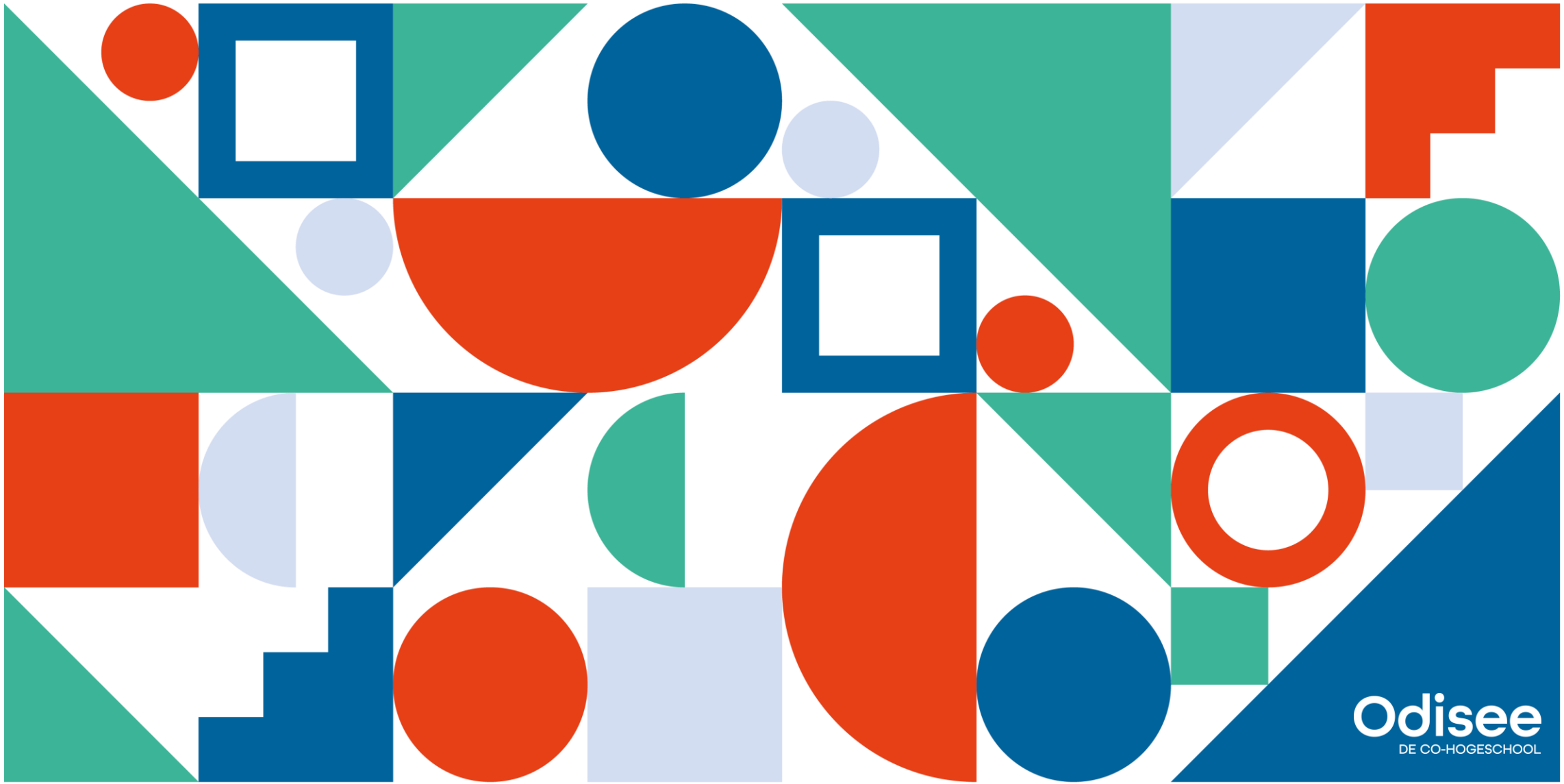

Odisee
DE CO-HOGESCHOOL

Interfaces & Exceptions

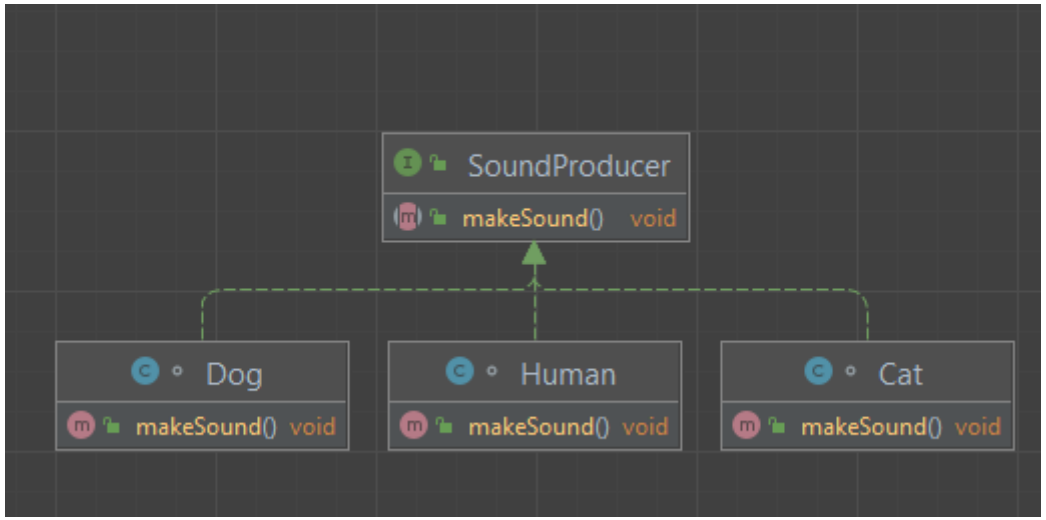
Interfaces

- Een interface is een koppelvlak waarmee twee systemen met elkaar communiceren
  - GUI: Grafische User Interface
    - Communicatie tussen user en interface
