Kom igång!

Vecka 1, Bildserie 1

Innehåll

- Språket Java
- Program och fil
- En mall för Java-program
- Kodstil
- Satser
- Utmatningssatser
- Block

Mål vecka 1

Kunna skriva enkla (textbaserade) spel och liknande m.h.a. grundläggande imperativa koncept

Veckans Produkt

```
Welcome to till NIM
There's 13 coins in the pile
Pick coins > 3
Player took 3 // TODO ta bort
There's 10 coins in the pile
Computer took 1
There's 9 coins in the pile
Pick coins > 3
Player took 3
There's 6 coins in the pile
Computer took 2
There's 4 coins in the pile
Pick coins > 3
Player took 3
Game over. Winner is: Human
```

Vi skriver ett program för spelet Nim

Att Läsa i Boken

- 1.4-1.6, 1.9, översiktligt
- 2.2

OBS! Att jag använder lite annorlunda kodstil än boken.

- Inledningsvis behövs aldrig ordet static eller public, bortse från dessa.
- Vi skriver inte System utan bara out, se mina exempel.

Programmering i Java

Språket Java beskrivs i en specifikation

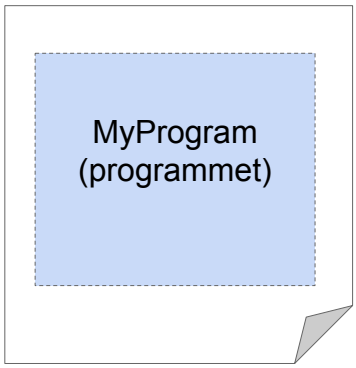
- Vi använder version 8 av språket, kallas även version 1.8
- Specifikationen beskriver hur vi skall skriva språket Java (syntax) och …
- … betydelsen av det vi skriver (semantik).

Popularitet för olika språk

OBS! Att det vi går igenom i kursen gäller för de många (de flesta) imperativa och/eller objektorienterade språken.

- Kursen är alltså inte en Java kurs.

Program och Fil



MyProgram.java
(en textfil)

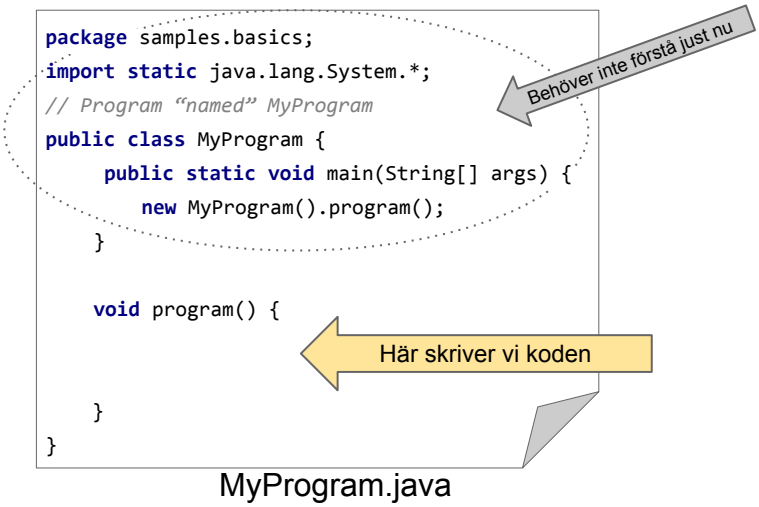
Ett Java-program skrivs och sparas i en textfil

- **Textfil** = fil med raduppbyggnad, läsbar för människor (kontra **binär fil**)
- Filen kallas källkodsfil (eller **källkoden, source code**)
- Java kräver teckenkodningen UTF-8 för källkodsfiler.

Varje program finns i en enda fil (tills vidare…)

- Programmet och filen det sparas i skall heta samma sak förutom suffixet på filnamnet (.java).
  - Båda skall inledas med stor bokstav.
- Om programmet (filen) har ett sammansatt namn skrivs det CamelCase (som i bilden)
- Man kan använda svenska namn på program men det gör inte vi: Använd engelska för **all** kod!

