Mats Vik Pettersen
Programmering 2

Oblig_2

Klasse

En **klasse** er en mal eller en blåkopi for å lage objekter. Den definerer hvilke **egenskaper** (variabler) og **oppførsel** (metoder) objektene skal ha. I Java opprettes en klasse med class-nøkkelordet.

```
class Bil {
    String farge;
    int hastighet;

    void kjør() {
        System.out.println(x:"Bilen kjører.");
    }
}
```

Her er Bil en klasse som definerer to variabler (farge og hastighet) og én metode (kjør()).

Objekt

Et **objekt** er en faktisk instans av en klasse. Når vi lager et objekt, reserveres minne til det, og vi kan bruke dets variabler og metoder.

Eksempel på opprettelse av objekt:

```
Bil minBil = new Bil();  // Oppretter et objekt av klassen Bil
minBil.farge = "Rød";  // Setter objektets egenskap
minBil.kjør();  // Kaller metoden kjør()
```

Hver gang vi oppretter et objekt, får det sin egen kopi av variablene.

Mats Vik Pettersen
Programmering 2

Instansvariabel

En **instansvariabel** er en variabel som er definert i en klasse, men som tilhører et spesifikt objekt. Hvert objekt har sin egen kopi av instansvariablene.

```java
class Person {
    String navn;   // Instansvariabel
    int alder;     // Instansvariabel
}
```

Hvis vi lager flere Person-objekter, vil hver person ha sitt eget navn og alder.

```java
Person p1 = new Person();
p1.navn = "Ola";
p1.alder = 25;

Person p2 = new Person();
p2.navn = "Kari";
p2.alder = 30;
```

Her har p1 og p2 sine egne verdier for navn og alder.

Overloading (Metodeoverlasting)

Overloading betyr at flere metoder i samme klasse kan ha samme navn, men med forskjellig antall eller typer av parametere.

Eksempel på overloading:

```java
class Kalkulator {
    int add(int a, int b) {
        return a + b;
    }

    double add(double a, double b) {
        return a + b;
    }

    int add(int a, int b, int c) {
        return a + b + c;
    }
}
```

Her har vi tre add-metoder, men de har ulike parameterlister. Java velger riktig metode basert på argumentene.

Overriding (Metodeoverstyring)

Overriding betyr at en subklasse gir en ny implementasjon av en metode som er definert i en superklasse.

Eksempel på overriding:

```
class Hund extends Dyr {
    @Override
    void lagLyd() {
        System.out.println(x:"Hunden bjeffer.");
    }
}
```

Her **overstyrer** Hund-klassen metoden lagLyd fra Dyr.

Extends (Arv)

extends brukes for å lage en **subklasse** som arver egenskaper og metoder fra en **superklasse**.

Eksempel:

```
class Kjøretøy {
    int hjul;
    void kjør() {
        System.out.println(x:"Kjøretøyet beveger seg.");
    }
}

class Bil extends Kjøretøy {
    String modell;
}
```

Her arver Bil egenskaper og metoder fra Kjøretøy, så Bil har tilgang til hjul og kjør().

Mats Vik Pettersen
Programmering 2

Private, public, protected (Tilgangsmodifikatorer)

```
class Person {
    private String navn;  // Kun tilgjengelig innenfor klassen
    public int alder;     // Tilgjengelig overalt
    protected String adresse;  // Tilgjengelig i subklasser og samme pakke
}
```

private-variabelen navn kan ikke nås direkte utenfor klassen.

public-variabelen alder kan brukes overalt.

protected-variabelen adresse kan brukes i subklasser og samme pakke.

this og super

this

- Brukes til å referere til den **nåværende** instansen av klassen.
- Nyttig når parametere har samme navn som instansvariabler.

Eksempel:

```
class Bil {
    String farge;

    Bil(String farge) {
        this.farge = farge; // "this.farge" refererer til instansvariabelen
    }
}
```

Mats Vik Pettersen
Programmering 2

super

- Brukes til å referere til **superklassens** konstruktør eller metoder.
- Brukes når en subklasse trenger å kalle en metode eller konstruktør fra superklassen.

Eksempel:

```
class Dyr {
    Dyr() {
        System.out.println("Et dyr er opprettet.");
    }
}

class Hund extends Dyr {
    Hund() {
        super(); // Kaller superklassens konstruktør
        System.out.println("En hund er opprettet.");
    }
}
```

Når vi lager et Hund-objekt, vil først super() kalle Dyr sin konstruktør, deretter fortsetter Hund sin konstruktør.