  - Binnen een programmeertaal
    - Communicatie tussen twee klassen/objecten
    - De interface van een klasse omvat alle publieke methoden van de klasse

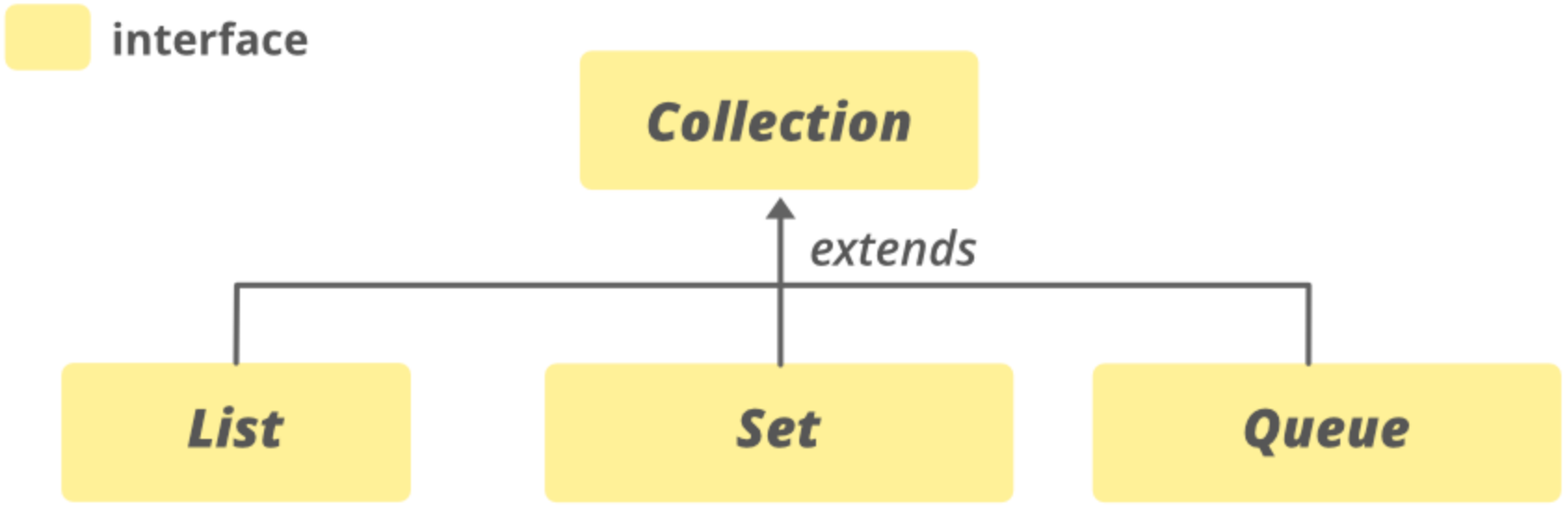
Interfaces

- Variant op overerving
  - Een abstract klasse met enkel abstracte methoden en geen attributen

