1 Semester SWK Modul 09-10

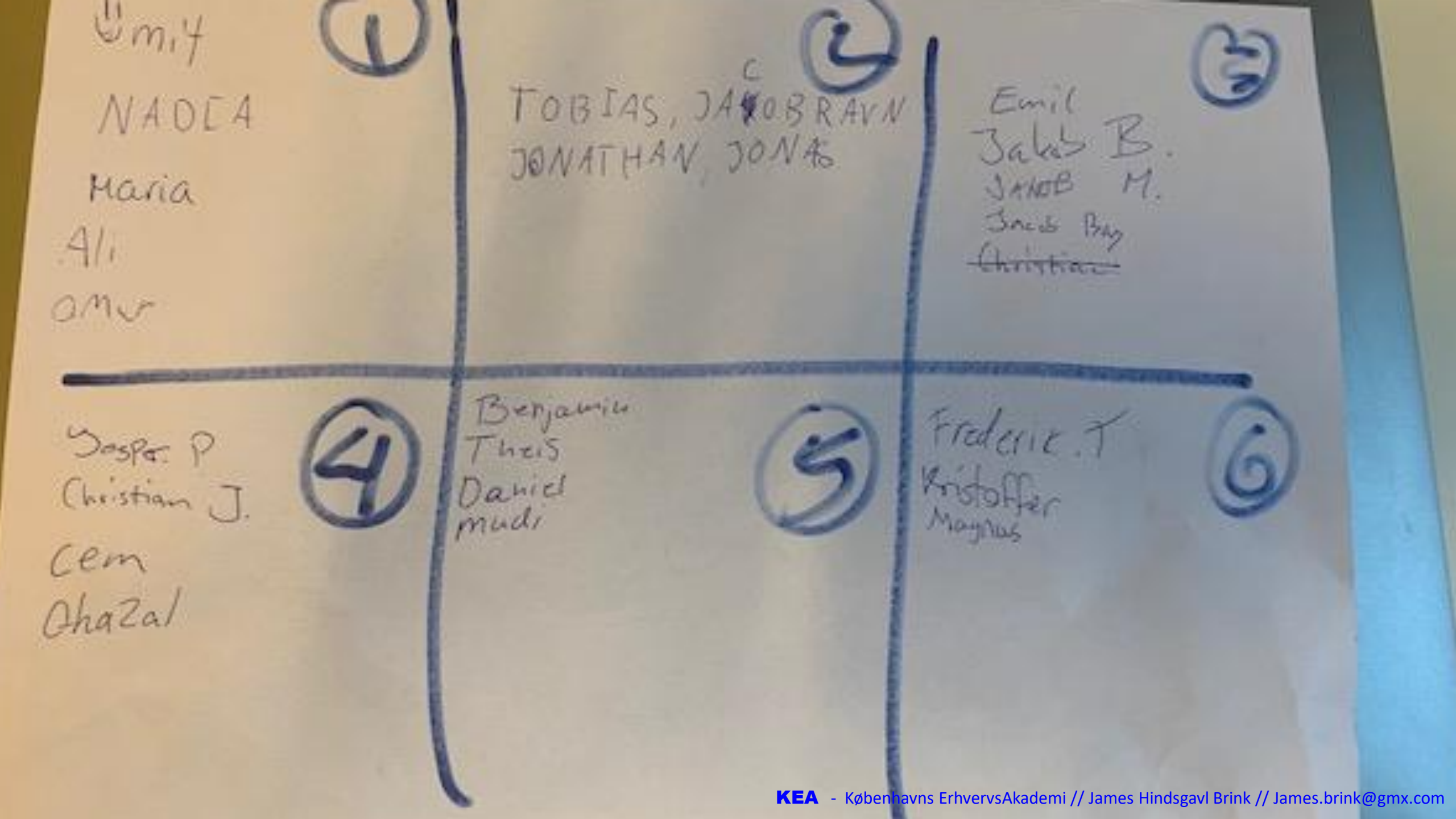
Formål:

At lære om **While** og **Do/While** og se på begreberne **Fencepost** og **Sentinel**
At lære **Scanner Class** at kende – vi ser på input fra **Console**

Emner:

- Simple typer
- Boolske operationer &&, ||, <=, =>=, !=
- **While**
- Klasser, metoder, konstruktører, parametre og returværdier, import
- Encapulation, Private, This, Getter,Setter
- **Scanner**

