 4 != 0$  (rad 12)

Modulo 4 ger resten när talet delas med 4. Om resten inte är 0 betyder det att talet inte är delbart med 4. Exempel:  $7 \% 4 = 3 \rightarrow$  inte delbart,  $8 \% 4 = 0 \rightarrow$  delbart.

$\text{input} \% 10 == 7$  (rad 15)

Modulo 10 ger alltid den sista siffran (entalssiffran) av talet. Om den sista siffran är 7, uppfylls villkoret. Exempel:  $17 \% 10 = 7$ ,  $123 \% 10 = 3$ .

$(\text{input} / 100) \% 10 == 7$  (rad 18)

Här delas först talet med 100, vilket tar bort de två sista siffrorna (entals- och tiotalssiffran), och sedan modulo 10 för att ta den nya sista siffran, alltså hundratalssiffran. Exempel:  $712 / 100 = 7$ ,  $7 \% 10 = 7 \rightarrow$  hundratalssiffran är 7.

## Uppgift 5 – Veckodagar

### Uppgift 5a

```
1  import java.util.Scanner;
2
3  public class WeekDays {
4      Run | Debug
5      public static void main(String[] args) {
6          Scanner sc = new Scanner(System.in);
7          System.out.print(s:"Ange en veckodag, måndag är första dagen och börjar på 1: ");
8          int weekday = sc.nextInt();
9          if(weekday == 1) {
10             System.out.println(x:"Idag är det måndag.");
11         } else if(weekday == 2) {
12             System.out.println(x:"Idag är det tisdag.");
13         } else if(weekday == 3) {
14             System.out.println(x:"Idag är det onsdag.");
15         } else if(weekday == 4) {
16             System.out.println(x:"Idag är det torsdag.");
17         } else if(weekday == 5) {
18             System.out.println(x:"Idag är det fredag.");
19         } else if(weekday == 6) {
20             System.out.println(x:"Idag är det lördag.");
21         } else if(weekday == 7) {
22             System.out.println(x:"Idag är det söndag.");
23         }
24     }
```

Figur 8: Lösningförslag för uppgift 5a.

Ett enkelt block av if- och else if-satser gör att lösningen skriver ut den veckodag som motsvarar den siffra som anges, från 1 till 7.

## Uppgift 5b

```

1  import java.util.Scanner;
2
3  public class WeekDays {
4      public static void main(String[] args) {
5          Scanner sc = new Scanner(System.in);
6          System.out.print(s:"Ange en veckodag, måndag är första dagen och börjar på 1: ");
7          int weekday = sc.nextInt();
8          switch (weekday) {
9              case 1:
10                 System.out.println(x:"Måndag");
11             case 2:
12                 System.out.println(x:"Tisdag");
13             case 3:
14                 System.out.println(x:"Onsdag");
15             case 4:
16                 System.out.println(x:"Torsdag");
17             case 5:
18                 System.out.println(x:"Fredag");
19                 break;
20             case 6:
21                 System.out.println(x:"Lördag");
22                 break;
23             case 7:
24                 System.out.println(x:"Söndag");
25                 break;
26             }
27         }
28     }

```

Figur 9: Lösningförslag för uppgift 5b.

Lösningen bygger på att använda en switch-sats. Beroende på vilken siffra som anges skrivs alla dagar ut från och med den motsvarande dagen fram till att ett break påträffas. break markerar slutet på den aktuella grenen i switchen, vilket gör att inga fler utskrifter sker därefter. Självfallet hade det gått att använda if- och else if-satser, men det hade blivit väldigt repetitivt.



