Introduktion till Java

– för Pythonprogrammerare

Bakgrund:
Varifrån kommer Java? Varför använda det?

Historia: C, C++

jonkv@ida

1960-talet: **CPL** → **BCPL** → **B**

↓

1969-1973: **C** → skrev om delar av Unix-kärnan

↓

1978: **K&R C** – "*The C Programming Language*" av Kernighan & Ritchie

↓

1979: **C with Classes** – OO-finesser från Simula (Bjarne Stroustrup)

↓

1983: **C++** – Vidareutveckling

"C++":
”C += 1” or
"C = C + 1"

↓

1985: **The C++ Programming Language** (bok)

↓

1989: **C++ 2.0**

Historia: Oak 1

jonkv@ida

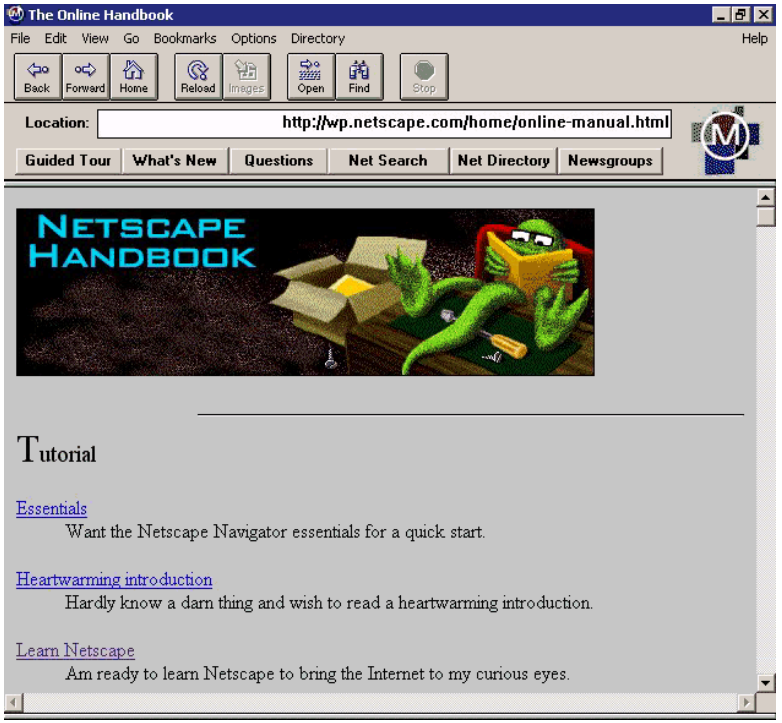
- 1990: Sun Microsystems påbörjade projekt **StarSeven**
  - Avancerad interaktiv ”multikontroll”: TV, video, satellit, …
    - *Touchskärm*, dra program i listan till en video, …
    - *Trådlöst nätverk*
    - Unix-OS

Historia: Oak 2

- Ville ha ett objektorienterat **programmeringsspråk**
  - C++ sågs som problematiskt
  - James Gosling utvecklade sitt eget: "**C++ ++ – –**"
    - Omdöpt till "Oak"
  - 1991–1994:
    - Skapa spinoff-företag
    - Bygga, programmera, demonstrera
    - Gå i konkurs

Historia: Oak 3

- Hitta en **annan användning**!
  - 1994: WWW på gång
    - Nov 1993: 500 WWW-servrar!
    - Okt 1994: Netscape Mosaic 0.9!
  - Webben var väldigt statisk
    - Interaktivitet? Formulär!
    - JavaScript? Slutet av 1995…
  - **Oak passade perfekt**!
    - Portabelt, plattformsoberoende
    - Bra stöd för nätverkskommunikation
  - Språket döptes om till **Java**
    - Många arv kvar från C++, C, till och med B (1969)



Inte Python igen: Behöver **breda** kunskaper, skilda perspektiv!

Learn at least a half dozen programming languages.
– *Peter Norvig (2001): Teach Yourself Programming in Ten Years, http://norvig.com/21-days.html*

Språk:

Varför Java, när det finns så många OO-språk?

