



Java

TDDC77

Objektorienterad Programmering

Föreläsning 3

Sahand Sadjadee
IDA, Linköpings Universitet

Outline

- ❑ Att styra programflödet
 - ❑ If (4.1, 4.2, 4.5)
 - ❑ While (5.1)
 - ❑ for (5.4)
 - ❑ do-while (5.7)
 - ❑ break/continue (5.2)
 - ❑ switch(4.7)
 - ❑ “Ternary” operatorn
 - ❑ Nästling (6.2, 6.3)

Att styra programflödet

Ordning

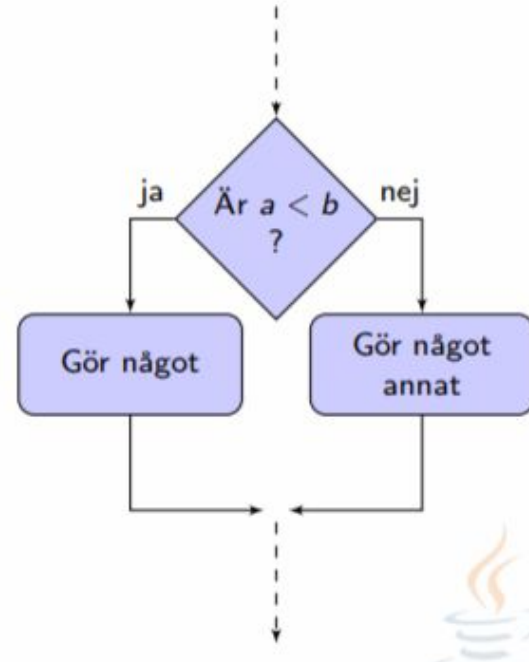
- Normalt sett körs programmets kodrader “i ordning”, alltså uppifrån och ned.
- Ibland vill man ändra på detta, exempelvis när man vill jämföra två heltal.

kontrollstrukturer

- Istället för att följa den “normala ordningen”, kan man hoppa över rader.
- I Java får man INTE göra hopp hur som helst
- Man använder istället färdiga kontrollstrukturer som ändrar ordningen beroende på något villkor.

if-else satsen

```
if (a < b) {  
    gör något;  
} else {  
    gör något annat;  
}
```



if-else satsen

```
if (villkor)
    //En singel instruktion som körs när villkoret är true.
else //valfritt
    //En singel instruktion som körs när villkoret är falskt.
```

- Villkoret kan vara en boolsk literal, variabel eller uttryck.
- För att köra flera instruktioner ska block användas.
- Att ha `else` är valfritt.
- `if` kallas för “single-selection” kontrollstruktur
- `if-else` kallas för “double-selection” kontrollstruktur.

Block

```
{  
  Instruktion 01  
  Instruktion 02  
  Instruktion 03  
  .  
  .  
  .  
  Instruktion n  
}
```

- Ett block är ett antal instruktioner som ligger mellan ett par måsvingar.
- Ett block ses som en singel instruktion av kompilatorn.
- Block kan användas för att ha flera instruktioner i kontrollstrukturer.

if-else satsen

```
if (villkor){
```

```
    //instruktionerna som ligger i blocket körs om villkoret är true.
```

```
}else{
```

```
    //instruktionerna som ligger i else-blocket körs om villkoret är false.
```

```
}
```

Att programmera är att bryta ner en uppgift i små steg som en dator kan göra

- Låt användaren skriva in två heltal x och y . Vi vill räkna upp skillnadens absoluta värde (dvs, $|x-y|$) och skriva ut resultatet.
 - Fråga användaren efter ett heltal
 - Läs in ett heltalet x
 - Fråga användaren efter ett heltal
 - Läs in ett heltalet y
 - Om x är större än y , räkna upp $(x - y)$
 - Om y är större än x , räkna upp $(y - x)$
 - Skriv ut resultatet

```

/* AbsoluteSkillnad.java
 * Programmet demonstrerar användandet av
 * if instruktioner
 */
import java.util.Scanner;
class AbsolutSkillnad
{
    /* Metoden läser in två heltal från användaren
     * och skriver ut skillnaden mellan det större och det
     * minsta talet
     */
    public static void main (String[] args)
    {
        Scanner in = new Scanner (System.in);
        int x, y, tmp;

        System.out.println ("Mata in ett heltal x: ");
        x = in.nextInt();
        System.out.println ("Mata in ett heltal y: ");
        y = in.nextInt();

        if (y > x){
            tmp = x;
            x = y;
            y = tmp;
        }

        System.out.println ("|x-y| = " + (x-y));
    }
}

