Javaprogrammering

Folkuniversitetet - Del 1

Välkomna!

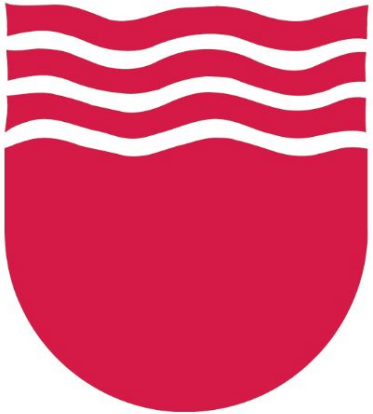
Studiecirkel

Gemensamt lärande

Varje tillfälle (3h):

1. Genomgång
2. Bensträckare
3. Uppgifter att arbeta med (hjälp från mig och varandra)

Obs håll avstånd, tvätta händerna ofta



Jag

Joakim Loxdal

Går 4e året datateknik på KTH

loxdalen@gmail.com

Programmering

- Instruktioner/kommandon till en dator
- Programmeringsspråk är gjorda för oss människor
- Översätts till något datorn förstår (maskinkod)
- Har ni någon erfarenhet sen innan av programmering?

Java

- Ett populärt programmeringsspråk, ett av de mest använda
- Skapat i början av 90-talet
- Objektorienterat språk (förklarar mer om det snart)
- Går att köra på alla operativsystem
- Javakod är hyfsat läsbart för människor
  - Exempel:

```
if (person.name == “Kungen”)  {
        person.getMoney(1000000);
}
```

- Java måste **kompileras** för att en dator ska kunna förstå det
  - Kompilera = göra om koden till ettor och nollor (maskinkod, datorns ‘språk’)

Hur skriver och kör vi ett Javaprogram?

- Java-kod kan skrivas i vilken textredigerare som helst
  - Notepad/Textedit (ett vanligt textdokument)
  - Sumblime, Atom, Brackets, VSCode (Textredigerare gjorda för programmering)
- Men! En textredigerare låter oss inte kompilera eller köra javaprogram
  - Det går dock att kompilera och köra javaprogram i terminalen
- Istället använder många en IDE (Integrated Development Environment)
  - I en IDE kan vi skriva Javaprogram, kompilera Javaprogram och köra Javaprogram
  - Vi kommer använda IntelliJ, finns även andra men IntelliJ är bäst enligt mig

JShell

- Vi använder JShell för att lära oss några grunder i Java
- I JShell skriver man in lite Java-kod och koden körs och man får svar direkt
- I IntelliJ: Tools -> JShell console

När ni öppnat JShell kan vi börja lära oss några grunder i Java

Heltal och variabler i Java

Heltal

- Kallas int (integer) i Java
- Det vi använder t.ex. när vi vill räkna matematik med heltal
- Maxvärdet på en int är 2147483647

Variabler!

- För att spara undan ett heltal (1337) i vårt javaprogram så skriver vi:

```
int a = 1337;
```

- `a` är namnet på vår variabel. En variabel är som en ‘låda’ där vi lägger in ett värde
- Vi skriver `int` för att Java ska förstå att `a` är ett heltal
- Likamedtecknet betyder: *Lägg värdet till höger i variabeln till vänster*, därför läggs värdet `1337` i variabeln `a`
- Sen kan vi använda `a` i vårt Javaprogram som om det var ett heltal (1337)

Regler om variabler

Alla variabelnamn måste börja med en bokstav

Efter första bokstaven kan variabelnamnet innehålla bokstäver och siffror, inga mellanrum

Variabelnamnet kan vara hur långt som helst men tänk på att du troligen kommer behöva skriva det flera gånger manuellt

Stora och små bokstäver gör skillnad. “Namn” är en annan variabel än “namn”

Du kan inte använda ett ‘Java-ord’ som variabelnamn, som t.ex. int.