## Uppgift 5c

```

1  import java.util.Scanner;
2
3  public class Cure {
4      Run | Debug
5      public static void main(String[] args) {
6          Scanner sc = new Scanner(System.in);
7          System.out.print(s:"Lyrics to Friday I'm in love, starting from assigned day: ");
8          int weekday = sc.nextInt();
9          switch (weekday) {
10             case 1:
11                 System.out.println(x:"Monday, you can fall apart");
12             case 2:
13                 System.out.print(s:"Tuesday, ");
14             case 3:
15                 System.out.println(x:"Wednesday, break my heart");
16             case 4:
17                 System.out.println(x:"Oh, Thursday doesn't even start");
18             case 5:
19                 System.out.println(x:"It's Friday, I'm in love");
20             case 6:
21                 System.out.println(x:"Saturday, wait");
22             case 7:
23                 System.out.println(x:"And Sunday always comes too late");
24                 break;
25             }
26     }
27 }

```

Figur 10: Lösningsförslag för Uppgift 5c.

Lösningen bygger på en switch-sats liknande uppgift 5b, men med färre break. Att tisdag och onsdag skrivs ut på samma rad, trots att de finns i olika case, beror på att tisdag skrivs ut med print och inte println.

## Uppgift 6 – Besparing

```

1  import java.util.Scanner;
2
3  public class Savings {
4      public static void main(String[] args) {
5          Scanner sc = new Scanner(System.in);
6          double weeklySavings;
7          double amountOfWeeks;
8          final int amazingSaver = 100;
9          final int goodSaver = 50;
10         double saved;
11         double total;
12
13         System.out.println(x:"Välkommen till sparningsprogrammet!");
14         System.out.print(s:"Ange antalet att spara i veckan: ");
15         weeklySavings = sc.nextDouble();
16         System.out.print(s:"Ange hur många veckor sparandet sker: ");
17         amountOfWeeks = sc.nextDouble();
18
19         saved = weeklySavings * amountOfWeeks;
20         if(saved >= 5000) {
21             System.out.println(x:"Du är en duktig sparare, du får en bonus på 100kr!");
22             total = saved + amazingSaver;
23         } else if(saved >= 2500 && saved < 5000) {
24             System.out.println(x:"Du är en duktig sparare, du får en bonus på 50kr!");
25             total = saved + goodSaver;
26         } else {
27             System.out.println(x:"Du är ej bra på att spara!");
28             total = saved;
29         }
30         System.out.printf(format:"Du har sparat %.2f under loppet av %.1f veckor!", total, amountOfWeeks);
31     }
32 }

```

Figur 11: Lösningförslag för uppgift 6.

Lösningen bygger på att både sparbeloppet per vecka och antalet veckor kan vara flyttal, det vill säga de behöver inte vara jämna tal. De enda värden som är konstanter är bonusarna, eftersom de inte ska ändras under körningen.

## Uppgift 7 – Fortsättning störst tal

```

1  import java.util.Scanner;
2
3  public class ThreeCompare {
4      Run | Debug
5      public static void main(String[] args) {
6          Scanner sc = new Scanner(System.in);
7
8          System.out.print(s:"Ange det första talet: ");
9          int a = sc.nextInt();
10         System.out.print(s:"Ange det andra talet: ");
11         int b = sc.nextInt();
12         System.out.print(s:"Ange det tredje talet: ");
13         int c = sc.nextInt();
14
15         if (a == b && b == c) {
16             System.out.println(x:"Talen är lika stora.");
17         } else if (a == b && a > c) {
18             System.out.println(x:"De två första talen är lika och störst.");
19         } else if (a == c && a > b) {
20             System.out.println(x:"Första och tredje talet är lika och störst.");
21         } else if (b == c && b > a) {
22             System.out.println(x:"Andra och sista talet är lika och störst.");
23         } else if (a > b && a > c) {
24             System.out.println(x:"Det första talet är störst.");
25         } else if (b > a && b > c) {
26             System.out.println(x:"Det andra talet är störst.");
27         } else {
28             System.out.println(x:"Det tredje talet är störst.");
29         }
30     }
31 }

```

Figur 12: Lösningsförslag för uppgift 7.

Varje villkor i programmet utvärderas med hjälp av selektioner. else if-satser används efter den första if-satsen för att säkerställa att endast ett alternativ skrivs ut.

