Oblig 2 – Programmering 2

Thomas Waaler

Oppgave 1.1

Class

Class blir brukt til å definere klasser. En klasse inneholder et klassenavn og har en kodeblokk hvor man kan definere variabler, metoder og konstruktøren som blir kjørt når man oppretter et objekt ut ifra denne klassen. For eksempel:

```
class MyClass {
    public String name;

    public MyClass(String name) {
        this.name = name;
    }
}
```

Object

Et objekt er en instans av en klasse som har variabler og metoder som er definert i klassen. Objekter kan man bruke som egendefinerte datatyper. For eksempel:

```
MyClass objectInstance = new MyClass("Some name");
objectInstance.name = "Other name";
```

Instansvariabel

Instansvariabler er variabler definert i en klasse utenfor metoder, vanligvis på toppen av klassen. Dette er variabler som er relatert til den klassen og er tilgjengelig i hele klassen For eksempel en klasse med instansvariabler for lengde og bredde:

```
class Rectangle {
    public float width;
    public float height;
}
```

Overloading

Overloading er at man lager flere forskjellige definisjoner av en metode. Det vil si at man kan endre returverdien, antall parametere og rekkefølgen på datatypene på parameterne. Dette blir brukt dersom man ønsker å kalle én metode på forskjellige måter. For eksempel en metode som returnerer areal enten som int eller float:

```
public int getArea(int length, int height) {
    return length * height;
}
```

```
public float getArea(float length, float height) {
    return length * height;
}
```

Overriding

Overriding vil si at man overskriver/omdefinerer en metode fra foreldreklassen. For eksempel så er alle objekter et barne objekt av Object, så da kan man overskrive metoden toString sånn at den returnerer en bedre tekst enn det er som default. For å vise at man forsøker å overskrive en metode skriver man @Override før metoden:

```
@Override
public String toString() {
    return "New String here";
}
```

Extends

Extends brukes til å definere forelderklassen som klassen skal arve fra. Dette vil si at barneklassen har nå de samme variablene og metodene som er definert i foreldreklassen (Se bort ifra tilgangsmodifikatorer). For eksempel:

```
class OtherClass extends MyClass {
    public int length;

    public OtherClass(String name, int length) {
        super(name);
        this.length = length;
    }
}

OtherClass objRef = new OtherClass("Name", 23);
objRef.name = "Text";
objRef.length = 53;
```

Polymorphism

Polymorphism vil si at hvert objekt av en barneklasse er også et objekt av hver klasse den arver fra. Si vi har følgende klasser:

```
class Shape {...}
class Rectangle extends Shape {...}
class Circle extends Shape {...}
```

Klassene Rectangle og Circle kan man bruke som om de var et Shape objekt, men man kan ikke bruke et Shape objekt som om det var et Rectangle eller Circle objekt.

Private, public, protected

Private, public og protected er forskjellige tilgangsmodifikatorer man kan bruke på variabler og metoder i klasser for å definere hva slags tilgang vi har på de variablene og metodene utenfor klassen. Private vil si at man bare har tilgang til variablet eller metoden innenfor klassen, mens

Public vil si at man har tilgang utenfor klassen. Protected vil si at man har bare tilgang dersom man er en barneklasse eller er i samme package.

```
class Person {
    public String firstname;
    private String lastname;
    protected int age;

    public Person(String firstname, String lastname, int age) {
        this.firstname = firstname;    // Has access
        this.lastname = lastname;      // Has access
        this.age = age;                // Has access
    }
}
class Employee extends Person {
    public int id;

    public Employee(String firstname, String lastname, int age, int id) {
        //super(firstname, lastname, age); Normally uses this for initializing
        this.firstname = firstname;    // Has access
        this.lastname = lastname;      // Has NO access
        this.age = age;                // Has access
        this.id = id;                  // Has access
    }
}
```

this og super

This brukes til å referere til objektet man bruker det i. Dette blir vanligvis brukt dersom man har en metode som har parametere som heter det samme som instansvariablene, da kan man bruke this.variabelnavn for å bruke instansvariabel. Eksempel på dette kan man se på klasse definisjonene ovenfor.

Super brukes i konstruktøren på barneklasser for å kjøre konstruktøren til forelderklassen hvor man også spesifiserer foreldreklassen sine parametere. I eksempelet ovenfor istedenfor at Employee skal definere firstname, lastname og age slik som i Person så kan man bruke super() sånn at Person sin konstruktør tar seg av det.