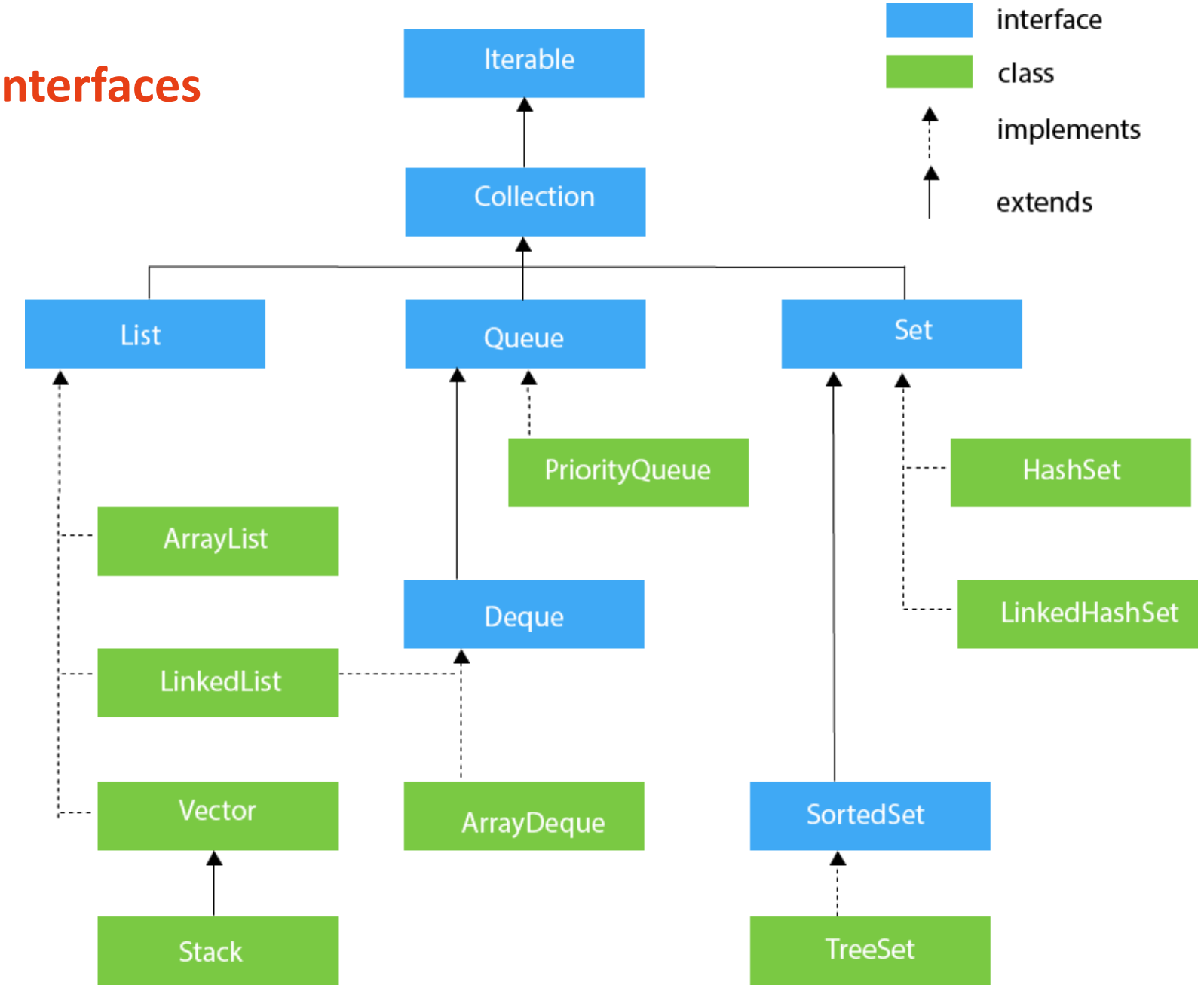
- Een interface geeft een API aan van hoe een klasse kan gebruikt worden
- Een klasse kan 1 of meerdere interfaces implementeren
  - In tegenstelling tot single inheritance


SoundProducer
makeSound() void
Dog
makeSound() void
Human
makeSound() void
Cat
makeSound() void

