

Formål:

Fortsat fra modul 03: at forstå lidt mere af hvad der foregår når vores objekter oprettes: Vi ser på **Private**, hvad det er, og herunder: **Gettere og Settere**. BEMÆRK: vi har en lille demo af hjælpeordet: **This**

Fokus i dag er dog begrebet ReturParameteren i Metoder.

Emner: Return value eller Returparameter

Litteratur: Reges: Kap.: 3 Fokus: 3.1, 3.2
Regés: Kap.: 8 Fokus: 8.3

HUSK at læse pensum – og ikke bare det vi fokuserer på!
Vi gennemgår det udfordrende og finder du i pensum elementer du ikke forstår som bring det op i klassen.

Metoder med Parametre

I OOP ønsker vi at være stramme og strukturerede i vores logik og måden vi håndterer objekterne. Det betyder at vi ønsker at vores **objekter kun tilgås via vores metoder**. BRUG **private** til at beskytte jeres variable!

På den måde ved vi som udviklere altid, hvad der sker med objektet og hvordan det ændres.

Med indførelse af parametre kalder vi objektets metoder med værdier, som vi så kan arbejde med i vores objekt.

Og med returværdier i metode vil vi se eksempler på hvordan vores kode bliver forenklet, mere intuitiv, lettere at læse!

Access Levels

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

Kodeeksempel med returparameter

```
public class Dog{
```

```
    String name = "<blank>";  
    String color = "<hollow>";  
    int age = -1;  
    double weight = -1;  
    String barkingSound = "Silent";
```

```
    public Dog(){  
        name = "Ikke oplyst";  
    }
```

```
    public Dog(String name){  
        this.name = name; //BEMÆRK: this bruges til at skelne objektets variabel name fra metoden paramter da de har samme navn!!!  
    }
```

```
    //metode til give en tekststreng af hundedata
```

```
    public String getDogData(){  
        return "Navn:" + name + "\n Farve:" + color + "\n Alder:" + age + " Aar \n Vaegt:" + weight + " kg \n";  
    }
```

```
    public int getage(){  
        return age;  
    }
```

BEMÆRK: VIGTIG at kende **THIS**
Den er normalt underforstået –
men her skal den bruges for at
sikre forskellen mellem
parameteren i metoden og
objektets egen variabel da de
begge hedder name 😊

Metoden returnerer en streng
BEMÆRK: Når kommandoen return
anvendes er det afslutningen på
metoden. Der kan altså ikke køres flere
kodelinier i metoden efter return er kørt!

Returparameter

Når vi kalder en metode med returparameter vil vi på en enkelt måde kunne skrive og arbejde med kode. Fx. Kan en metode returnere et tal.

Fx en metode Sqrt der udregner kvadratroden og returner denne....

```
public double sqrt(double x){  
    return Math.sqrt(x);  
}
```

- En metode der ikke er 'void' skal altid slutte med en return-kommando
- En metode med returværdi bruges som den variabel-type (attribut) den returnerer. Returneres en boolean bruges metoden som logisk udtryk, er det en int bruges den som et heltal osv.
- Med parametre og returværdi bliver vores metoder pludselig meget kraftfulde og i ser nu hvordan objekter kan styres udelukkende gennem deres metoder.

Øvelse:

Skriv en stump kode der indeholder en metode der returnerer en værdi til hovedprogrammet. Sæt jer to og to og udtænk programmet. I skal hver i sær skrive koden selv på jeres egen PC.

1. Kod et Java projekt med en klasse
2. Variable skal oprettes med private
3. Opret en konstruktør der kan kaldes med en eller flere parametre
4. Opret en metode der kan sætte variable (en setter).
 - Benyt evt. samme navne i parameterkaldet og brug **this**-kommandoen til at skelne
5. Opret mindst en metode med returværdi
 - Benyt evt. også metoder fra matematik-klassen – se i bogen 😊
6. Opret nu et hovedprogram som gør brug af klassen og dine metoder

Repetition

Vi har nu gennemgået de indledende og mest centrale begreber i OOP vedrørende Klasser, objekter og metoder.

- Hvad er en **Klasse**?
 - En skabelon der definerer identifiere og methods som objekter skabes ud fra
- Hvad er et **Objekt**? (uddyb forskellen på Class og Object)
 - Når en identifikator skabes med **new** kommandoen skabes et object efter den givne classes anvisning. Objektet er den forekomst der kan arbejdes med, hvor klassen bare er en model for objektet. Fx:
`Dice dice1 = new Dice();`
- Hvad er en **Metode**?
 - Metoder er en række handlinger der kan kaldes, hvorefter metoderne så foretager noget med objektet.
- Hvad er en **Konstruktør**? (hvad adskiller Constructor fra Method)
 - En konstruktør kan erklæres og vil så blive udført i forbindelse med 'NEW'kommandoen. Typisk bruges konstruktøren til at initialisere data med en given startværdi.

Note: På nuværende tidspunkt skal man have en **viden** om, hvad begreberne dækker over, og have en **grundlæggende forståelse** af konceptet med at **oprette og arbejde med klasser og objekter**.

Diskutér emnerne, læs litteraturen og tal med din underviser hvis der er begrebsmæssig forvirring.

Repetition

Vi har nu gennemgået de indledende og mest centrale begreber i OOP vedrørende Klasser, objekter og metoder.

- Hvad er en **private variabel**?
 - En variabel der erklæres private er beskyttet mod direkte tilgang. Kun objektets metoder kan se den
- Hvad menes med **Getter og Setter** ?
 - Metoder der oprettes for at hente (en getter) en variabel, henholdsvis sætte variabelen kaldes gettere og settere, men er blot metoder som vi kender allerede.
- Hvad gør **this**?
 - This kommandoen sikrer at vores objekts variabel kan skelnes fra variabelen i metoden, da de belejligt hedder det samme. Med **this.var1** henvises til objektes egen variabel, **var1** mens var1 er den der optræder i metodens parameter;
- Hvad betyder **Encapsulation**?
 - Encapsulation, eller indkapsling, er når vi isolerer objektets variable så de ikke kan tilgås direkte. Kommandoen private er en måde at sikre dette.

Note:

På nuværende tidspunkt skal man have en **viden** om, hvad begreberne dækker over, og have en **grundlæggende forståelse** af konceptet med at **oprette - og arbejde med - klasser og objekter**.

Diskutér emnerne, læs litteraturen og tal med din underviser hvis der er begrebsmæssig forvirring.

Formål: At udvide vores kode med betingelser og løkker i de objekter og metoder som vi nu kender.

Emner:

Conditional Execution

- IF..Then
- For...Next

**NÆSTE GANG
i Modul 5&6**

Litteratur:Reges: Regés: Kap.: 4 Fokus på 4.1 + 4.2

Med parametre i metoderne undgår vi at skulle tilgå vores datastrukturer direkte. Vi ønsker kun at arbejde gennem metoder når vi håndterer objekter! Med parameterbegrebets introduktion bliver konstruktøren pludselig interessant og begrebet Overloading indføres. Returparameter bidrager til fleksible løsninger når vi arbejder med metoderne.