Eftersom vi inte får ha mellanrum i variabelnamn så brukar man skriva varje ord med stor bokstav (förutom det första). Som t.ex. `String minHemstad = “Stockholm”`

De viktigaste datatyperna

Heltal (int)

- `int siffra = 1337; // finns också long (för större tal)`

Decimaltal (float)

- `float pi = 3.141592; // finns också double (för större tal)`

Boolska värden (boolean)

- `boolean kungenHeterCarl = true; boolean jagÄrKung = false;`

Bokstäver (char)

- `char betyg = 'A'` (Skrivs inom single quotes)

Textsträngar (String)

- `String fornamn = "Joakim"` (Skrivs inom dubbla citattecken)
- Stor bokstav i början (S) = det är ett objekt, har fler egenskaper än vanliga datatyper
- Exempel: toLowerCase, toUpperCase

OBS: glöm inte citattecknen! Testa vad som händer utan dom.

Matematik

Inbyggd matematik: Multiplikation (*), Addition (+), Subtraktion (-), Division (/)
Om man vill öka eller minska en siffras värde så kan man skriva ++ eller -- bredvid den

Vi kan använda den inbyggda klassen Math för att göra olika uträkningar.

Exempel:

```
Math.max(10, 15);
Math.min(10, -9);
Math.sqrt(9);
Math.random();
```

Testa själv!

Hur är ett Javaprogram uppbyggt?

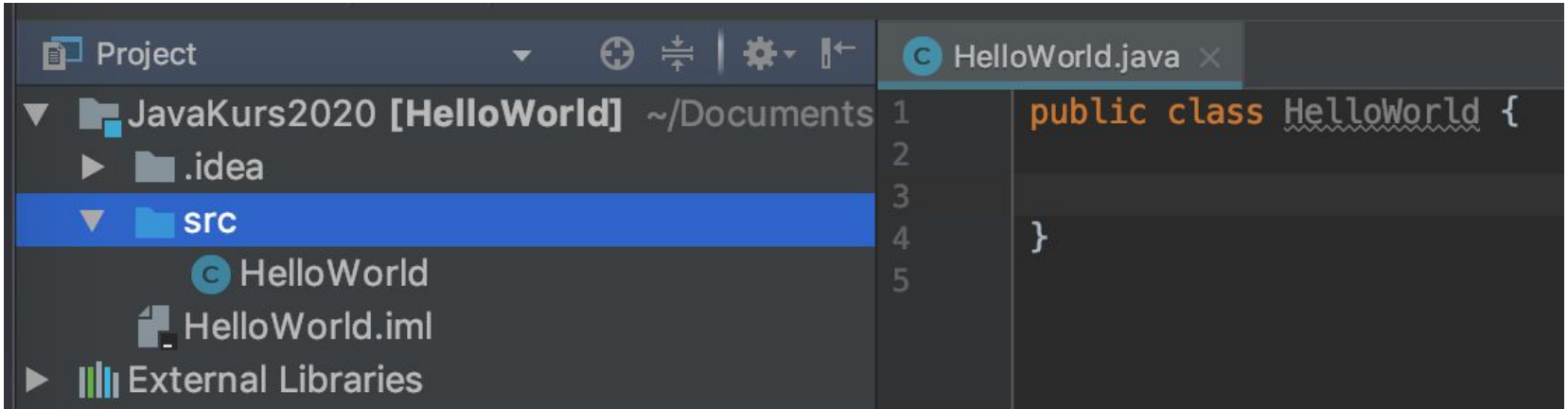
- En eller flera (.java-)filer, kallas för Klasser
- Varje klass börjar med att vi skriver:

```
public class KlassNamn {
}
```

- Inne i de två fiskmåsvingarna så programmerar vi vår klass
- Vi kan spara undan värden i variabler i klassen
  - Ex. `String name = “Kungen”;`
- Vi kan skapa miniprogram inom klassen som utför saker (kallas metoder)
  - Visar snart!

