EDAA20
Programmering och databaser

Föreläsning 2, 2023-08-29, Niklas Fors

Dagens föreläsning

- `if`-satser – utför något ibland – om ett villkor är sant
- `for`/`while`-satser – iteration, utför något många gånger
- Slumptal
- Objekt, metodanrop, argument

Villkorlig exekvering – `if`-satser

```
Scanner scan = new Scanner(System.in);
int nbr1 = scan.nextInt();
int nbr2 = scan.nextInt();
if (nbr2 != 0) {
  System.out.println("Kvoten är " + (nbr1 / nbr2));
} else {
  System.out.println("Det går ej dela med 0");
}
```

Med en `if`-sats kan man utföra olika saker beroende på ett eller flera villkor.

Logiska uttryck

Villkoret i `if`- och `while`-satser är ett logiskt uttryck

Ett logiskt uttryck beräknas till antingen `true` eller `false`

Jämförelseoperatorer

Operator	Beskrivning	Exempel
`<`	Mindre än	`n < 10`
`<=`	Mindre än lika med	`n <= 10`
`>`	Större än	`y+1 > n`
`>=`	Större än lika med	`n >= 5`
`==`	Lika med	`n == y`
`!=`	Skilt från	`n != 10`

De fyra första jämförelseoperatorerna kan användas för tal (heltal/flyttal).

De två sista kan användas för fler typer, exempelvis för logiska (`b == true`)

Logiska operatorer

Ett villkor kan bestå av delvillkor med ***logiska operatorer***

Logisk operator	Namn	Exempel
&&	och	`n >= 1 && n <= 10`
||	eller	`n < 0 || n > 0`
!	icke	`!(n >= 1 && n <= 10)`

Villkorlig exekvering – `if`-satser

```
if (villkor) {
  ...           // utförs om villkoret är sant (true)
}
```

```
if (villkor) {
  ...           // utförs om villkoret är sant
} else {
  ...           // utförs annars
}
```

```
if (villkor 1) {
  ...           // utförs om villkor 1 är sant
} else if (villkor 2) {
  ...           // utförs annars om villkor 2 är sant
} else {
  ...           // utförs annars (dvs, både villkor 1 och 2 är falska)
}
```

Övning

Skriv färdigt följande kod som ska skriva ut "negativt", "noll" eller "positivt" beroende på det inlästa talet (n).

```
Scanner scan = new Scanner(System.in);
int n = scan.nextInt();
if (n < 0) {
  System.out.println("negativt");
}
```

Allmän form av `if`-sats:

```
if (villkor 1) {
  ...
} else if (villkor 2) {
  ...
} else {
  ...
}
```

Övning

Allmän form av `if`-sats:

```
if (villkor 1) {
  ...
} else if (villkor 2) {
  ...
} else {
  ...
}
```

Skriv färdigt följande kod som ska skriva ut "negativt", "noll" eller "positivt" beroende på det inlästa talet (n).

```
Scanner scan = new Scanner(System.in);
int n = scan.nextInt();
if (n < 0) {
  System.out.println("negativt");
} else if (n > 0) {
  System.out.println("positivt");
} else {
  System.out.println("noll");
}
```

Läs in 5 tal och summera dem

```
Scanner scan = new Scanner(System.in);
int sum = 0;
sum = sum + scan.nextInt();
sum = sum + scan.nextInt();
sum = sum + scan.nextInt();
sum = sum + scan.nextInt();
sum = sum + scan.nextInt();
System.out.println(sum);
```

Samma kodrad flera gånger… kanske finns det ett bättre sätt?

Iteration – `for`-sats

```
Scanner scan = new Scanner(System.in);
int sum = 0;
for (int i = 1; i <= 5; i++) {
  sum = sum + scan.nextInt();
}
System.out.println(sum);
```

Programmet läser in 5 heltal från användaren och summerar dem.

Iteration – `for`-sats

```
Scanner scan = new Scanner(System.in);
int sum = 0;
for (int i = 1; i <= 5; i++) {
  sum = sum + scan.nextInt();
}
System.out.println(sum);
```

Samma sak som:

```
i = i + 1;
```

Programmet läser in 5 heltal från användaren och summerar dem.

Iteration – `for`-sats

```
for (int i = start; i <= slut; i++) {
   ...
}
```

Denna `for`-sats utförs för `i`-värdena start, start+1, start+2, …, slut.

Man kan använda variabeln `i` inuti `for`-satsen.