Voorbeeld interfaces

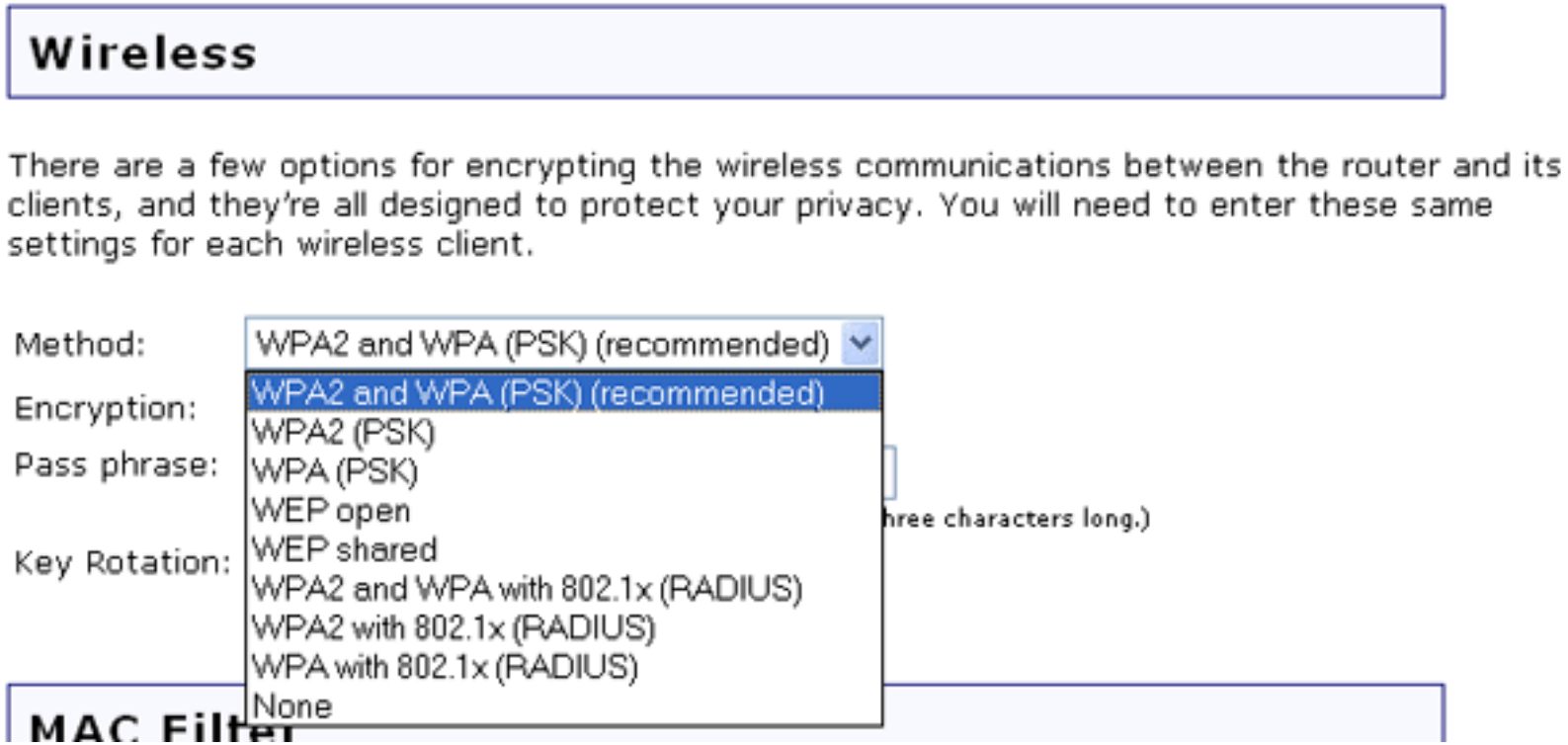


Voorbeeld interfaces



Voorbeeld interfaces

- User mag zelf encryptiealgoritme instellen



Voorbeeld interfaces

- Encryptiealgoritmes verschillen sterk maar kunnen allen encrypten en decrypten.

```
public interface EncryptionAlgorithm {

    String encrypt(String text);
    String decrypt(String text);
}
```

```
public class Database {

    private ArrayList<String> encryptedPasswords = new ArrayList<>();
    private EncryptionAlgorithm algoritme = null;

    public Database(String gekozenAlgoritme) {
        if (gekozenAlgoritme.equals("WPA3")) { algoritme = new WPA3Implementation(); }
        else if (gekozenAlgoritme.equals("WPA2")) { algoritme = new WPA2Implementation(); }
        else { algoritme = new WEPImplementation();} //default
    }

    public void storePassword(String password){
        String encrypted = algoritme.encrypt(password);
        encryptedPasswords.add(encrypted);
    }
}
```

