1 semester SWK Modul 03

Formål:

At se på begrebet **Parameter** i **Metoder**.

At forstå lidt mere af hvad der foregår når vores objekter oprettes:
Vi ser på **konstruktøren**, hvad det er og introducere endnu et begreb: **Overloading** samt **Private** kommandoen

Emner: Parameter og Metoder

Litteratur: Reges: Kap.: 3 Fokus på 3.1
Regés: Kap.: 8 Fokus på 8.3, 8.4

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**.

Vi kan sikre os mod 'snyd' med det reserverede ord **private**

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.

Fx kan vi sikre at hvis et metodekald til et objekt kalder med en værdi der overskrider grænsen for vores interval kan vi vælge at objektets værdi tilrettes til max. Og ikke den værdi der kaldes med.

Encapsulation

Private – vores variable er nu indkapslede!!!

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

```
//definer hundeklasse (Vi ser på PARAMETRE CONSTRUCT og OVERLOAD samt PRIVATE)
public class Dog{

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

    //Konstruktøren oprettes public Hund = klassenavnet



    public Dog(){
      //konstruktøren overskriver name som er sat til at være <blank>
      name = "Ikke oplyst";
    }
```

BEMÆRK:
Private er det nye reserverede ord vi nu kan bruge.
Når vi skriver private foran en variabel betyder det at disse variable indkapsles – dette kaldes ENCAPULATION og gør at variablerne slet ikke kan ses uden for objektet.
Vi bruger nu udelukkende metoder til at hente eller sætte værdier for disse variable. Metoder der gør dette kaldes ”Get’ere og Set’ere’ fordi vi typisk vil navngive metoderne med getParamX eller setParamX.

Kodeeksempel med parametre

```
//Metode til at sætte hundeparametre
public void setDogData(String newname, String newcolor, int newage, double newweight, String newbarkingsound){
    name = newname;
    color = newcolor;
    age = newage;
    weight = newweight;
    barkingsound = newbarkingsound;
}

//metode til hundelyd
public void dogBarking(){
    System.out.println(name + "\t siger:\t" + barkingsound);

}
//hundelyd overloadet - nu med mulighed for at definer ny lyd
 public void dogBarking(String newbark){
    barkingsound = newbark;
    System.out.println(name + "\t siger nu:\t" + barkingsound);

}
```

Bemærk:
To metoder med SAMME navn
Én med og en uden parametre.

Constructor – Det i har savnet indtil nu ☺

Når et Objekt erklæres, vil det ofte være bekvemt at kunne oprette det i en **begyndelsestilstand**. Vi har tidligere set med eksempel at **variable erklæres med en forud defineret værdi.** Med en konstruktør kan objektet startes op med forskellige startværdier. fx gem et userID på den medarbejder der opretter en motorcykel i systemet.

- **En konstruktør har eksakt samme navn som klassen!**
- **Der er altid en konstruktør! Hvis du ikke erklærer en vil den være der 'usynligt'!**
- **Det vil ofte være et ønske at benytte parametre i forbindelse med en konstruktør!**

Hovedprogrammet med constructor

```
//definer hundeklasse (Vi ser på PARAMETRE CONSTRUCT og OVERLOAD)
public class Dog{

String name = "<blank>";
String color = "<hollow>";
int age = -1;
double weight = -1;
String barkingsound = "Silent";
    //Konstruktøren oprettes public Hund = klassenavnet

    public Dog(){
      //konstruktøren overskriver name som er sat til at være <blank>
      name = "Ikke oplyst";
    }
    // BEMÆRK : Her kommer der endnu en konstruktør? OVERLOADING
    // Når vi ønsker at kunne bruge samme metode/konstruktør med forsekllige parametre
    // På denne måde kan vi oprette hunde-objekter både med og uden navn
    public Dog(String newname){
      name = newname;
    }
```

Overloading

Når en metode kan kaldes med forskellige parametre og evt. forskellige returværdier!

- Det kan være smart at kunne kalde en metode både med og uden parametre.
- JAVA understøtter Overloading som betyder at en metode – eller konstruktøren, kan kaldes med forskellige parametre.
- I praksis oprettes en metode blot én gang til, men med nye parametre (Det er et krav da JAVA jo ellers ikke kan skelne hvilke af de to metoder der skal kaldes på et givent tidspunkt)
- Fx kan man så oprette et object med 'NEW'kommandoen uden parametre og andre gange vælge at sætte parametre i oprettelsen.

Øvelse: ”Kitchen Stove Flame”

- Kod et Java projekt med en klasse for køkken og en for komfur og en for et blus på komfuret!

1. Hvilke variable skal Køkkenet have (fx 220volt) og et komfur.
2. Opret en konstruktør til komfuret der inititalieres med køkkenets netspænding.
3. Opret i komfuret fire blus af klassen Flame
4. I Flame oprettes en konstruktør der modtager blus-nr. (1..4)
5. Opret en metode der kan konfigurere blussets indstillinger. (Setter)
6. Opret en metode der kan udskrive komfurets aktuelle tilstand (spænding og niveauet på de fire blus)
7. Opret nu et hovedprogram som gør brug af dine metoder (Tænd blus 3 på indstillng 4)

BEMÆRK: Nu kan du ikke længere bare oprette Stove og Flame med new uden også at anføre parametre!!!

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 definer identifiers og methods som objekter skabes ud fra
- Hvad er et **Objekt**? (uddyb forskellen på Class og Object)
  - Når en identifier skabes med 'NEW' kommandoen skabes et object efter den givne klasses anvisning Objektet er den forekomst der kan arbejdes med, hvor klassen bare er en model for objektet.
- 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 initiailsere 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 variablen kaldes gettere og settere, men er blot metoder som vi kender allerede.
- Hvad gør **New**?
  - New er den kommando vi kalder når vi opretter et objekt af en given klasse: fx:
    Dice dice1 = **new** Dice();
- 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.

1 semester SWK Modul 03

Formål:

Vi fortsætter med konstruktører og overloading men fokuserer nu på begrebet **ReturParameter** i **Metoder**.

Emner: Returparameter

Litteratur: Reges: Kap.: 3 Fokus: 3.1, 3.2

Regés: Kap.: 8 Fokus: 8.3

NÆSTE GANG i Modul 4

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 biddrage til fleksible løsninger når vi arbejder med metoderne.