En Mall för Java Program



För att förenkla för nybörjare använder vi en "mall" för våra program.

Analys av mallen, se bilden (Som sagt: Vi skriver all kod på engelska)

- Filen med programmet måste ligga i en speciell (under) mapp, anges med **package** överst i filen. Filer kan alltså inte flyttas hur som helst!
  - Var programfilen finns i filsystemet och package måste stämma!
  - Normalt ligger allt som behövs för ett program i samma mapp (kan senare vara flera filer).
  - sample.basics betyder mappen src/samples/basics i IntelliJ (man utgår alltid från src)

- Vid **import** anger vi vilka färdiga Java-resurser programmet behöver (finns väldigt mycket färdiga i Java).
  - Ibland får vi lägga till någon rad här, utifrån behov.
  - I mallen anger vi att vi behöver allt (= asterisken) från "java.lang.System" (bl. a. resurser för att hantera skärm och tangentbord)

- public class MyProgram anger att programmet heter MyProgram (ungefär, mer senare …)

    - Vad programmet skall heta kan vi välja själva (program ni skall skriva har normalt ett givet namn).

- De matchande krullparenteserna, { … }, kallas ett **block** (en avgränsad del av ett program)
    - Block kan ligga inuti block = **nästlade** block.

- public static void main(...) och blocket direkt efter MÅSTE stå där!
    - Raden med new MyProgram(), betyder ungefär "skapa" programmet.
    - Exakt vad det betyder återkommer vi till, tills vidare får vi acceptera detta utan förklaring.

- Vårt program börjar vid void program()
    - void bekymrar vi oss inte för just nu, återkommer…
    - I blocket efter program() skriver vi **satser (statements)** mer strax …
    - När vi når slutet på blocket (sista krullparentesen) är programmet slut

- Vänster marginal är indragen för att visa den nästlade strukturen, kallas **indentering**
    - Indentering är mycket viktigt för förståelsen av ett program, vi använder alltid!
    - IntelliJ kan hjälpa till med detta (kallas även formatering, instruktioner kommer i labbarna)

- I programkoden kan man lägga in **kommentarer**.
    - Inleds med `//`, gäller en rad
    - eller omsluts av `/* … */` (alt. `/**…*/`) för flera rader.
    - Kommentarer har ingen påverkan på programmet. De tas automatiskt bort innan programmet körs.
        - Kommentarer är till för människor, skall underlätta förståelsen.
        - Kommentarer skrivs också på engelska.
        - IntelliJ visar kommentarer med grå kursiv stil

- Lite grundläggande syntax och semantik för språket Java:
    - Skillnad på liten/stor bokstav (string och String tolkas som två olika saker)
    - "Vita" tecken (blankslag, nyrad, etc.) spelar oftast ingen roll, dock får de inte stå "mitt i" t.ex. ett "ord" (isf blir det 2 ord …)
    - Tomma rader spelar ingen roll.

- Parenteser skall alltid matcha (...), {....}, [...], <...>, {{ … }}, …
- Satser avslutas med ";" (mer strax)
- Vissa ord är reserverade för språket Java, **reserverade ord (keywords).**
    - "import" är ett reserverat ord, inget vi namnger får heta "import", ett program får inte heta import.
    - IntelliJ visar reserverade ord i mörkblått med fet stil.
- Bryter vi mot syntaxreglerna för vi ett syntaxfel (syntax error)
    - Visas med röd understrykning i IntelliJ (peka på markering så kanske något tips visas)