Vanligt:
#4 på TIOBE index
(Python, C, C++)
#3 på RedMonk
(efter JavaScript, Py)
#2 på
GitHub/GitHut
(efter Python)

Vissa liheter med vanliga språk:
C, C++, C#,
(PHP, Javascript, …)

Använder vanliga begrepp/tankesätt:
Typning, klasser,
statisk scoping, …

Medelkomplexitet:
Lättare att börja med
än t.ex. C++

Passar till undervisning:
Färre fallgropar än vissa språk,
bra utvecklingsmiljöer,
bra grundbibliotek av kod/klasser, …

Java:
Ett långsamt språk?

Java – långsamt?

jonkv@ida

- Vi testar…
  - Multiplicera matriser av storlek 1024*1024 (nästlade loopar)

```
n = 1024

A = [[random.random() for row in range(n)] for col in range(n)]
B = [[random.random() for row in range(n)] for col in range(n)]
C = [[0 for row in range(n)] for col in range(n)]

start = time.time()

for i in range(n):
    for j in range(n):
        for k in range(n):
            C[i][j] += A[i][k] * B[k][j]

end = time.time()
```

Java – långsamt? [Ryzen 3900x, Ubuntu 21.10]

jonkv@ida

- Matrismultiplikation, 1024x1024:
  - **104 sek** **Python 3.9, ”vanliga loopar”**
  - **0.51 sek** **Java 17, ”vanliga loopar”**

Extremfall:
Simpla loopar, 1 miljard add, 1 miljard multiplikationer, kan *vektoriseras*, …

  - 53.9 Python, ”optimerade loopar” (sum, zip – mer sker ”internt” i C-kod)
  - 10.3 PyPy 7.3.5, samma optimeringar, (fungerar inte med allt!)
  - 4.2 PyPy, ”vanliga loopar” – *snabbare* för PyPy
  - 0.20 Python med NumPy 1.22 (~~loopar~~ ➔ anropa 500 rader optimerad C-kod)
  - 0.13 Java, diverse (andra) optimeringar
  - 2.8 C, gcc 11.2.0, ”vanliga loopar”
  - 0.14 C, gcc –O3 (optimerande), ger *vektoriserad kod* – extremt snabbt **just för matriser**
  - 0.09 C, gcc –O3 –march=native, *nyare vektorinstruktioner används*

Dra inte för starka slutsatser av ett enda specialtest!
Denna kod ger bland de extremaste skillnaderna för Python/Java/C!

Python 3.11: (104 -> 52) sek, (53.9 -> 37) sek

Python och Java:
Hur skriver vi kod?

Många nya begrepp, mycket ny syntax...

jonkv@ida

Python:
Flera programmeringsparadigm
Inklusive imperativ procedurell kod

Java:
Klassbaserad objektorientering
All kod är i en klass,
alla satser i en metod (funktion)

HelloWorld.py

```
print("Hello World")
```

Kod på toppnivå körs när man "kör filen"

HelloWorld.java

```
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello World!");
   }
}
```

main() startas när man "kör klassen"

