Interface

→ Interface = en “fullstendig abstrakt klasse”. Brukes til å gruppere relaterte metoder uten implementasjon

Hvorfor?

- Skjule visse deltajer og bare vise de viktigste delene av et objekt (interface)
- I Java kan man ikke arve fra flere klasser samtidig (en klasse kan bare arve fra en superklasse) → en klasse kan **implementere** flere interfaces samtidig

`implements`: innholdet til et interface blir implementert (nesten samme måte som extends) av en annen klasse

```
interface Grensesnitt {
  void tullemetode(); //ikke noe implementasjon, kun definere metode-signaturen.
}
class Klasse implements Grensesnitt {

  @Override
  void tullemetode(){
    System.out.println("Dette er tull!");
  }
}
```

Vi kan også bruke interface som referanse type:

```
Grensesnitt tullevariabel = new Klasse();
```

Alle som implementerer samme grensesnitt får like egenskaper, f.eks. lagre i sammen beholder → array med ovbjekter som har “Grensesnitt” implementert:

```
Grensesnitt[] beholder = new Grenesnitt[10];
```

```
// Interface
interface Animal {
  public void animalSound(); // interface method (does not have a body)
  public void sleep(); // interface method (does not have a body)
}

// Pig "implements" the Animal interface
class Pig implements Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
  public void sleep() {
    // The body of sleep() is provided here
    System.out.println("Zzz");
  }
}

class Main {
  public static void main(String[] args) {
    Pig myPig = new Pig();  // Create a Pig object
    myPig.animalSound();
    myPig.sleep();
  }
}
```

Merk:

- Interfaces kan ikke brukes til å opprette objekter
- Interface kan ikke ha en konstruktør
- Metodene er “tomme”, innholdet skrives i klassen som implementerer interface
- En klasse `@Override` alle metoder fra interface
- Interface metoder: default `abstract` og `public`
- Interface attributter: default `public`, `static` og `final`

Implementere flere interfaces

```
interface FirstInterface {
  public void myMethod(); // interface method
}

interface SecondInterface {
  public void myOtherMethod(); // interface method
}

class DemoClass implements FirstInterface, SecondInterface {
  public void myMethod() {
    System.out.println("Some text..");
  }
  public void myOtherMethod() {
    System.out.println("Some other text...");
  }
}

class Main {
  public static void main(String[] args) {
    DemoClass myObj = new DemoClass();
    myObj.myMethod();
    myObj.myOtherMethod();
  }
}
```

Egne exceptions

- Unntak: try{} catch (Exception e) {}
- Lage egne unntak: https://trix.ifi.uio.no/assignments/557