Metodik

```
/*coin = new ImageIcon(ImageIO
       .read(new
File("src/exercises/optional/gold_coin_single.png")))
       .getImage();*/
coin = new ImageIcon(this.getClass()
       .getResource("gold_coin_single.png"))
       .getImage();
// getAudioInputStream() also accepts a File or InputStream
//AudioInputStream ais = AudioSystem.
// getAudioInputStream(new
File("src/exercises/optional/atari.wav"));
AudioInputStream ais = AudioSystem
     .getAudioInputStream(this.getClass()
     .getResourceAsStream("atari.wav"));
```

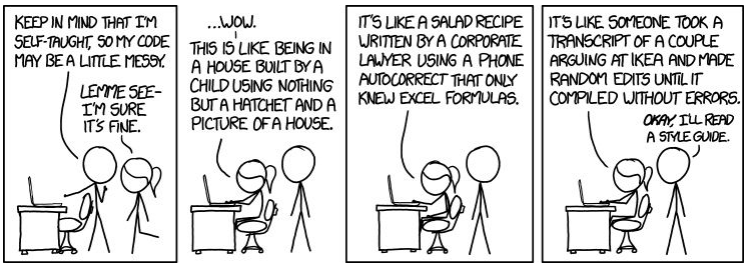
Använd kommentarer istället för att ta bort kod!

- Lätt att få tillbaka den gamla koden om den visade sig vara bra!

I bilden

- Gammal kod bortkommenterad (tills vi är säkra på att den nya är bättre, annars, kommentera ut den nya och avkommentera den gamla)

Kodstil



Java-program skall skrivas med en viss kodstil (t.ex. var krullparenteser skall stå m.m.)

- Vi följer Googles stil (i princip, … för mycket detaljer för denna kurs …)
- Använd samma stil som exempelkoden!

Satser

Exempel

```
out.println("Too small");

nGuesses++;

int theNumber = 87;

return found;
```

"Imperative programming is a programming paradigm that describes computation
in terms of statements" // Wikipedia

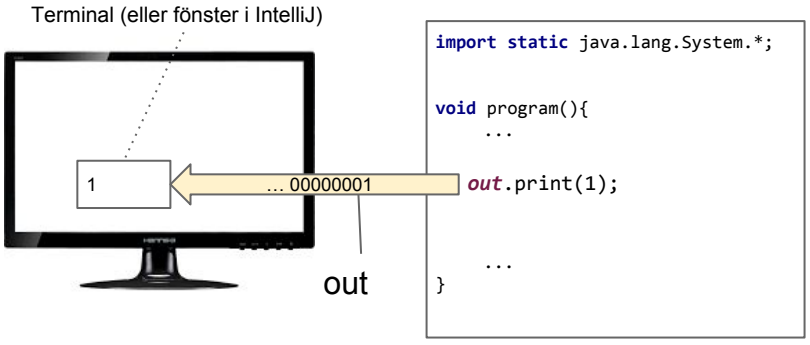
Ett Java-program är uppbyggt av en följd (en sekvens) av satser (statements).

- Ett Java program körs sats för sats tills det inte finns fler satser att exekvera.

En sats i Java

- Är den minsta fristående enheten (atomen).
  - Det finns mindre delar men dessa kan inte köras, de måste ingå i en sats.
- Är ett **imperativ,** en uppmaning till datorn att göra något (därav imperativ programmering)
- Måste avslutans med semikolon, ";" visar var satsen är slut.
  - Jämför: en mening på svenska, avslutas med punkt (eller ?, !)
- Den tomma satsen skrivs ";" (satsen exekveras men inget utförs)
- Ordningen på satserna är mycket viktig!
  - Datorn exekverar (kör) satserna i den ordning vi skrivit dem (vanlig läsordning vänster höger, uppifrån och ner)!
  - Ofta skriver vi en sats per rad.