Litteratur: Reges kap. 5
Reges kap. 3


Emil
NADIA
Maria
Ali
Omar
1
TOBIAS, JACOB RAVN
JONATHAN, JONAS
2
Emil
Jakob B.
JAKOB M.
Jacob Bay
~~Christian~~
3
Jasper P
Christian J.
Cem
Ghazal
4
Benjamin
Theis
David
mudi
5
Frederik T
Kristoffer
Magnus
6

Repetitions øvelse

6 grupper præsenterer oplæg

1. Vis kode med **Klasser** og **Objekter** og forklar hvad der sker undervejs i koden – **new()**. Forklar også hvad en **metode** er, og hvad ideen med metoder er.

2. Forklar hvad **Getter** og **Setter** er. Forklar hvordan **paramtre** og **returværdier** bruges og vis med kodeeksempel hvordan syntaksen er.

3. Hvad er en **Konstruktør**? Hvad er **Overloading?** Vis kodeeksempler hvor **new()** anvendes.

4. Forklar hvad **Encapsulation** går ud på. Hvad er **private**? Og hvorfor findes **this.** Og skal den altid bruges?

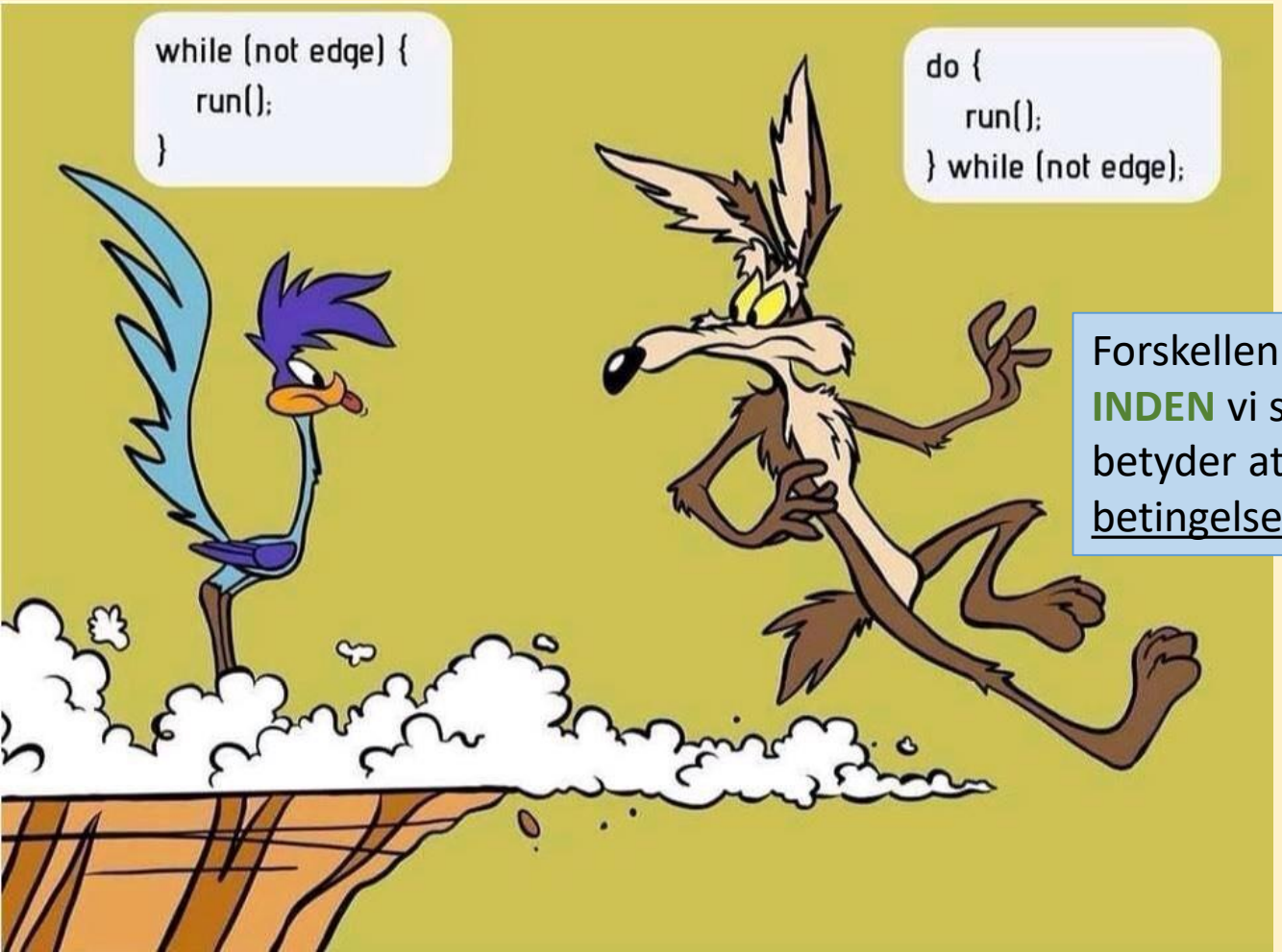
5. Forklar hvad **If..Then..Else** gør i vores kode og gennemgå en forklaring. Kom ind på **boolske udtryk** – forskellen på fx **a=5** og **a==5**