## Uppgift 8 – Tandläkarklinik

### Uppgift 8a

```

1  import java.util.Scanner;
2
3  public class DentalClinic {
4      Run | Debug
5      public static void main(String[] args) {
6          Scanner sc = new Scanner(System.in);
7          int option = 0;
8          double cost = 0;
9          final double checkup = 600;
10         final double cleaning = 30000;
11         final double cavityRepair = 1500;
12         final int achievedDiscount = 3000;
13         final double discount = 0.9;
14
15         System.out.printf(format:"Välkommen till tandläkarkliniken!\nVälj:%n1 för kontroll%n2 för rengöring%n3 för lagning av hål.%n");
16         option = sc.nextInt();
17
18         switch (option) {
19             case 1:
20                 cost += checkup;
21                 System.out.println(x:"Lade till kontrollkostnad till total.");
22                 break;
23             case 2:
24                 cost += cleaning;
25                 System.out.println(x:"Lade till rengöringskostnad till total.");
26                 break;
27             case 3:
28                 cost += cavityRepair;
29                 System.out.println(x:"Lade till lagning av hål till kostnad.");
30                 break;
31         }
32     }
33 }

```

Figur 13: Lösningförslag för uppgift 8a.

Lösningen använder en switch-sats för att hantera de olika alternativen och addera rätt belopp till kostnaden. Konstanterna representerar värden som aldrig ska ändras under körning. Observera att bilden ovan inte visar hela koden.

## Uppgift 8b

```

1  import java.util.Scanner;
2
3  public class DentalClinic {
4      Run | Debug
5      public static void main(String[] args) {
6          Scanner sc = new Scanner(System.in);
7          int option = 0;
8          double cost = 0;
9          final double checkup = 600;
10         final double cleaning = 30000;
11         final double cavityRepair = 1500;
12         final int achievedDiscount = 3000;
13         final double discount = 0.9;
14
15         System.out.printf(format:"Välkommen till tandläkarkliniken!\nVälj:%n1 för kontroll%n2 för rengöring%n3 för lagning av hål.%n");
16         option = sc.nextInt();
17
18         switch (option) {
19             case 1:
20                 cost += checkup;
21                 System.out.println(x:"Lade till kontrollkostnad till total.");
22                 break;
23             case 2:
24                 cost += cleaning;
25                 System.out.println(x:"Lade till rengöringskostnad till total.");
26                 break;
27             case 3:
28                 cost += cavityRepair;
29                 System.out.println(x:"Lade till lagning av hål till kostnad.");
30                 break;
31         }
32         if (cost >= achievedDiscount) {
33             cost = cost*discount;
34             System.out.printf(format:"Kostnaden för besöket blev %.2fkr efter rabatten på 10%%", cost);
35         } else {
36             System.out.printf(format:"Kostnaden för besöket blev %.2fkr", cost);
37         }
38     }
39 }

```

Figur 14: Lösningsförslag för uppgift 8b.

Denna lösning omfattar hela 8a och bygger vidare med tillägg för hur rabatten ska hanteras. Rabatten beräknas i if-satsen på rad 31 genom att multiplicera den nuvarande kostnaden med 0,9, vilket motsvarar att 10 % dras av från totalen. Resultatet sparas sedan i cost och skrivs ut till användaren. Observera även att variabeln cleaning har satts till ett högt värde för att kunna trigga if-satsen.

## Uppgift 9 – Sanningstabell

### Uppgift 9a

Är uttrycket  $(!a || !b)$  är likvärdigt med  $(!(a \&\&b))$ ?

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

Uttrycken är lika! Om uttrycken !a, !b eller a&&b hade utelämnats skulle uppgiften inte kunna besvaras korrekt, eftersom dessa uttryck är grundläggande för att kunna jämföra och negera motsvarigheterna.

### Uppgift 9b

Är uttrycket  $(!a || b)$  är likvärdigt med  $(a \&\&!b)$ ?