Vårt första Java-program

- Nu lämnar vi JShell och öppnar vår IDE som heter IntelliJ IDEA
- Vi startar ett nytt projekt
- Se till att Project SDK är ifyllt (versionen av Java), sen next
- Välj namn “HelloWorld”
- IntelliJ startar upp, till vänster är alla våra mappar och klasser (inga klasser än)
- Skapa en klass, högerklicka på src -> New Java Class -> HelloWorld



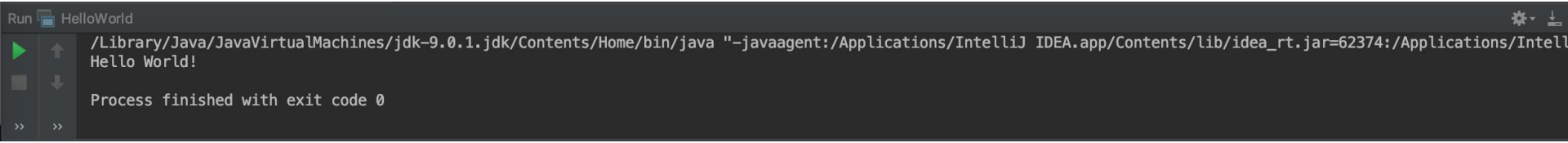
Vårt första Java-program

- Nu har vi vår första klass, som ni ser heter den HelloWorld
- Nu behöver vi berätta för Java var programmet ska börja köras, då behöver vi en *metod* som heter main. Det är där som Java-programmet kommer börja läsa kod.
- När vi skapat main-metoden så måste vi göra något i den, målet är att programmet ska skriva en text till användaren i konsolen
- Kommentarer skrivs såhär `// Allt efter kommentaren ignoreras av Java`
- För att skriva en text till användaren så skriver vi:

```
System.out.println(“Hello World!”);
```

OBS! Glöm inte att avsluta alla kommandon med semikolon;

```java
public class HelloWorld {
    // Detta program skriver ut texten "Hello World!" till användaren
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```



Vårt första väldigt simpla Java-program!

Klasser

- Vad är en klass? Java är objektorienterat och det betyder att nästan allt vi arbetar med i språket är klasser eller objekt som i sin tur är skapade med hjälp av en klass
- Tänk exempelvis att vi har en klass som heter “Person” som vi har sagt kan ha ett namn och en ålder (genom att lägga in det som variabler i klassen)
- Vi kan då skapa ett objekt som är en “Person” och ge den specifika personen ett namn och en ålder genom att skriva

```
Person m = new Person(“Alfred”, 16);
// m är en variabel där vi sparar undan en Person (Alfred)
```

- Vi testar att göra klassen Person tillsammans för att få bättre förståelse

Klasser

- För att säga till Java hur vi skapar ett Person-objekt så måste vi ha en konstruktor
- I klassen Person skriver vi helt enkelt:

```
public Person(String name, int age) {
    this.name = name;
    this.age = age;
}
```

- Detta gör att Java förstår att vi vill ge varje Person ett eget namn & ålder
- Nu kan vi skapa personer med hjälp av vår klass!

Objekt

Nu när vi har en klass “Person” så kan vi enkelt skapa hur många objekt vi vill som är Personer. För att göra det så skriver vi (i vår main-metod):

```
person1 = new Person(“Jocke”, 23); // Skapar en människa
person2 = new Person(“Karl”, 12); // Spararas i variabel
```

Men vad ska vi göra med alla Personer vi har? De har ingen funktion ännu

Metoder

- En metod är som ett litet miniprogram i din klass, som kan köras. Man kan också tänka att metoderna är alla handlingar en klass kan utföra
- En metod kan ta in värden och returnera värden
- Main i HelloWorld-klassen är en metod
  - Den är en speciell metod som berättar för Java var den ska börja läsa
- Just nu kan Personer inte göra någon form av ‘handling’ så vi skapar en metod för att de ska kunna hälsa
- En metod i en klass kan bara användas om vi skapat ett objekt av den klassen (undantag: static). För att använda metoden så skriver vi:

```
objektnamn.metodnamn();
```

Metoder

Vi testar att skapa två metoder för Person (läggs i public class Person):

```
public void greeting() {
    System.out.println(“Hej jag heter ” + name);
}

public int getBirthyear() {
    return(2020 - age);
}
```

Objektorienterad programmering

Det vi just gjort är objektorienterad programmering. Det handlar om att allt vi programmerar med är objekt som tillhör klasser. De har metoder och olika värden/egenskaper.

Det finns mer att lära om objektorienterad programmering, till exempel så kan klasser ärva egenskaper från andra klasser.

Exempel: Vi har en klass Däggdjur och en klass Fåglar. Båda är djur, vilket betyder att dom båda kan ärva egenskaper från klassen Djur. Alla djur har ju vissa gemensamma egenskaper, som att de kan äta, låta, föröka sig osv. Men vi går vidare nu!

Importera andra klasser till Java

Java har några inbyggda klasser, t.ex. String är en klass som är inbyggd. Även Math. Men det finns många fler som vi enkelt kan importera.

Låt säga att vi vill generera slumptal på ett mer korrekt sätt i Java. Då importerar vi högst upp i programmet en random-klass.

```
import java.util.Random; // Importera Random-klassen
```

Sen i main-metoden:

```
Random rand = new Random(); // Skapa random objekt
int rand_int1 = rand.nextInt(1000); // Generera ett random tal
System.out.println("Random int: " + rand_int1); // printa
```

Indata

Nu ska vi lära oss hur vi kan be användaren om indata, d.v.s. ställa användaren en fråga och få ett svar av användaren.

- System.out.println(); använde vi för att ge text till användaren, vad är motsvarigheten när vi vill få text från användaren?
- För att få indata så använder vi oss av något som kallas för en Scanner
- Vi importerar först Scanner högst upp i vårt program

```
import java.util.Scanner;
```

- Sen så skapar vi ett Scanner-objekt, och ber användaren om något

```
Scanner reader = new Scanner(System.in);
System.out.println("Enter a number: ");
int n = reader.nextInt();
System.out.println(n * 10);
```

Gemensam uppgift - ett program som hälsar på användaren

Vi vill göra ett program som hälsar trevligt på användaren. Det ska:

1. Fråga vad användaren heter
2. Säga “Hej på dig [NAMN]!”
3. Be användaren säga hur gammal den är
4. Säga “Du är [ÅR] gammal [NAMN]. Då är du nog född [ÅR]”

Gemensam uppgift - lär dig objektorienterat

En uppgift för att lära in hur objektorienterad programmering fungerar.

Skapa 2 klasser

- Bil.java
  - Ska kunna ha ett registreringsnummer, en årsmodell och ett märke
  - Ska ha en konstruktor så vi kan skapa Bil-objekt från klassen Bil
  - Ska ha en metod ‘printInformation’ som printar ut all information om bilen på ett snyggt sätt
  - Ska ha en metod ‘getInformation’ som returnerar informationen utan att printa den
- Main.java
  - Ska ha en main-metod (likadan som den vi hade i det första programmet)
  - Ska i main-metoden skapa 3 stycken bilar av olika model, årsmodell och registreringsnummer
  - Ska sen printa ut information om alla de bilarna med hjälp av ‘printInformation’

Egen uppgift - väldigt basic miniräknare

Ett program som multiplicerar (gångrar) ihop 2 tal
Skapa 1 klass

- Calculator.java
  - Ska ha en main-metod (likadan som den vi hade i det första programmet)
  - Ska i main-metoden ha en Scanner (måste importeras högst upp)
  - Ska be användaren om ett tal och spara i en variabel (tal1)
  - Ska be användaren om ett till tal och spara i en variabel (tal2)
  - Printa det första talet gånger det andra talet (tal1*tal2)

Jämförelser mellan element

- Jämför om två värden är samma
  - `1 == 1 => true`
  - `1 == “1” => error`
  - `10 == 100 => false`
  - `“Hej”.equals(“hej”) => false`
  - `“halloj”.equals(“halloj”) => true`

- “Jämför om ett element är större/mindre än ett annat
  - `3 > 5 => false`
  - `5 > 3 => true`
  - `“Sträng” < 100 => error`
-

If-satser

För att göra olika saker beroende på jämförelser av värden i vårt program så använder vi if-satser.

Exempel:

```
Scanner reader = new Scanner(System.in);
System.out.println("Hur gammal är du? : ");
int age = reader.nextInt();

if (age > 17) {
    System.out.println("Välkommen in");
} else if (age > 14){
    System.out.println(“Du får åka moppe, men inte komma in på klubben”);
} else {
    System.out.println("Tyvärr, du är inte gammal nog");
}
```

If-satser

Ibland vill man kontrollera fler saker samtidigt i en if-sats

```
int age = 23;
String name = “John”;

if (age > 17 && name == “John”) {
    System.out.println("Du är vuxen och heter John");
} else {
    System.out.println("Du är inte en vuxen person som heter John");
}
```

Äventyrsspel

Med hjälp av if-satser kan vi skapa äventyrsspel där vi frågar användaren vad den vill göra:

```
String name = scanner.nextLine();
System.out.println(“Hej ” + name + “, vill du gå in i huset?”);
String answer = scanner.nextLine();
if (answer.equals(“ja”)) {
     System.out.println("Du är nu i huset. Vill du öppna källardörren?");
     answer = scanner.nextLine();
     if (answer.equals(“ja”)) {
          System.out.println(“Ett monster attackerar dig”);
     } else {
          System.out.println(“Du överlevde. Fort ut ur huset nu!”);
     }
} else {
     System.out.println("Du gick inte in i huset, tur du överlevde!");
}
```

Egen uppgift- Gissningsspel

Med hjälp av Random och if-satser så kan vi göra ett spel där användaren får gissa ett slumptal, och låta användaren veta om det var rätt eller fel svarat. Försök själv!

- Skapa en klass: Gissningsspel.java
  - Skapa en main-metod
  - Generera ett slumptal (importera java.util.Random eller använd Math.random()) och spara undan i en variabel
  - Importera och skapa en Scanner för att kunna få input
  - Be användaren gissa på slumptalet
  - Printa ‘grattis’ om det var rätt, säg annars att det var fel gissat, och printa det rätta svaret

Fortsätt lära dig på egen hand

https://www.w3schools.com/java/

https://www.w3schools.com/java/exercise.asp?filename=exercise_syntax1

Cheat Sheet

1. // [comment] Single line comment.
2. /* [comment] */ Multi line comment.
3. public This can be imported publically.
4. static Going to be shared by every [object].
5. final Cannot be changed; common to be defined with all uppercase.
6. double/float: Integer with numbers that can have decimals.
7. ; Put after every command.
8. String Just a string of characters.
9. int Can store numbers from 2^-31 to 2^31.
10. fields are attributes
11. boolean Can have true or false as the value.
12. { } These are used to start and end a function, class, etc.
13. char Just lets you put in one chracter.
14. Scanner This lets you get user input.
15. new [object constructor] This will let you create a new object.
16. System.in This lets you get data from the keyboard.
17. public [class]() This will be the constructor, you use it to create new objects.
18. ++ Will increment the amount.
19. -- Will decrement the amount.
20. += [amount] Increment by [amount]
21. -= [amount] Decrement by [amount]
22. *= [amount] Multiply by [amount]
23. /= [amount] Divide by [amount]