```



if-else satsen (exempel)

```
/* Systemet.java
 * Programmet demonstrera användandet av
 * if-then-else instruktioner
 */
import java.util.Scanner;
class Systemet{

    //Fråga användaren om sin åld och skriver ut om användaren får
    static public void main(String[] args){

        final int MIN_LIMIT=20, LEG_LIMIT=25;
        Scanner scan = new Scanner(System.in);

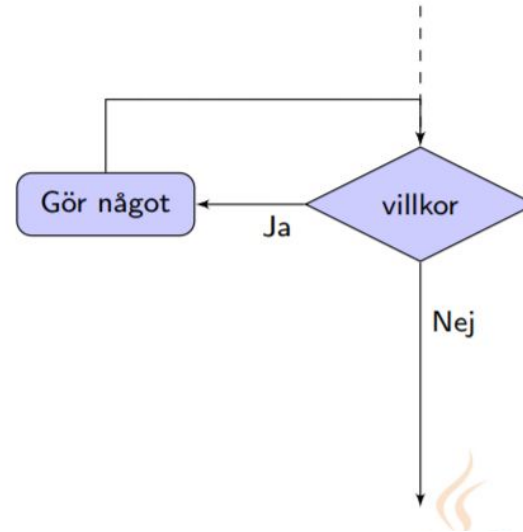
        System.out.println("Hur gammal är du ?");
        int age = scan.nextInt();

        if(age>=MIN_LIMIT){
            if(age<= LEG_LIMIT){
                System.out.println("Du får köpa från Systemet"
                    + " men du måste visa leg!");
            }else{
                System.out.println("Du får köpa från Systemet!");
            }
            System.out.println("Du får köpa från Systemet!");
        }else{
            System.out.println("Du får inte köpa från Systemet!");
        }
    }
}
```



while slingor

```
while (a < b) {  
    gör något;  
}
```



while

```
while (villkor)
```

```
    //En singel instruktion som körs när villkoret är true.
```

- Villkoret kan vara en boolsk **literal**, **variabel** eller **uttryck**.
- Instruktionen körs så länge villkoret är `true`.
- En **sentinel-controlled** kontrollstruktur.
- För att köra flera instruktioner ska block användas.

while: kontrollera inputen - ett exempel

Skriv ett program som tar emot ett heltal från användaren. Om inmatade värdet är mellan 0-10 då ska programmet fortsätta annars frågar om användare att mata in ett annat värde igen.

while: kontrollera inputen

```
/* Constrained.java
 * Programmet illustrerar hur man kan begränsa
 * vad användaren mata in
 */
import java.util.Scanner;
public class Constrained{

    /*Fortsätt att be användaren om att mata in
    * ett tal tills den är mellan 0 och 10
    */
    static public void main(String[] args){
        Scanner scan = new Scanner(System.in);
        int input;
        final int MIN=0, MAX=10;

        System.out.println("please enter a number between "
            + MIN + " and " + MAX);
        input = scan.nextInt();
        while(input<MIN || MAX<input){
            System.out.println("please enter a number between "
                + MIN + " and " + MAX);
            input = scan.nextInt();
        }

        System.out.println("Thank you. You entered: " + input);
    }
}
```



while: terminering (termination)

```
/* Countdown.java
 * Illustrerar hur ett programmet terminerar inte
 */
public class Countdown{

    //Exekvera en slinga som terminerar inte
    static public void main(String[] args){
        int count=2;
        while(count!=0){
            System.out.println(count + "...");
            count -=5;
        }
    }
}
```

