Anvendelse af metoder - Programmering

En forhåbentlig rigtig god forklaring på hvad metoder er og hvordan de anvendes. Lidt om private og public, retur typer og hvad jeg ellers kunne komme på. Forklaring på babysprog. Håber jeg.

Skrevet den **05. feb 2009** af **kalp** I kategorien **Programmering / Java** | ★★★☆☆

ARTIKEL ER HERMED GRATIS:) 27/5-05

Hvis du ikke har programmeret meget vil jeg anbefale du først læser http://www.eksperten.dk/artikler/600 om objekter for så vil du helt sikkert bedre kunne forstå denne artikel om metoder:) Vil sige denne er bygget videre på den anden.

Derfor starter vi ud med, at oprette en tom klasse vi vil kalde for "Person".

```
public class Person
{
  public Person()
  {
  }
}
```

public Person()

Er ikke en metode. Det er en konstruktør for klassen og på nuværende tidspunkt er den tom. Dvs. når man opretter et objekt af klassen Person er der ikke noget kode, som først skal blive kørt.
Hvordan kan man se det er en konstruktør og ikke en metode? Det er faktisk ret simpelt. En konstruktør har samme navn, som klassen. I dette tilfælde hedder klassen "Person" og det samme hedder konstruktøren altså også. Yderliger har en konstruktør ingen retur værdi! Det betyder, at når man opretter et objekt af klassen Person så skal der ikke sendes noget information tilbage til den klasse, som vil oprette objektet. Med andre ord er forventes aldrig, at der sendes noget retur fra en konstruktør, men maks, at noget kode skal blive kørt. Det vil give mere mening så snart du har læst resten af denne tekst igennem:)

Hvordan ser skelletet til en metode så ud?

En metode har følgende punkter.

Punkt 1. Skal den være synlig for andre klasser? eller skal metoden kun kunne kaldes af klassen selv?
Til dette anvendes følgende
- public = synlig for alle klasser
- private = usynlig for andre klasser.

Punkt 2. Metoden skal have en retur type. Her afhænger dette af hvad man vil have der skal sendes tilbage til der hvor metoden bliver kaldt!
Til dette anvendes (uddrag)
- void = der skal ikke sendes noget retur, men kun afvikles den kode i metoden.
- String = der skal sendes en String retur

- int = der skal sende int retur
- Object = Der skal sendes et objekt retur. Her kan man også godt skrive "Person" hvis der skal sende et person objekt tilbage.

Punkt 3. Navn på metoden. Det bestemmer du helt selv! Men helst noget sigende! Det bedste er, at skrive det første ord med småt og efterfølgende ords første bogstav med stort. Prøv at holde det så kort som muligt og dog stadig sigende.
Eks. hentAlder, nytNavn.

Punkt 4. Parametre som man gerne vil have med når metoden kaldes. Det kan være metoden skal udregne summen af to tal så ville det være smart hvis man sendte de to tal med som parametre og så kan metoden udregne hvad summen giver og sende det retur:)

Nåh! men her får i nogen eksempler på forskellige metoder så det giver lidt mere mening.

Vi vil lave nogen public metoder i vores person klasse, metoder som sender data retur, metoder som kræver man sender parametre med og metoder som ikke sender noget retur altså lidt af hvert.

nytNavn();

Denne metode skal være public eftersom vi vil have, at alle andre klasser skal kunne se den. Den skal ikke sende noget data retur. Den skal have en parameter med nemlig en String (navnet man vil give personen).

```
public void nytNavn(String navn)
{
fornavn = navn;
}
```

for at dette skal virke må vi lige lave en attribut i vores person klasse som hedder fornavn. Mit ser sådan her ud

```
public class Person
{

private String fornavn;

  public Person()
  {
  }

  public void nytNavn(String navn)
{
fornavn = navn;
}

}
```

og grunden til, at der står private foran String er fordi jeg ikke vil have man skal kunne se den fra andre klasser. Hvis man vil ændre fornavn bliver man altså nød til, at kalde metoden nytNavn som er public og

altså derfor synlig.

public er altså punkt 1 i en metodes opbygning, void er punkt 2, nytNavn er punkt 3 og det i paranteserne er punkt 4.

Nu hvor vi kan give vores person et navn så lad os også lave en metode som kan sende navnet retur for eftersom vi har gjort fornavn private kan vi ikke få fat i det uden brug af en metode.!

En metode til dette ville kunne se ud som følgende

```
  public String hentNavn()
{
return fornavn;
}
```

public (punkt1) er stadig public fordi vi gerne vil have metoden er synlig for andre klasser. String er altså her punkt 2. Vi skriver string som retur værdi fordi vi vil sende fornavn tilbage når man kalder metoden og fornavn er, som vi kan se en String. hentNavn er punkt 3 og altså noget man selv bestemmer hvad skal hedde. De 2 tomme () fortæller bare der ikke skal sendes nogen parametre med når man kalder metoden. (parametre står for data/informationer)

Angående private metoder så er private mest anvendt til, at støtte andre metoder i samme klasser og er netop af den grund ikke synlige for andre klasser. Jeg er løbet lidt tør for ideer til et godt eksempel på private:(Jeg vil dog alligevel lave et meget latterligt eksempel blot for, at illustrere:)

Vi får brug for lidt nyt i vores person klasse. En ny attribut kaldet "alder". Og vi laver to metoder.

Den første metode skal hedder fodselsdag();
Den skal man kunne kalde når personen har fodselsdag.
Den skal ikke sende noget data retur.