Enkel uppgift – enkel kod

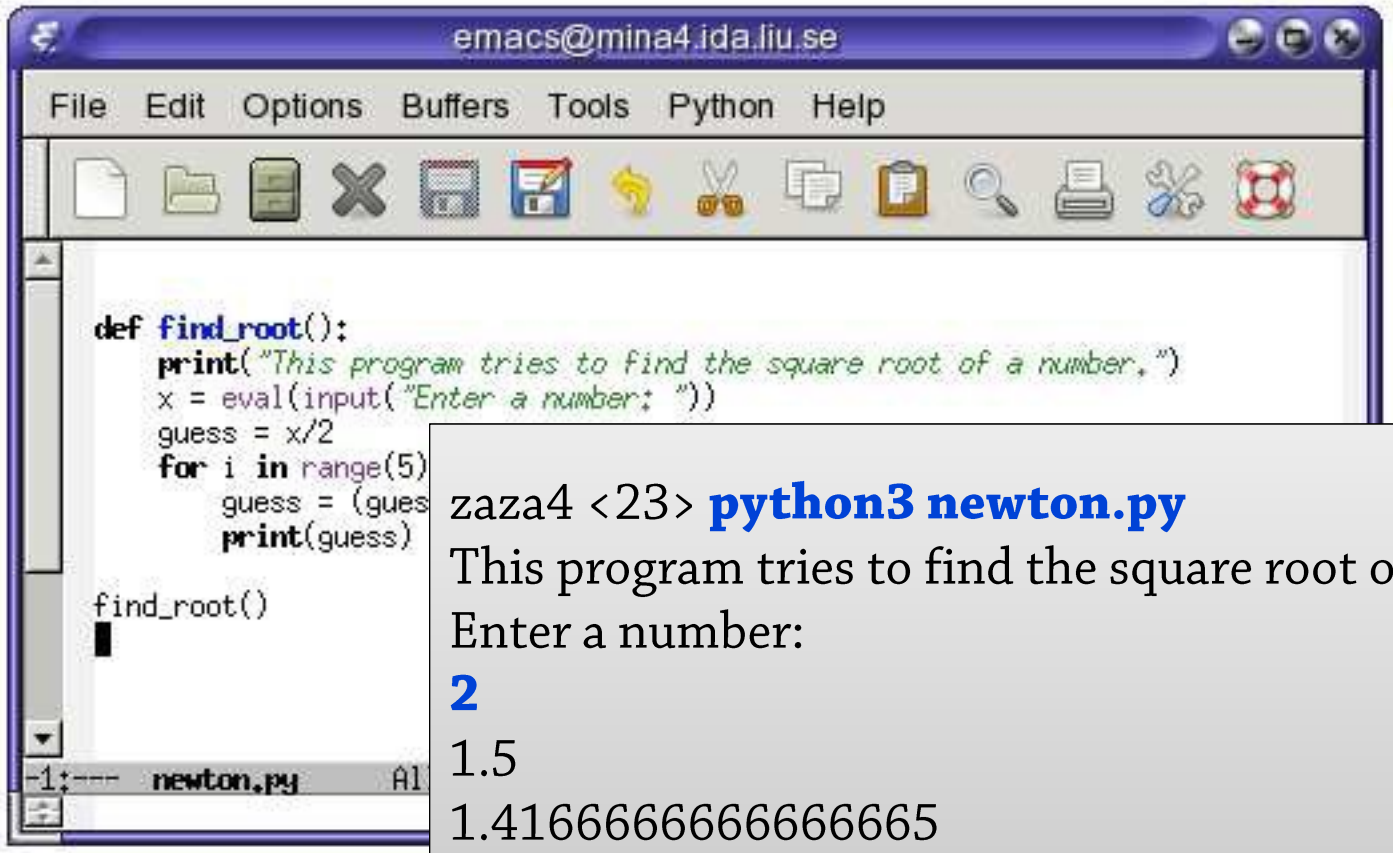
Mer "overhead" runtomkring…
Märks mindre i större program!

Kan inte förklara allt på en gång!
Acceptera class, static, … tills vidare –
förklaras mer senare…

klass = class
metod = method

Interaktivitet (1)