while: nested loops

```
/* Palindrome.java
 * Illustrerar användandet av nästlad slingor
 */
import java.util.Scanner;
public class Naestlad{
    /* Upprepa att be om en text och att kolla upp om
     * det är ett palindrome
     */
    static public void main(String[] args){
        String candidate, answer="y";
        Scanner scan = new Scanner(System.in);

        while(answer.equals("y")){
            System.out.println("Please enter a candidate: ");
            candidate = scan.nextLine();

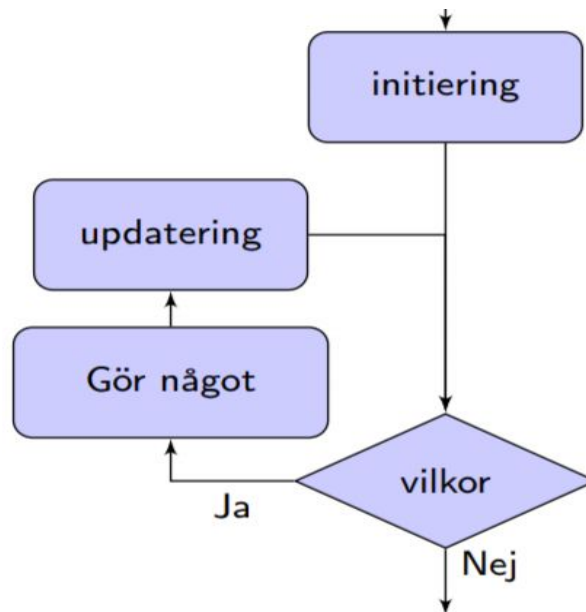
            int index = 0;
            boolean isPalindrome = true;
            while(index < (candidate.length() / 2)){
                if(candidate.charAt(index) != candidate.charAt(candidate.length()-index-1))
                    isPalindrome = false;
                index++;
            }
            if(isPalindrome)
                System.out.println(candidate + " is a palindrome");
            else
                System.out.println(candidate + " is not a palindrome");

            System.out.println("Repeat? y/n");
            answer = scan.nextLine();
        }
        System.out.println("Good bye!");
    }
}
```



for slingor

```
final int MAX = 10;  
  
for(int i=0; i<MAX; i++){  
    System.out.println(i);  
}
```



for

```
for (initiering; villkoret ; uppdatering )  
    //En singel instruktion som körs när villkoret är true.
```

- Initiering: deklaration av en eller flera variabel(räknare)
- Villkoret kan vara en boolsk literal, variabel eller uttryck.
- Uppdatering: räknarens värde uppdateras.
- Instruktionen körs så länge villkoret är true.
- En counter-controlled kontrollstruktur.
- För att köra flera instruktioner ska block användas.

for (fort.)

```
/* Multiplikation.java
 * Illustrerar användandet av while och for slingor
 */
import java.util.Scanner;
public class Multiplikation{

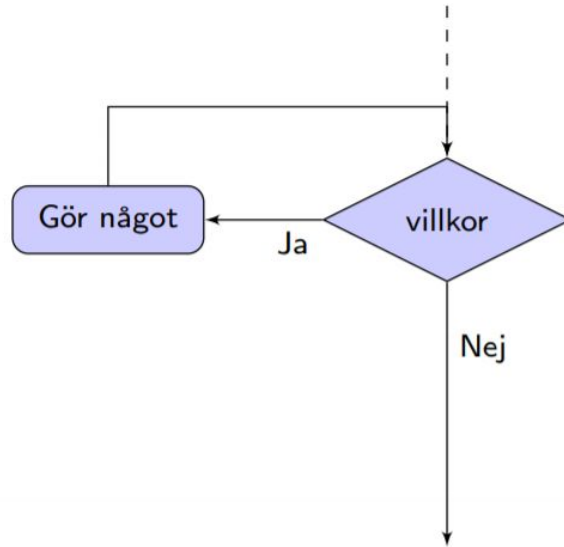
    //Be om ett tal mellan 0 och 9 och skriv ut multiplikationstabellen för talet.
    static public void main(String[] args){
        Scanner scan = new Scanner(System.in);
        int number=0;

        while(number<=0 || 10<number){
            System.out.println("Please enter an integer between 1 and 10?");
            number = scan.nextInt();
        }

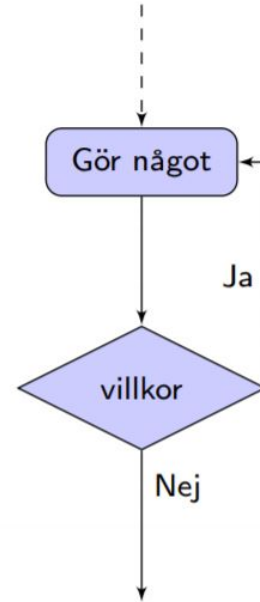
        for(int k=0; k<=10; k++)
            System.out.println(number + " x " + k + " = " + number*k);
    }
}
```