Denne metode skal kalde en anden metode, som er private! den anden metode sætter alderen op med 1. Vi kalder vores private metode for nyAlder();

```
public void fodselsdag()
{
nyAlder();
}
```

nyAlder() skal se ud som følgende.

```
private void nyAlder()
{
alder = alder + 1;
}
```

for ikke at miste overblikket får i her det samlede billede.

```
public class Person
{

private String fornavn;
private int alder;

  public Person()
  {
  }

  public void nytNavn(String navn)
{
fornavn = navn;
}

  public String hentNavn()
{
return fornavn;
}


public void fodselsdag()
{
nyAlder();
}


private void nyAlder()
{
alder = alder + 1;
}

}
```

Her er nyAlder() metoden selvfølgelig total overflødig for man kunne bare nøjes med at skrive alder = alder + 1; i fodselsdag, men det blot for at illustrere man stadig kan ændre værdien i sin attribut via. en private metode, men den kan ikke tilgås direkte fra andre. Det kan altså kun ske indirekte via. public metoder. Som fodselsdag.

Lad os gøre det lidt interessant. Vi må lige lave det så man skal indtaste et navn og en alder når man opretter en person og det gøres i konstruktøren.

så vores

```
  public Person()
  {
  }
```

skal ændres til

```
  public Person(String navn, int enalder)
  {
fornavn = navn;
alder = enalder;
  }
```

her tager man de parametre der bliver sendt med når objektet skal oprettes og gemmer deres værdier i de rigtige attributter.
Fældes for konstruktøren og metoder er, at de navne man giver sine attributter mellem parenteserne gerne må hedde præcis det samme som de øverste attributter i klassen!
Man må altså gerne skrive

public Person(String fornavn, int alder)

De vil nemlig blive betragtet som local variabler. Dvs de bliver oprettet når metoden kaldes og forsvinder igen når alt koden i metoden er kørt. Problemet er bare det svært at se forskel på hvilke attributter der hører til metoden og hvilke der hører til klassen. Den simpleste måde at sikre sig man få fat i klassens attributer er ved at skrive this foran. Så skulle det gøres på følgende måde.

```
  public Person(String fornavn, int alder)
  {
this.fornavn = fornavn;
this.alder = alder;
  }
```

så this.fornavn og this.alder er altså de øverste attributter i klassen.

Nu er vores Person klasse faktisk færdig og der kan oprettes objekter af den!
Du kan teste det ved lige at lave en lille main klasse som denne.

```
public class Main
{
public static void main(string[] args)
{

Person enKopiAfPerson1 = new Person("Hans", 23);
Person enKopiAfPerson2 = new Person("Tine", 22);
}
}
```

Nu har du lavet to objekter af vores klasse(2 kopirer)
Prøv nu at kalde de forskellige metoder der er i klassen!
Hvis du anvender Jbuilder eller Jdeveloper kan du gøre følgende.

skrive

enKopiAfPerson1.

altså med punktum efter så kommer der en lille boks/menu frem. Her er alle metoder i klassen person præsenteret. Lig mærke til at nyAlder() metoden ikke er med i listen. Den er jo private:) Men de andre.. vores public metoder er med! Disse

fodselsdag()
hentNavn()
nytNavn(String navn)

De er skrevet med fed... alt det der er skrevet med tynd vil jeg ikke gå ind i. Kort fortalt er det metoder som er arvet fra klassen Objekt. En klasse alle klasser stammer fra. Altså også dem vi laver:)

Prøv at gå tilbage i vores person klasse og ændre

private String fornavn;

til

String fornavn;

prøv så at skrive

enKopiAfPerson1.

og du vil se at "fornavn" kommer med i listen. Og det er det vi gerne vil undgå:)
Det er derfor vi laver metoder til, at hente, ændre eller slette fornavn. Alt skal helst foregå via. metoder!

Håber du fik noget ud af at læse dette:) God programmeringslyst!
Ps. Problemer eller spørgsmål undervejs? Spørg endelig løs!

Tak til simonvalter... fejlen er rettet... den var lille men en fejl;) sådan er det at skrive i E's lille tekstvindue:o)

Kommentar af simonvalter d. 04. feb 2005 | 1

Udemærket. Jeg ville også gerne have set alle 4 slags visibility for metoder og brugen af javabeans konventionen ved navngivning af nogle metoder har sine fordele.

Kommentar af medions d. 01. feb 2005 | 2

En rigtig god gennemgang! - flot skrevet og det hele! - det er der helt sikkert mange der vil kunne gøre gavn af! Glæder mig til at se flere artikler fra dig!

//>Rune

Kommentar af visualdeveloper d. 01. nov 2005 | 3

jeg synes også at det er en godt artikel...
bare der dog var nogle flere af disse slags artikler på eksperten...
bare skriv løs kAlp :P

Kommentar af alister_crowley d. 01. feb 2005 | 4

lovely :) specielt "Punkt 2" har jeg manglet forklaring på længe.
Danske forklaringer på nettet, er en absolut mangelvare (specielt ved java).

Kommentar af schwarz84 d. 22. jun 2005 | 5

normalt bruger man hhv getNavn(String navn) og setNavn(String navn) til at tilgå og ændre variablen navn, pyt.
Derudover er indentering en dyd, også i små eksempler. Det gør koden mere læselig og når man skal lære fra sig er det en god ting.
I dit sidste eksempel skal String[] være med stort i main-metoden. Det compiler ikke med lille.
Ellers en god artikel.

Kommentar af califfo d. 20. dec 2005 | 6

Guld værd. Andet kan jeg ikke sige. Jeg vil dog give "schwarz84" ret i at indentering er en dyd. Det tog lige et øjeblik at finde ud af hvor de forskellige klammer hørte til. Jeg er også meget enig i at det ville hjælpe med flere af den slags artikler for alle os grønne udviklere.