Iteration – `for`-sats

Startvärde,
utförs en gång i början

Gå ett steg framåt,
uppdateras i slutet av iterationen/steget

```
for (int i = start; i <= slut; i++) {
   ...
}
```

Ska vi fortsätta?
Kontrolleras i början av
varje iteration/steg

Övning – `for`-sats

Allmän form av `for`-sats:

```
for (int i = start; i <= slut; i++) {
  ...
}
```

Skriv satser för att skriva ut 50 asterisker (*):

Övning – `for`-sats

Allmän form av `for`-sats:

```
for (int i = start; i <= slut; i++) {
  ...
}
```

Skriv satser för att skriva ut 50 asterisker (*):

```
for (int i = 1; i <= 50; i++) {
  System.out.println("*");
}
```

Övning – `for`-sats

Allmän form av `for`-sats:

```
for (int i = start; i <= slut; i++) {
  ...
}
```

Skriv satser för att beräkna och skriva ut summan av:
1 + 2 + 3 + 4 + … + 10

Övning – `for`-sats

Allmän form av `for`-sats:

```
for (int i = start; i <= slut; i++) {
  ...
}
```

Skriv satser för att beräkna och skriva ut summan av:
1 + 2 + 3 + 4 + … + 10

```
int sum = 0;
for (int i = 1; i <= 10; i++) {
   sum = sum + i;
}
System.out.println(sum);
```

En summa-variabel behövs!

Iteration – `while`-sats

```
while (villkor) {
  ...                  // utförs så länge villkoret är sant
}
```

Man brukar använda `while` när man på förhand inte vet hur många iterationer man ska utföra.

Iteration – `while`-sats

```
Scanner scan = new Scanner(System.in);
int n = scan.nextInt();
int sum = n;
while (n != 0) {
  n = scan.nextInt();
  sum = sum + n;
}
System.out.println(sum);
```

Programmet läser in och summerar flera heltal tills det inlästa talet är en nolla (0).

Iteration – `for/while`-sats

```
for (int i = start; i <= slut; i++) {
   ...
}
```

Ovanstående `for`-sats kan skrivas om till en `while`-sats:

```
int i = start;
while (i <= slut) {
   ...
   i++;
}
```

Slumptal

Det finns en färdig klass Random för att generera slumptal. Två metoder:

double	**nextDouble**()	Returns the next pseudorandom, uniformly distributed double value between 0.0 and 1.0 from this random number generator's sequence.
int	**nextInt**(int bound)	Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.

Exempel, simulering av myntkast:

```
Random rand = new Random();
double d = rand.nextDouble();
if (d < 0.5) {
  System.out.println("krona");
} else {
  System.out.println("klave");
}
```

Övning - Slumptal

Det finns en färdig klass `Random` för att generera slumptal. Två metoder:

`double`	**`nextDouble`**`()`	Returns the next pseudorandom, uniformly distributed `double` value between `0.0` and `1.0` from this random number generator's sequence.
`int`	**`nextInt`**`(int bound)`	Returns a pseudorandom, uniformly distributed `int` value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.

Skriv kod för att dra och skriv ut ett slumptal mellan 1 och 6 (heltal):

```
Random rand = new Random();
```

Övning - Slumptal

Det finns en färdig klass Random för att generera slumptal. Två metoder:

double	**nextDouble**()	Returns the next pseudorandom, uniformly distributed double value between 0.0 and 1.0 from this random number generator's sequence.
int	**nextInt**(int bound)	Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.

Skriv kod för att dra och skriv ut ett slumptal mellan 1 och 6 (heltal):

```
Random rand = new Random();
int nbr = rand.nextInt(6) + 1;
System.out.println(nbr);
```

Mer komplicerad övning - Slumptal

`double`	**`nextDouble`**`()`	Returns the next pseudorandom, uniformly distributed `double` value between `0.0` and `1.0` from this random number generator's sequence.
`int`	**`nextInt`**`(int bound)`	Returns a pseudorandom, uniformly distributed `int` value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.

Skriv kod för att kasta en tärning tills det blir en 6:a (dvs, dra ett slumptal mellan 1 och 6 tills det blir en 6:a). Antal kast ska räknas och skrivas ut.
Tips: en `while`-sats och en räknare-variabel behövs.

```
Random rand = new Random();
```

Allmän form av `while`-sats:

```
while (villkor) {
  ...
}
```

Mer komplicerad övning - Slumptal

double	**nextDouble**()	Returns the next pseudorandom, uniformly distributed double value between 0.0 and 1.0 from this random number generator's sequence.
int	**nextInt**(int bound)	Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.

```
Random rand = new Random();
int nbr = rand.nextInt(6) + 1;
int count = 1;
while (nbr != 6) {
  nbr = rand.nextInt(6) + 1;
  count = count + 1;      // Man kan också skriva: count++;
}
System.out.println("Antal kast: " + count);
```

Annat exempel: ett fönster med en kvadrat

```
import se.lth.cs.pt.square.Square;
import se.lth.cs.pt.window.SimpleWindow;

public class DrawSquare {
  public static void main(String[] args) {
    SimpleWindow w = new SimpleWindow(600, 600, "Square");
    Square sq = new Square(20, 10, 40);
    sq.draw(w);
    System.out.println(sq.getArea());
  }
}
```

Det här programmet använder ett bibliotek skrivet av lärare på institutionen.

Ett fönster-objekt skapas med en viss storlek och titel.
Därefter skapas ett kvadrat-objekt med en position och en storlek, som sedan ritas i fönstret.
Till sist skrivs arean ut för kvadraten.

