

DA339A Laboration L6

Syfte

Den här laborationens syfte är att öva på att använda iteration som verktyg vid problemlösning och att skriva kod som innehåller iterationer. I detta ingår att kunna beskriva en lösning med pseudokod, rita ett aktivitetsdiagram och att implementera lösningar med iteration i kod och exekvera denna.

Redovisning

Laboration L6 ska inte redovisas i sig självt. Laborationen innehåller dock uppgifter som kan komma att användas och utökas i senare laborationer som i sin tur ska redovisas och i inlämningsuppgifter.

Förberedelser

Den här laborationen är direkt kopplad till det innehåll som finns i föreläsning F7 och F8. Vissa av uppgifterna i laboration L6 förbereddes i L4 som lösningar i pseudokod och/eller aktivitetsdiagram samt att delar av kod från laboration L5 kan används som del av lösningar i L6.

För laboration L6 förutsätts du även behärska tidigare genomgången innehåll på kursen.

Du behöver ladda ner filen `da339a_L6_filer.zip` och packa upp denna inför laborationen då den innehåller filer som du ska utgå ifrån för vissa av uppgifterna. I zip-filen finns en fil *NameOfProgram.java* med ett "skelett" till ett program som du kan använda när du ska skapa egna filer till dina program.

Om extra övningar

Vissa uppgifter innehåller en liten uppmaning om extra övningar. Vissa av dessa är en utmaning för de som vill ha något lite svårare att göra. Vissa uppgifter är mer en uppmaning att testa olika sätt att lösa en uppgift eller att göra utökningar av uppgiften. Finner du uppgifterna i sig själva tillräckligt utmanande så gör inte de extra övningarna med en gång. Spara då dessa som extra kod att experimentera med och övning till när du löst alla de övriga uppgifterna. Kom ihåg - programmera och skriva kod är ett hantverk och övning ger färdighet! Experimentera och lek runt och testa olika idéer i din kod när du gjort de regelrätta uppgifterna.

Uppgifter

Uppgift 1a

Skriv ett program som använder 5 stycken for-loopar (efter varandra – inte nästlade) för att skriva ut heltal så att du får följande utskrifter:

For-loop 1:

Heltal mellan 1-10 i stigande ordning: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

For-loop 2:

Heltal mellan 1-10 i fallande ordning: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

For-loop 3:

Jämna heltal mellan 1-10 i stigande ordning: 2, 4, 6, 8, 10

For-loop 4:

Udda heltal mellan 1-10 i fallande ordning: 9, 7, 5, 3, 1

For-loop 5:

Jämna 10-tal mellan 1-100: 10, 20, 30, 40, 50, 60, 70, 80, 90, 100

Uppgift 1b

Skriv ett program som ger samma resultat som i 1a men som använder while-loopar istället för for-loopar och ersätt for-loop i texten som skrivs ut med while-loop.

Uppgift 1c

Skriv ett program som ger samma resultat som i 1a men som använder do-while-loopar istället för for-loopar och ersätt for-loop i texten med do-while-loop.

Uppgift 2

Utgår från filen *L6Upp2.java*. Denna ger dig en utskrift enligt:

10 13 16 19 22 25

Komplettera programmet med en while-loop och en do-while-loop som gör samma sak som for-loopen i programmet så att du får följande utskrift:

10 13 16 19 22 25

10 13 16 19 22 25

10 13 16 19 22 25

Uppgift 3a

Utgå från uppgift L4 Uppgift 3a och skapa ett program som med hjälp av en for-loop skriver ut 3ans multiplikationstabell med utskrift enligt:

1*3=3

2*3=6

3*3=9

...

10*3=30

Uppgift 3b

Utgå från uppgift L4 Uppgift 3b och skapa ett program som frågar användaren vilken multiplikationstabell som ska användas och sedan skriver ut detta val på samma sätt som vi skrev ut 3ans tabell i uppgift 3a ovan.

Uppgift 5b

Ändra programmet i 5a så du gör motsvarande tabell för Fahrenheit till Celsius. Formeln är då:
 $C = (F - 32) * 5 / 9$

Extra övning: Använd `System.out.printf()` för att formatera utskriften så den blir bättre formaterad. Se Deitel kapitel 2.4 för ledning.

Uppgift 6

Det matematiska begreppet ! (fakultet, factorial på engelska) innebär produkten av alla heltal större än 0 från 1 och upp till talet självt. Så $4! = 1 * 2 * 3 * 4 = 24$. Skriv ett program som ber användaren om ett heltal som man ska räkna ut fakulteten för och använd en for-loop för att räkna ut vad fakulteten blir och visa uträkningen. Utskriften ska få formen:

```
Ange tal för fakultet: 4
4! = 1*2*3*4=24
```

Ledning: Använd inte `System.out.println("utskrift")` för att skriva ut allt då det ger en radbrytning. Använd `System.out.print("utskrift")` som inte ger radbrytning så du kan fortsätta skriva ut på samma rad. Produkten för fakulteten skrivs ut när for-loopen är klar.

Var uppmärksam på vad som händer om du multiplicerar något med 0.