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 likavärdiga! Om uttrycken !a och !b hade utelämnats skulle uppgiften inte kunna besvaras korrekt, eftersom dessa uttryck är grundläggande för att kunna jämföra och negera motsvarigheterna.

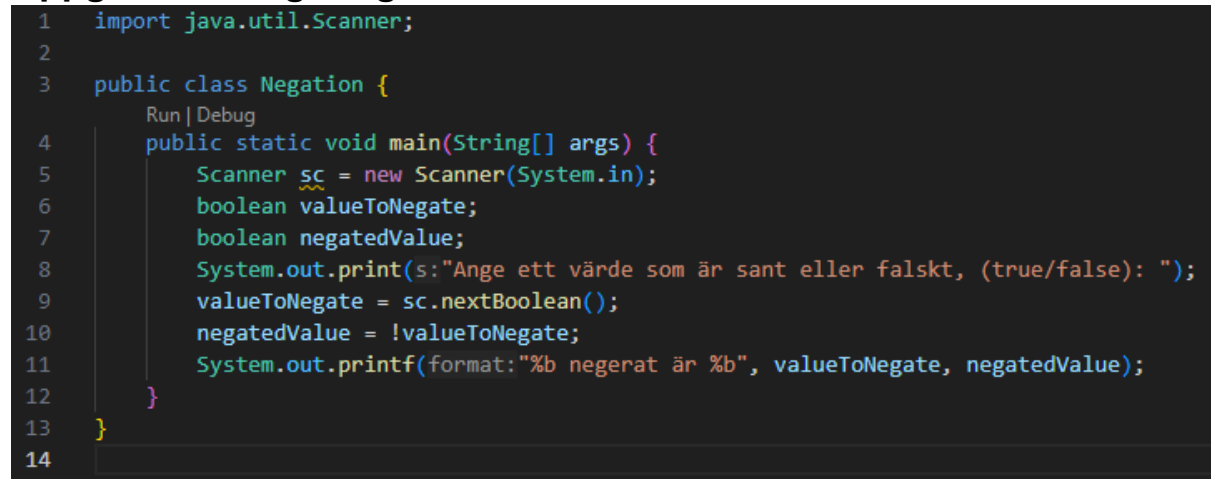
### Uppgift 9c

Är uttrycket  $(!(a || b) || c)$  är likvärdigt med  $((!a \&\&!b) || c)$ ?

a	b	c	!a	!b	a  b	!(a  b)	!(a  b)  c	!a&&!b	(!a&&!b)  c
T	T	T	F	F	T	F	T	F	T
T	T	F	F	F	T	F	F	F	F
T	F	T	F	T	T	F	T	F	T
T	F	F	F	T	T	F	F	F	F
F	T	T	T	F	T	F	T	F	T
F	T	F	T	F	T	F	F	F	F
F	F	T	T	T	F	T	T	T	T
F	F	F	T	T	F	T	T	T	T

Uttrycken är lika! Om uttrycken !a, !b, a||c, !(a||b) och !a&&!b hade utelämnats skulle uppgiften inte kunna besvaras korrekt, eftersom dessa uttryck är grundläggande för att kunna jämföra och negera motsvarigheterna.

## Uppgift 10 – Negering



```
1 import java.util.Scanner;
2
3 public class Negation {
4     Run | Debug
5     public static void main(String[] args) {
6         Scanner sc = new Scanner(System.in);
7         boolean valueToNegate;
8         boolean negatedValue;
9         System.out.print(s:"Ange ett värde som är sant eller falskt, (true/false): ");
10        valueToNegate = sc.nextBoolean();
11        negatedValue = !valueToNegate;
12        System.out.printf(format:"%b negerat är %b", valueToNegate, negatedValue);
13    }
14 }
```

Figur 15: Lösningsförslag för uppgift 10.

Genom att använda operatoren ! framför en boolesk variabel negeras värdet, det vill säga det omvandlas till motsatsen. I detta fall sparas resultatet i variabeln negatedValue.