6. Hvad er **For/Next** for en konstruktion og vis en stump kode med et **nested loop**. Forklar kort hvad **Random** er og hvor vi får objektet fra - **import**.

While og Do/While

Vi har tidligere set på Forløkken. I Forløkken så vi at antallet af gennemløb gives på forhånd,
i While konstruktionen er antal loops er ukendt!!

Vi gentager et While-loop sålænge betingelsen (Vores SENTINEL) siger det…



Forskellen på **While** og **Do/While** er at betingelsen for loopet chekkes **INDEN** vi starter løkken ved While og EFTER løkken ved Do/While. Det betyder at en Do/While altid **gennemløbes mindst en gang** – Uanset betingelsen for loop-konstruktionen!

KEA - Københavns ErhvervsAkademi // James Hindsgavl Brink // James.brink@gmx.com

While

```
While ( [Condition]){
   <KODE der køres til betingelsen er false>
}
```

Operatorer til Boolske udtryk

Operator	Result
&	Logical AND
|	Logical OR
^	Logical XOR (exclusive OR)
||	Short-circuit OR
&&	Short-circuit AND
!	Logical unary NOT
&=	AND assignment
|=	OR assignment
^=	XOR assignment
==	Equal to
!=	Not equal to

Betingelsen kan være **Primitiv (Int,Short,Boolean…)**
Men kan også være en **String**, eller **en metodes return-parameter**
BEMÆRK: Loopet kører så længe betingelsen er true. *Det er nemt at lave uendelige løkker!!!☺*

Operatorer der kan bruges på alle **typer**

Condition	Operator	Example
Is equal to (or "is the same as")	==	int i=1; (i == 1)
Is not equal to (or "is not the same as")	!=	int i=2; (i != 1)
Is less than	<	int i=0; (i < 1)
Is less than or equal to	<=	int i=1; (i <= 1)
Is greater than	>	int i=2; (i > 1)
Is greater than or equal to	>=	int i=1; (i >= 1)

```
while (dice!=6){
        cnt++;
        dice = r.nextInt(6)+1;
        if (dice==6){
           System.out.println("Så fik vi en sekser! på "+cnt+" forsøg2");
        }
```