- Python kan köras med kod **sparad i en fil**…

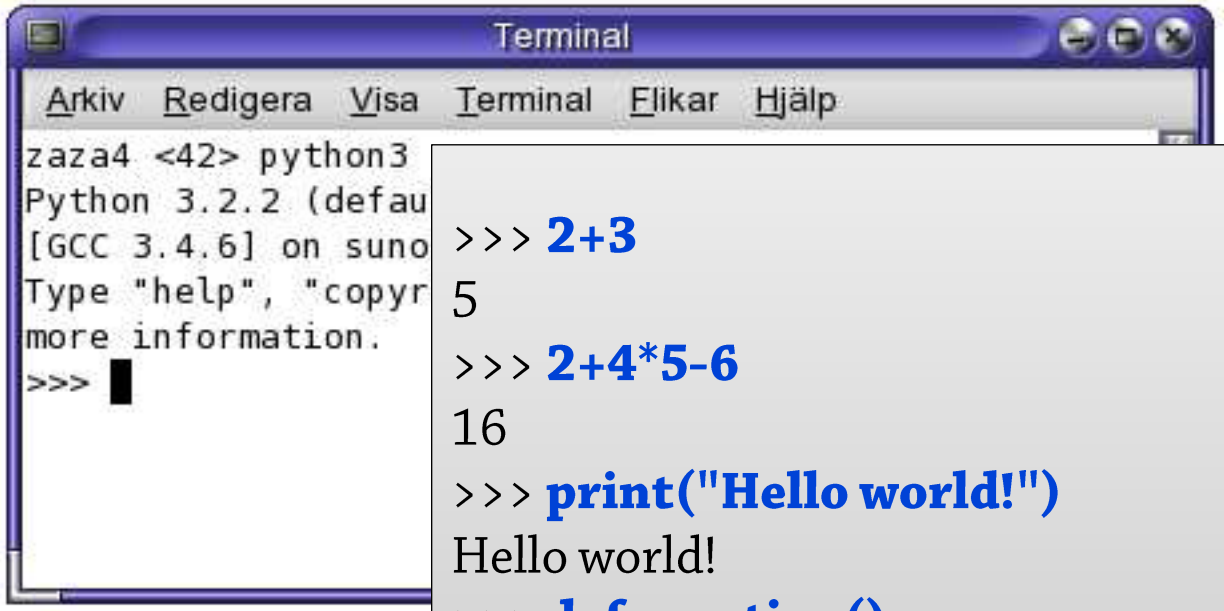



```
zaza4 <23> python3 newton.py
This program tries to find the square root of a number.
Enter a number:
2
1.5
1.4166666666666665
1.4142156862745097
1.4142135623746899
1.414213562373095
```

Interaktivitet (2)

jonkv@ida

- …eller i **interaktivt läge**




```
>>> 2+3
5
>>> 2+4*5-6
16
>>> print("Hello world!")
Hello world!
>>> def greeting():
... print("Nobody expects the Spanish Inquisition.")
... print("Our chief weapon is surprise... and fear.")
...
>>> greeting()
Nobody expects the Spanish Inquisition.
Our chief weapon is surprise... and fear.
```

Interaktivitet (3)

- Java körs *normalt* med kod **sparad i en fil**…

Fil: Newton.java

```
public class Newton {
   public static void findRoot() {
      System.out.println("This program tries to find…");
      System.in.read……;
      double guess = x/2;
      for (int i  = 1; i < 5; i++) {
         guess = (guess + x/guess) / 2;
         System.out.println(guess);
      }
   }
   public static void main(String[ ] args) {
      findRoot();
   }
}
```

Interaktivitet (4)

- …men numera finns också **jshell** (här i *verbose*-läge):

```
jshell> 2+3
$1 ==> 5
|  created scratch variable $1 : int

jshell> 2+4*5+6
$2 ==> 28
|  created scratch variable $2 : int

jshell> System.out.println("Hello world!")
Hello world!

jshell> void greeting() {
   ...>     System.out.println("Nobody expects the Spanish Inquisition.");
   ...>     System.out.println("Our chief weapon is surprise... and fear.");
   ...> }
|  created method greeting()

jshell> greeting();
Nobody expects the Spanish Inquisition.
Our chief weapon is surprise... and fear.
```

Har en del smarta funktioner (tab-komplettering, hjälptexter, …):
http://cr.openjdk.java.net/~rfield/tutorial/JShellTutorial.html

Python och Java:
Skillnader i ytlig struktur

Syntax: Kommentarer

jonkv@ida

Fil: Newton.py

```
# Vad ska vi skriva här?
```

Fil: Newton.java

```
// Vad ska vi skriva här?

/* En lång kommentar
   som kan sträcka sig
   över flera rader */
```

Syntax: Kod i klasser

jonkv@ida

Fil: Newton.py

Fil: Newton.java

```
public class Newton {
}
```

All kod måste ligga i en klass…

Det som ingår i klassen läggs inom { … }
(Java bryr sig inte om indentering!)