do-while slingor



While loop



Do loop

do-while

- Liksom while-loopen fast den kör alltid minst ett varv.
- Villkoret valideras efter varje varv.

```
int tal = 0;

do{
    System.out.print("Ange ett tall mellan " + "3 och 26: ");
    tal = in.nextInt();
} while((tal < 3)|| (tal > 26))
```

break/continue

- Om man exekverar `break` i en slinga, kommer exekveringen av slingan att stoppas. Instruktionen som följer slingan körs närmast.
- Om man exekverar `continue` i en slinga kommer kontrollen att hoppa till början av slingan. Villkoret valideras om innan första raden körs.
- Det är alltid möjligt att få samma effekt utan att använda sig av `break` och `continue`. Att använda `break/continue` anses som en DÅLIG PRAXIS som de skadar läsbarhet.

switch satsen

```
/* Betyg.java
 * Programmet illustrera användandet av switch instruktion
 */
import java.util.Scanner;
public class Betyg{
    // läser in ett betyg och översätta den till text
    static public void main(String[] args){
        Scanner scan = new Scanner(System.in);
        int grade;

        do{
            System.out.println("Ge ett betyg mellan 3 och 5: ");
            grade = scan.nextInt();
        }while(grade<3 || 5<grade);

        switch(grade){
            case 3:
                System.out.println("Godkänd!");
                break;
            case 4:
                System.out.println("Väl godkänd!");
                break;
            case 5:
                System.out.println("Mycket väl godkänd!");
                break;
            default:
                System.out.println("Jag ser inte hur vi har lyckats komma hit!");
        }
    }
}
```



switch satsen

- Evaluerar en variabel eller uttryck av typerna `char`, `byte`, `short`, `int` och `String`.
- jämför det resulterande värdet med alla konstanter en för en nedåt.
- Kör första `case : t` som matchar värdet.
- Om inget `case` matchar värdet, kör `default` automatiskt.
- Om det finns inget `default`, hoppar den till första instruktionen som följer `switch`.
- Varje `case` avslutas med `break`.

“Ternary” operatör

```
/* Villkorlig.java
 * Illustrerar användandet av den villkorlig sats
 */
import java.util.Scanner;
public class Villkorlig{
    //läser in två tal och skriver ut vilket är större
    static public void main(String[] args){
        int x,y,max;
        Scanner scan = new Scanner(System.in);

        System.out.print("Skriv in ett heltal: " );
        x = scan.nextInt();
        System.out.print("Skriv in ett heltal: " );
        y = scan.nextInt();

        max = (x>y? x : y);

        System.out.println("Maximum för " + x + " och " + y + " är : " + max);
    }
}
```

Nästling

- En kontrollstruktur kan ligga före eller efter en annan Kontrollstruktur i koden(sequence).
- **En kontrollstruktur kan ligga i en annan Kontrollstruktur i koden(Nästling).**
- **Det finns ingen begränsning när det gäller nästlingsdjupet.**



Thanks for listening!