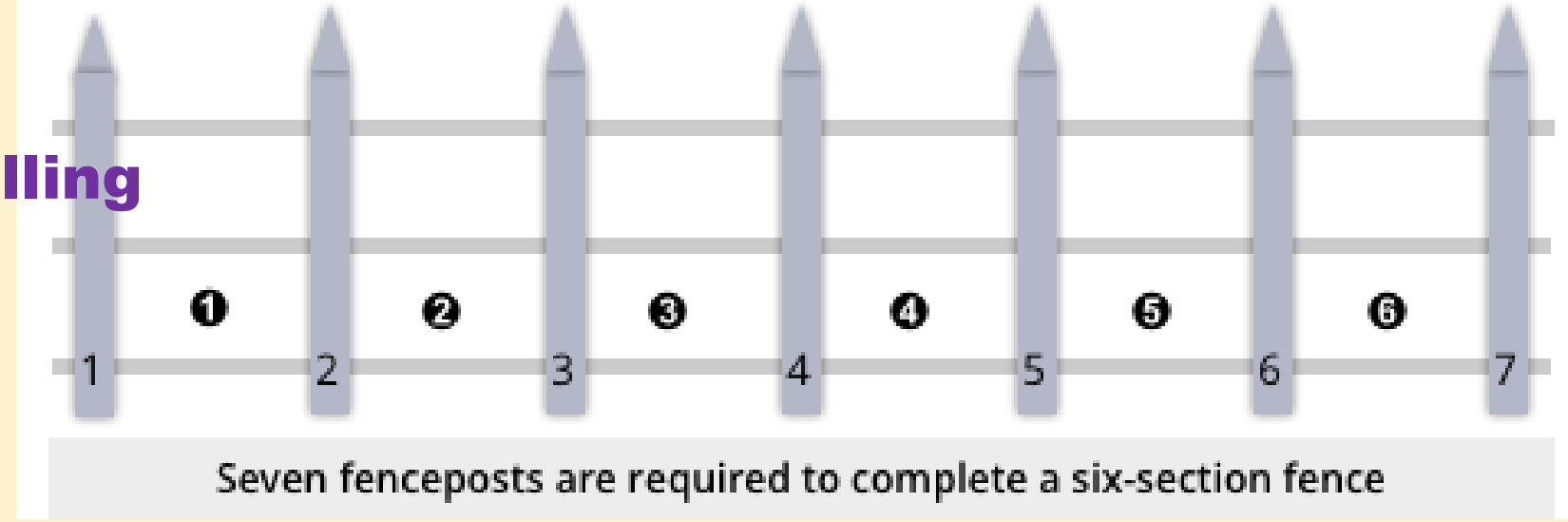
Do/While

```
Do {
   <KODE der køres >
} While ( [Condition]);
```

Betingelsen er som på forrige slide. Den væsensforskel der er på de to konstruktioner er at koden der ligger i løkken **altid afvikles mindst 1 gang!** (vi møder betingelsen første gang efter koden er kørt – derfor afvikles koden uanset og betingelsen er opfyldt eller ej)

```
 do {
    cnt++;
    dice = r.nextInt(6)+1;
    if (dice==6){
       System.out.println("Så fik vi en sekser! på "+cnt+" forsøg2");
    }
}while (dice!=6);
```

Fencepost
– en klassisk problemstilling



Seven fenceposts are required to complete a six-section fence

Når vi koder oplever vi ofte at vi vil flere ting i loopet, men ikke hele tiden!
Fencepost er et klassisk problem hvor et loop skal sætte et hegn!

Vi starter med en stolpe, så noget trådnet. Og gentager processen.
PROBLEM: Vi ender med et trådnet UDEN den afsluttende stolpe.

I bogen er et andet eksempel hvor vi ønsker tallene fra fra 1 til 10 udskrevet med kommaer imellem. Vi ønsker selvfølgelig ikke at skrive et komma efter sidste tal!

Eksempel :

```
Plant a post.
For (the length of the fence){
  Attach some wire.
  Plant a post.
}
```

Sentinel Loops

Når vi opretter en loop-struktur hvor vi ikke leder efter en tæller der når et maks men efter en særlig markør taler vi om **Sentinel Loops** (Sentinel = Vogter) !

Det kan fx. Være at vi modtager løbende data i form af positive tal og vores sentinel kunne være **-1**

Eksempel :

```
Sum = 0.
While (We haven't seen the sentinel ) {
  Prompt and read.
  Add to the sum.
}
```

Scanner Klassen

Med **Scanner** klassen får vi, som vi så det med Random-klassen, endnu en smart klasse fra **Java-biblioteket** der benytter den hardware vi kører på

- SCANNER-klassen ***kan fx læse data udefra*** og ind i vores program
- Vi starter med at se på Scanneren og vores konsol
  Det vil sige vi opretter et **ScannerObjekt** og lader det hente data
  fra vores konsol, hvor vi som brugere nu kan indlæse data i vores program!
- ScannererKlassen tillader os at arbejde med datastrømme
  Og vi vil senere lære at konsollen og fil-I/O er eksempler på dette…

```
import java.util.Scanner          <-- 1. Import Scanner Class

public class ScannerDemo
{
public static void main(String args[])
{
                                  2. Construct Scanner class Object
Scanner s=new Scanner(System.in);
System.out.println("Enter first no= ");

int num1, num2;                   <-- 3. Define Variable to Receive Input

num1=s.nextInt();

System.out.println("Enter 2nd no:");   4. Read Input from Keyboard

num2=s.nextInt();
System.out.println("Sum of no is= "+(num1+num2));
}
}
```