Uppgift 7a

Utgå från L4 Uppgift 4 och skriv ett program som använder två nästlade for-loopar för att skriva ut en matris med multiplikationstabellerna 1-10. Du ska få en utskrift enligt:

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
...
10 20 30 40 50 60 70 80 90 100
```

Extra övning: Som du ser så behöver matrisen inte vara snyggt formaterad med jämna kolumner. Vill du ha en utmaning så kan du fundera på hur du kan göra en snyggare formaterad utskrift. Det finns många olika sätt att lösa detta med att använda olika tecken som skrivs ut, eller räkna tecken och fylla på blanksteg till att nyttja bibliotek i Java och olika sätt att skriva ut och formatera utskriften – nyttja kurslitteraturen eller börja slipa på din google-fu (<https://en.wiktionary.org/wiki/Google-fu>) för att lösa programmeringsproblem.

Uppgift 7b

Modifiera ditt program i 7a så att du läser in två heltal från användaren som anger ett startvärde och ett slutvärde som ger ett intervall som ska skrivas ut, exempelvis så ska man skriva ut tabeller från 4 till och med 7:

```
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
```

Vad händer om användaren matar in ett startvärde som är större än slutvärdet?

Extra övning: Som en extra övning för den som vill ha lite mer att jobba med: Lös det så att även om användaren matar in ett startvärde som är större än slutvärdet så kommer intervallet att skrivas ut rätt med start på det minsta värdet och till det största värdet. Ledning: Du vet från L4 och L5 hur du avgör vilket tal som är det största av två tal - när du vet det vad kan du göra då med de två inlästa talen du har?

Uppgift 8

Utgå från din lösning i L4 Uppgift 6 och filen *GuessTheNumber.java*. Börja med att titta på filen *GuessTheNumber.java* och bekanta dig med koden för att generera ett slumptal. Kompilera och kör programmet och se vilka slumptal du får. Prova att ändra intervallet för slumptalet (instruktioner för detta i kommentarer i källkoden).

Utöka sedan programmet så att du implementerar kod för spelet att gissa ett heltal enligt modellen: Datorn slumpar fram ett tal. Användaren anger sedan ett tal som hen gissar är det rätta. Programmet ska svara användaren om hen gissat rätt eller om det tal man gissat är större eller mindre än det rätta talet. Man håller på tills användaren gissat rätt tal.

Ledning: Du behöver en while-loop eller en do-while-loop som pågår så länge användaren inte gissat rätt tal och om fel tal gissats läsa in ett nytt tal från användaren. Inuti loopen behöver du använda en if-sats för att ge rätt respons till användaren beroende på om hen gissar ett tal som är större eller mindre än det rätta talet.

Bryt ner din lösning i flera olika steg och försök inte göra allt på en gång. En bra ordning att göra saker i:

- Lös först hur du läser in ett tal från användaren.
- Skriv en if-sats som anger om det inlästa talet är större än eller mindre än eller lika stort som det slumpade talet. Gå inte vidare förrän du fått rätt på detta.
- Modifiera den tidigare if-satsen och lägg till loopen för att gå tillbaka och läsa in ett tal till och jämför detta med det slumpade talet och skriv ut om det är större eller mindre.

Extra övning: Lös uppgiften en gång med en while-loop och en gång med en do-while-loop. Tekniskt sätt skulle vi kunna skriva en lösning med en for-loop men det här är ett typiskt exempel på när en while-loop eller do-while är lämpligare än en for-loop.

Uppgift 9

Utgå från L4 Uppgift 8 och skriv ett program som implementerar lösningen för spelet att ange det största talet i ett antal rundor. Vi får låtsas som att användarna inte ser vad den andra användaren skriver in i nuläget.

Öva på att bryta ner problemet i mindre delar när du ska implementera koden. Implementera en bit i taget – testa att koden fungerar som du tänkt. När det fungerar som tänkt gå vidare och gör en liten bit till – testa igen. När det fungerar som tänkt gå vidare med nästa delar av lösningen - testa igen, och så vidare till du implementerat alla delar av lösningen.

En bra ordning att göra saker i kan vara:

- Gör delen som läser in tal och svarar på om talen är lika eller vem som vann.
- Gör loopen så att delen ovan upprepas ett antal gånger. Skriv ut variabler med värden så som antal rundor och räknare så du ser vilken runda det är och kan kontrollera att koden fungerar som du tänkt.
- Lägg till delen efter loopen som anger vem som vann eller om matchen blev oavgjord.

Extra övning: I stället för att låta användaren ange tal - låt datorn slumpa fram tal och simulera en spelomgång mellan två användare. Du kan kopiera kod från filen/lösningen till uppgift 5 ovan för att få fram slumpstal. Observera att du då även behöver raden

```
import java.util.Random;
```

som står i början av filen.

Uppgift 10

Utgå från L4 Uppgift 7 och din pseudokod-lösning för att skriva ett program för en reception på en tandläkarklinik. Du ska ha delar av lösningen redan från L5 Uppgift 8b. Lägg till den sista biten av lösningen som innehåller en loop så man kan ange kostnad för flera olika behandlingar innan eventuella avdrag görs på kostnaden.