Namnstandard för klasser:
VarjeOrdHarStorBokstav
JavaTest, ArrayList, …

Unicode – kan använda
å, Θ, π, Σ, Я, Õ̓

Mer om klasser, filer, … en kommande gång! Nu: Syntax

Syntax: Funktioner

jonkv@ida

Fil: Newton.py

```
def find_root():
```

Fil: Newton.java

```
public class Newton {
    public static void findRoot() {
    }
}
```

Just nu vill vi inte använda objekt
Ändå måste metoder vara inuti klasser

public ➔
vem som helst får använda

static ➔
behöver inget objekt av typen Newton
("vanlig funktion, inte objektorienterad")

void ➔
returnerar inget värde (*procedur*)

Syntax: Klamrar

jonkv@ida

Fil: Newton.py

```
def find_root():
```

Fil: Newton.java

```
public class Newton {
   public static void findRoot() {
```

Klamrar runt metodens kod: { … }

Namnstandard:
storBokstavUtomFörstaOrdet

Syntax: Strängar, utskrifter

jonkv@ida

Fil: Newton.py

```
def find_root():
    print("This program tries to find....")
```

Python-strängar:
"Hello" eller 'Hello'

Jämförelse: str1 == str2

Fil: Newton.java

```
public class Newton {
    public static void findRoot() {
        System.out.println("This...");
    }
}
```

Java-strängar:
"Hello"

Jämförelse: str1.equals(str2)
*(för **objekt**; mer en annan gång!)*

Apostrofer kan användas för enskilda *tecken*: 'H'

Syntax: Semikolon, indentering

jonkv@ida

Fil: Newton.py

```
def find_root():
    print("This program tries to find....")
```

Fil: Newton.java

```
public class Newton {
    public static void findRoot() {
        System.out.println("This...");
    }
}
```

Utskrift är en *metod*, "println", i ett objekt, "System.out"
➔ Mer info senare!

Semikolon efter satser
Radbrytning räcker inte!

Vi indenterar för läsbarhet
Ignoreras av språket

```
public class Newton { public static void  findRoot() { System.out.println("This..."); } }
```

Inmatning

jonkv@ida

Fil: Newton.py

```
def find_root():
    print("This program tries to find....")
    x = eval(input("Enter a number: "))
```

Fil: Newton.java

```
public class Newton {
    public static void findRoot() {
        System.out.println("This...");
        ...;
```

Java är mindre anpassat för "textprogram"

Enklare att visa en dialogruta (kommer på labben)

```
    }
}
```

Deklarera variabler

Fil: Newton.py

```
def find_root():
    print("This program tries to find....")
    x = eval(input("Enter a number: "))
    guess = x/2
```

Det kan väl datorn fatta själv?

Ja, men tänk om du skriver *guss* nästa gång.
Mer att skriva men extra säkerhetsbälte.

Det kan väl datorn fatta själv?

Delvis, men det har också nackdelar.
Vi återkommer till typning nästa gång!

Fil: Newton.java

```
public class Newton {
    public static void findRoot() {
        System.out.println("This...");
        ...;
        double guess = x/2;
```

**Variabler måste deklareras:
"Här kommer en ny variabel"
(annars säger kompilatorn
att "*guess* finns inte")**

```
}
```

**Java har explicit typning:
Ange alltid vilken typ en
variabel ska ha
(double = "decimaltal")**

deklarera = declare

Namnstandard

jonkv@ida

Fil: Newton.py

```
def find_root():
    print("This program tries to find....")
    x = eval(input("Enter a number: "))
    guess = x/2
```

Fil: Newton.java

```
public class Newton {
    public static void findRoot() {
        System.out.println("This...");
        ...;
        double guess = x/2;
```