Scanner syntax

- The `Scanner` class is found in the `java.util` package.

```
import java.util.*;  // Importér Klassen der rummer Scanneren
```

- Constructing a `Scanner` object to read console input:

```
Scanner name = new Scanner(System.in);
```

  - Example:

```
Scanner console = new Scanner(System.in);
```

Scanner(System.in)

```
import java.util.*;//UTIL for at anvende Scanner-klassen

public class InputXY{
  public static void main (String args[]){

    Scanner scan = new Scanner(System.in);//Create scanner Object;

    System.out.print("Type your age: ");
    int age = scan.nextInt();//get a number USERINPUT
    System.out.println("\n You are "+age+" years old");
  }
}
```

Bemærk: Ligesom vi så med Random-klassen opretter vi et objekt. MEN her kaldes konstruktøren med en parameter!!! Parameteren peger på den kilde vi ønsker at sætte os i forbindelse med og hente data fra: Her er det System.in. System er jo vores konsol som vi tidligere har sendt til med System.out….

Bemærk: Ligesom vi så med Random-klassen har vores scanner-objekt en række metoder. Her bruger vi **nextInt** – en metode der går til konsollen og afventer at læse data – OG vel at mærke et heltal (indtastes andet fejler programmet)

Methods	Description
public String next()	it return the next token from the scanner
public String nextLine()	it moves the scanner position to the next line and returns the value as a string
public byte nextByte()	it scans the next token as a byte value
public short nextShort()	it scans the next token as a short value
public int nextInt()	it scans the next token as a int value
public long nextLong()	it scans the next token as a long value
public float nextFloat()	it scans the next token as a float value
public double nextDouble()	it scans the next token as a double value

Input tokens

- **token**: A unit of user input, as read by the `Scanner`.
  - Tokens are separated by *whitespace* (spaces, tabs, new lines).
  - How many tokens appear on the following line of input?

```
23  John Smith   42.0  "Hello world"  $2.50  "  19"
```

- When a token is not the type you ask for, it crashes.

```
System.out.print("What is your age? ");
int age = console.nextInt();
```

Output:

```
What is your age? Timmy
java.util.InputMismatchException
        at java.util.Scanner.next(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        ...
```

Scanner-klassen

– værd at vide!

Scanner Class

- The **java.util.Scanner** class is a simple text scanner which can parse primitive types and strings using Regular Expressions.
- Following are the important points about Scanner class:
  - A Scanner breaks its input into tokens using a delimiter pattern, which by default matches whitespace (blanks, tabs, and newline).
  - A scanning operation may block waiting for input.
  - A Scanner is not safe for multithreaded use without external synchronization.

Bemærk: Vi opretter objektet af Scannerklassen med en konstruktør som vi så på forrige slide. I det vi opretter objektet peger vi på det sted vi ønsker at modtage data fra – Her konsollen, men det kunne jo også – fx – være en fil. ☺

Øvelse: While While/do Scanner

- På det numeriske tastatur kan i se pile. Skriv nu et program hvor en spiller kan flytte sig rundt i et koordinatsystem på 10x10 felter! (start i 5,5)
- Der skal være et while-loop med en sentinel – tallet 5 som slutter.
  - Kig på num-tasterne: 4=tv, 7=tv+op, 8=op, 9=op+th osv.
- Hver gang du har ”tur” skrives på skærmen ”Din tur (5 for stop).:”
- Indlæs brugerdata og flyt koordinaterne efter input.
  - Hvis brugeren står i kanten af banen ignoreres koordinater der er udenfor banen og sættes til min/max alt efter hvor man står.
- Når rykket er udført skrives ”Du er nu på: (x,y)”

1 Semester SWK Modul 09-10

Formål:

Repetere While og Do/While og se på begreberne Fencepost og Sentinel
Repetere Scanner Class at kende – vi ser på input fra Console
Bemærk: Vi arbejder hele tiden med Objekter, klasser og metoder og konstruktørerg parametre er stadig en udfordring så det arbejder vi med i alle øvelser!!!

Emner:

- Simple typer
- Boolske operationer &&, ||, <=, =>, !=
- While
- Klasser, metoder, konstruktører, parametre og returværdier, import
- Encapulation, Private, This, Getter,Setter
- Scanner

Litteratur:

- Reges kap. 5
- Reges kap. 3

NÆSTE GANG i Modul 10