Utmatningssatser

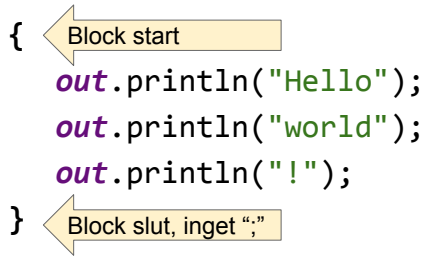


Utmatning (output)

- Java program har automatiskt tillgång till en byte-ström, out.
  - Byte-ström = en kanal för att skicka bytes
- out är i vårt fall kopplad till datorns skärm.
- Genom att använda namnet "out" i koden får programmet tillgång till strömmen
- För att skriva ut något använder vi en utmatningssats: out.print(…);
  - Betyder att det inom parentesen skall skickas till strömmen för att slutligen hamna på skärmen
  - Vill vi ha en **nyrad** efter utskriften skriver vi: out.println(…); (ln = new line)
- Vi kommer att se utskrifterna i ett fönster i IntelliJ.
- Vi måste skriva import static java.lang.System.* för att kunna använda out.
  - En hel del kodexempel visar System.out.println(…) … för jobbigt att skriva, vårt sätt är bättre …

Block med Satser

Exempel



En sats kan alltid ersättas med ett block (av eventuellt flera satser)

- Ett block räknas som en sats bestående av 0-n ihopbakade satser
  - Tomma block används sällan (framför allt inte i denna kurs)
- Inget ";" efter block,
  - Behövs inte, det syns var blocket (satsen) slutar även utan semikolon …
  - .. nämligen vid }
  - Skriver man ; efter så betyder det bara en tom sats efter blocket.

Programmering

Vi testar några utmatningssatser

Laboration 0

- Nu i eftermiddag!
- För att komma igång!

Behöver ej redovisas!

Praktikaliteter

Hitta salar (schema)

- En sal utan datorer!

Hitta hjälplistan

Skapa kursmapp, tda548!

Registrera sig för handledare!

Starta IntelliJ

Windows

- Start > Jetbrains > IntelliJ

Linux

- Öppna terminal skriv
- /chalmers/groups/ws-devel/intellij

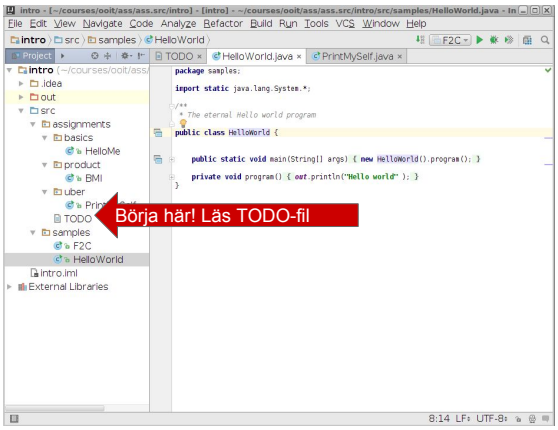
Måste ange SDK (=JDK) för båda

- File > Project Structure > Project SDK > New
- Linux: /chalmers/groups/ws-devel/jdk1.8.0_25
- Windows: Mapp med Java, välj SDK!

Om du skall installera IntelliJ på din egen dator måste du se till att du har Java JDK installerat (inte Java JRE).
Ladda ner och installera Java JDK 1.8 (kallas ochså Java SDK) och därefter IntelliJ.

Hämta Laboration

Ladda ner från kurssida, zippa upp, öppna i IntelliJ



Viktigaste kommandot: Ctrl z!

- Labben behöver inte redovisas

Kodstil

- Jag använder inledningsvis en viss kodstil (idiom)
  - Skall vara enkelt för nybörjare att komma igång
  - Vill komma igång med problemlösning så snabbt som möjligt
  - När kursen är över skriver vi “standard” Java (ingen anledning till oro …)
  - Bokenen stil skiljer sig inledningvis, runt vecka 4 har vi i princip samma stil.
  - All kod på engelska (även kommentare)

- Använd alltid i första hand mina exempel
  - Allt du behöver och behöver kunna finns som kodexempel!
  - Varning för Webb:en!!! Finns tyvärr mycket dåligt/dumt