Objekt

- Ett ***objekt*** har ***egenskaper (attribut)*** och ***operationer (metoder)***
  - Ett objekt är en *instans* av en **klass** (mall)

- Ett objekt skapas med operatorn new. Exempel:

```
Square sq = new Square(20, 10, 40);
```

- En kvadrat har attributen: koordinat (20, 10) och storlek (40)
- Exempel på metodanrop: rita kvadraten i ett fönster (w)

```
sq.draw(w);
```

Metodanrop

```
public class ComputeSquareRoot {
  public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    System.out.println("Skriv ett tal");
    double nbr = scan.nextDouble();
    double sq = Math.sqrt(nbr);
    System.out.println(sq);
  }
}
```

Programmet läser in ett flyttal och beräknar dess kvadratrot.

Vi anropar färdiga metoder för delproblemen:
- println
- nextDouble
- sqrt

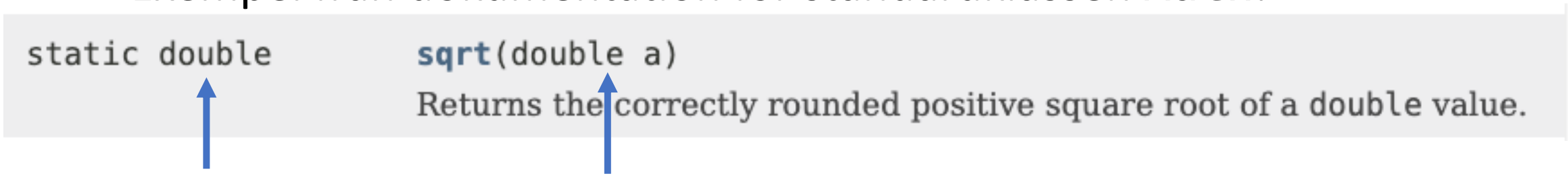
Metodanrop

- När man ***anropar*** en metod ska noll eller flera ***argument*** skickas med
- Argumenten är värden som metoden behöver för att lösa sin uppgift
- Antal argument och argumentens typer framgår av metodens dokumentation
- Exempel på ***metodanrop***:

```
double sq1 = Math.sqrt(20.5);
double sq2 = Math.sqrt(5 + 14.7 * 3);
double nbr = 42;
double sq3 = Math.sqrt(nbr);
```

Hur vet vi vad vi kan anropa?

- Hur vet vi att man ska skriva `Math.sqrt(20.5)` för att beräkna $\sqrt{20.5}$?
- Vi måste få reda på:
  - Det finns en klass `Math` med metoden `sqrt`
  - Hur många argument man ska skicka med
  - Om metoden ger tillbaka ett värde och vilken typ i sådana fall
- Exempel från dokumentation för standardklassen `Math`:



Metoden ger tillbaka ett värden av typen `double` (returtyp)

Metoden behöver ett argument av typen `double`

Metodanrop

```
public class ComputeSquareRoot {
  public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    System.out.println("Skriv ett tal");
    double nbr = scan.nextDouble();
    double sq = Math.sqrt(nbr);
    System.out.println(sq);
  }
}
```

Vad är det för likheter/skillnader mellan anropen `nextDouble` och `sqrt`?

Dokumentation för klassen `Math`:

`static double`	**`sqrt`**`(double a)` Returns the correctly rounded positive square root of a `double` value.

`static` ⟹ man kan anropa metoden direkt med klassnamnet:

```
int n = Math.sqrt(...);
```

Dokumentation för klassen `Scanner`:

`double`	**`nextDouble`**`()`	Scans the next token of the input as a `double`.

Ej `static` ⟹ man måste först skapa ett objekt som man sedan anropar på:

```
Scanner scan = new Scanner(System.in);
double d = scan.nextDouble();
```

Läsbara program

- Korrekt Java-program men ej läsbart:

```
import java.util.Scanner;public class p {public static void main(String[]
args) {System.out.println("Skriv två tal"); Scanner a=new Scanner(System.in);
double b=a.nextDouble(); double c=a.nextDouble(); double d=b+c;
System.out.println("Summan av talen är "+d);}}
```

- Välj goda och beskrivande namn på klasser, metoder och variabler
- Klassnamn börjar med stor bokstav och variabler med liten bokstav
- Indentera koden (tab/mellanslag i början av raden) så man lättare ser vad som hör till vad. Man brukar göra det i samband med block { }

Uppdelning

- I Java delar man upp programmet i ***klasser***
  - Normalt en fil per klass
- En klass kan innehålla flera ***metoder***
- En metod är en sekvens av ***satser***

Vi börjar med en klass med en metod, men senare kommer vi att skriva flera klasser med flera metoder i varje klass.

Checklista på vad du ska kunna

- Förklara begrepp:
  - Algoritm, program, källkod, kompilering, exekvering
  - Variabel, datatyp, tilldelningssats
  - Metod, argument
- Förstå att programkod delas upp i klasser, metoder och satser
- Skriva enkla program:
  - Deklarera variabler och tilldela dem värden
  - Läsa in värden från tangentbordet, skriva ut på skärmen
  - Använda `if`-, `for`- och `while`-satser
  - Läsa dokumentationen och anropa metoder
  - Använda metoder i `Random` för att dra slumptal
- Editera, kompilera och exekvera program