Namnstandard: storBokstavUtomFörstaOrdet

```
    }
}
```

Loop över heltal

jonkv@ida

Fil: Newton.py

```
def find_root():
    print("This program tries to find....")
    x = eval(input("Enter a number: "))
    guess = x/2
    for i in range(5):
```

Fil: Newton.java

```
public class Newton {
    public static void findRoot() {
        System.out.println("This...");
        ...;
        double guess = x/2;
        for (int i  = 0; i < 5; i++) {
```

Heltalstyp: int

Annan loopsyntax: (start; villkor; steg)

Deklarera heltal i = 0
Så länge som i < 5 {
**   Utför "kroppen" av loopen**
**   i++ (öka värdet på i)**
}

Loop med klamrar

jonkv@ida

Fil: Newton.py

```
def find_root():
    print("This program tries to find....")
    x = eval(input("Enter a number: "))
    guess = x/2
    for i in range(5):
```

Fil: Newton.java

```
public class Newton {
    public static void findRoot() {
        System.out.println("This...");
        ...;
        double guess = x/2;
        for (int i  = 0; i < 5; i++) {
```

Loopens kropp läggs också inom klamrar { … }

Efter klamrar: Inget semikolon

```
        }
    }
}
```

Tilldelning och uttryck

Fil: Newton.py

```
def find_root():
    print("This program tries to find....")
    x = eval(input("Enter a number: "))
    guess = x/2
    for i in range(5):
        guess = (guess + x/guess) / 2
```

Fil: Newton.java

```
public class Newton {
    public static void findRoot() {
        System.out.println("This...");
        ...;
        double guess = x/2;
        for (int i  = 0; i < 5; i++) {
            guess = (guess + x/guess) / 2;
        }
    }
}
```

Nu är *guess* redan deklarerad!
Många operatorer liknar Python…

Referens: Operatorer i Python och Java

jonkv@ida

Python		Java
Räknesätt:	+ - * / %	*Samma*
Upphöjt till:	**	Använd: Math.pow(bas, exponent)
Division avrundad nedåt:	//	Använd: (int) (x / y)
Jämförelser:	== != > < >= <=	*Samma*
Olikhet:	<>	Använd !=
Tilldelning:	= += -= *= /= %=	*Samma*
Tilldelning:	i += 1	Använd i++
Tilldelning:	i -= 1	Använd i--
Bitoperatorer:	& | ~ << >>	*Samma (men finns flera)*
Villkor:	and	Använd: &&
	or	||
	not	!

Mera utskrifter

jonkv@ida

Fil: Newton.py

```
def find_root():
    print("This program tries to find....")
    x = eval(input("Enter a number: "))
    guess = x/2
    for i in range(5):
        guess = (guess + x/guess) / 2
        print(guess)
```

Fil: Newton.java

```
public class Newton {
    public static void findRoot() {
        System.out.println("This...");
        ...;
        double guess = x/2;
        for (int i  = 0; i < 5; i++) {
            guess = (guess + x/guess) / 2;
            System.out.println(guess);
        }
```

Utskrift igen...

```
    }
}
```

Klart!

jonkv@ida

Fil: Newton.py

```
def find_root():
    print("This program tries to find....")
    x = eval(input("Enter a number: "))
    guess = x/2
    for i in range(5):
        guess = (guess + x/guess) / 2
        print(guess)


    print "Done!"
```

Fil: Newton.java

```
public class Newton {
    public static void findRoot() {
        System.out.println("This...");
        ...;
        double guess = x/2;
        for (int i  = 0; i < 5; i++) {
            guess = (guess + x/guess) / 2;
            System.out.println(guess);
        }

        System.out.println("Done!");
    }
    ...
}
```

Sammanfattning: Satser och gruppering

jonkv@ida

Fil: Newton.py

```
def find_root():
    print("This program tries to find....")
    x = eval(input("Enter a number: "))
    guess = x/2
    for i in range(5):
        guess = (guess + x/guess) / 2
        print(guess)
    print "Done!"
```

Radbrytning avslutar sats

Gruppering via indentering

Fil: Newton.java

```
public class Newton {
    public static void findRoot() {
        System.out.println("This...");
        ...;
        double guess = x/2;
        for (int i  = 0; i < 5; i++) {
            guess = (guess + x/guess) / 2;
            System.out.println(guess);
        }
        System.out.println("Done!");
    }
    ...
}
```

Semikolon eller {} avslutar sats

Gruppering via klamrar {}

Indentering ändå viktigt för läsbarhet!

Repetition: Funktioner

jonkv@ida

- Pythons funktioner på toppnivå kan "emuleras" i Java
  - Använd en **public static**-funktion
  - *Bara till vi har hunnit läsa om objekt!*

Ange alltid returtyp
Om inget returneras: void

Fil: Newton.java

```
public class Newton {
   public static double findRoot() {
      System.out.println("This program tries to find...");
      System.in.read.....;
      double guess = x/2;
      for (int i  = 1; i < 5; i++) {
         guess = (guess + x/guess) / 2;
      }
      return guess;
   }
   public static void main(String[ ] args) {
      System.out.println(findRoot());
   }
}
```

Returnera värde med **return**

Mer Java:
Villkorssatser

sats = statement

Villkor: if

jonkv@ida

Python

```
if condition:
    statement1
    statement2
elif condition2:
    statement1
    statement2
elif condition3:
    statement1
    statement2
else:
    statement1
    statement2
```

Java

```
if (condition) {
    statement1;
    statement2;
} else if (condition2) {
    statement1;
    statement2;
} else if (condition3) {
    statement1;
    statement2;
} else {
    statement1;
    statement2;
}
```

Test av samma uttrycks värde

jonkv@ida

```
if (x + y == 0) {
    System.out.println("Exakt noll");
} else if (x + y == 1 || x + y == 2) {
    System.out.println("Ett eller två");
} else if (x + y == 3) {
    System.out.println("Exakt tre");
    System.out.println("Fler satser här");
} else {
    System.out.println("Något annat");
}
```

Här jämför vi värdet på x+y med några kända *konstanter*…

Använd == för "enkla" datatyper som heltal (mer om det senare)

konstanter = constants

Satser: switch

jonkv@ida

```
switch (x + y) {
   case 0:
      System.out.println("Exakt noll");
      break;
   case 1:
   case 2:
      System.out.println("Ett eller två");
      break;
   case 3:
      System.out.println("Exakt tre");
      System.out.println("Fler satser här");
      break;
   default:
      System.out.println("Något annat");
}
```

Kan göras med en switch-sats!
Fungerar för heltal, strängar och *enum-konstanter*

"Klar": Hoppa ur switch-satsen

Flera fall (1 och 2) ger samma kod att utföra

Frivilligt: Vad händer när inget av fallen passar?

Tydligt att man testar värdet på ***ett*** uttryck; skriv uttrycket (x+y) ***en gång***

Kräver extra "break"…

Satser: switch med fallthrough

jonkv@ida

```
switch (x + y) {
    case 0:
        System.out.println("Exakt noll");
        break;
    case 1, 2:
        System.out.println("Ett eller två");
    case 3:
        System.out.println("Ett, två eller tre");
        break;
    default:
        System.out.println("Något annat");
}
```

Inget 'break' här,
så vi *fortsätter* in i nästa fall

Fallthrough kan förvirra – oftast bra att skriva på annat sätt!

Satser: "Ny" switch (från Java 12)

jonkv@ida

```
switch (x + y) {
    case 0 -> System.out.println(“Exakt noll");
    case 1, 2 -> System.out.println(“Ett eller två");
    case 3 -> System.out.println(“Bara tre");
    default -> {
        System.out.println(“Något annat");
        System.out.println(“Flera satser inom klamrar");
    }
}
String monthName = switch (monthNumber) {
    case 1 -> “January”;
    case 2 -> “February”;
    …
}
```

Ingen fallthrough finns!

Se även https://docs.oracle.com/en/java/javase/21/language/switch-expressions.html

Kommande föreläsningar:

När variabler har datatyper: I Java och många andra språk

Objektorientering...