Oefening – saving

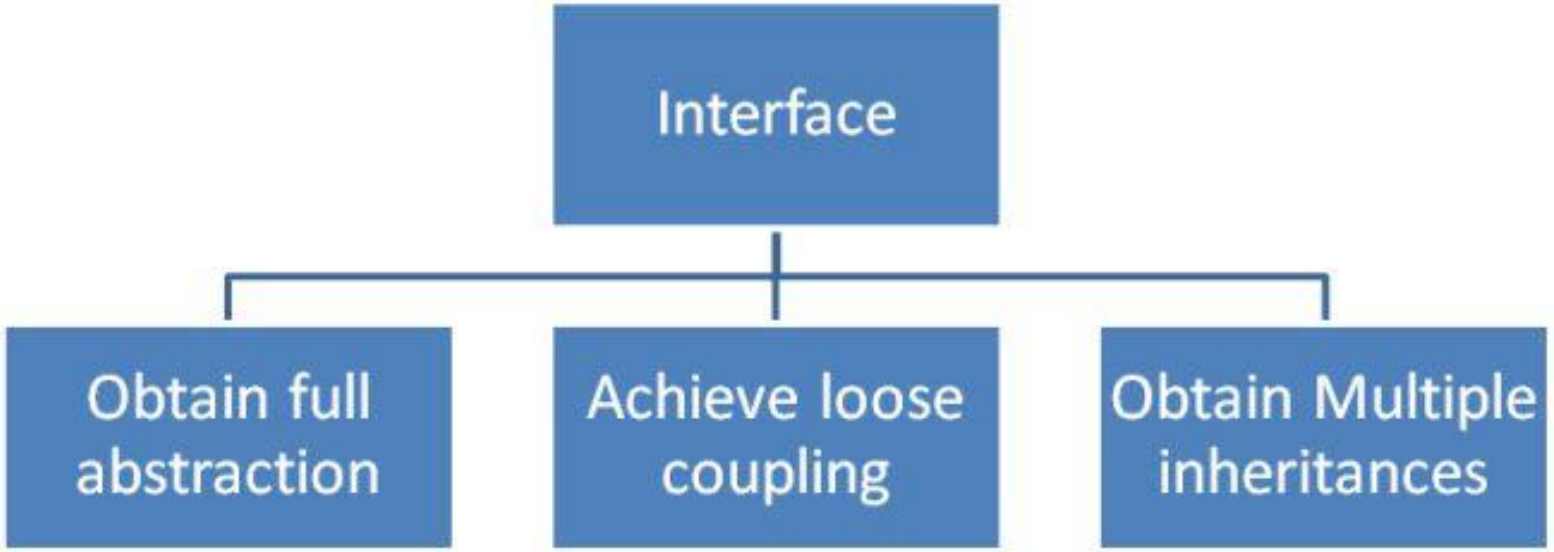
- Maak een klasse Factuur aan met een datum en een nummer
- We willen de Factuur kunnen opslaan op 2 manieren: textFile en DB.
- Maak een codeskelet welke dit mogelijk maakt met deze code.

```
public class Factuur {
        private int nummer;
        private LocalDate factuurDatum =LocalDate.now();

        public void save(final SomeSaveMethod saveObject)
{
            saveObject.saveInt(nummer);
            saveObject.saveLocalDate(factuurDatum);
        }
}
```

Oefening – saving

- Een textfile saven heeft deze functionaliteit
  - Je kan vragen of de file bestaat
  - Controleer of de file wel of niet readOnly is
  - Je kan de naam van het path opgeven
- In een database saven vereist deze functionaliteit
  - Met behulp van een connectiestring kan je een verbinding opzetten
  - Een controle of de laatste operatie wel geslaagd was
  - Instellen van de db user en password
- Wat is de interface? Welke class implementeert het?

Waarom interfaces in Java?



Exceptions

Exceptions

- Wat zijn exceptions?
- Is een exception een error?
- Hoe kan je een exception verwerken?

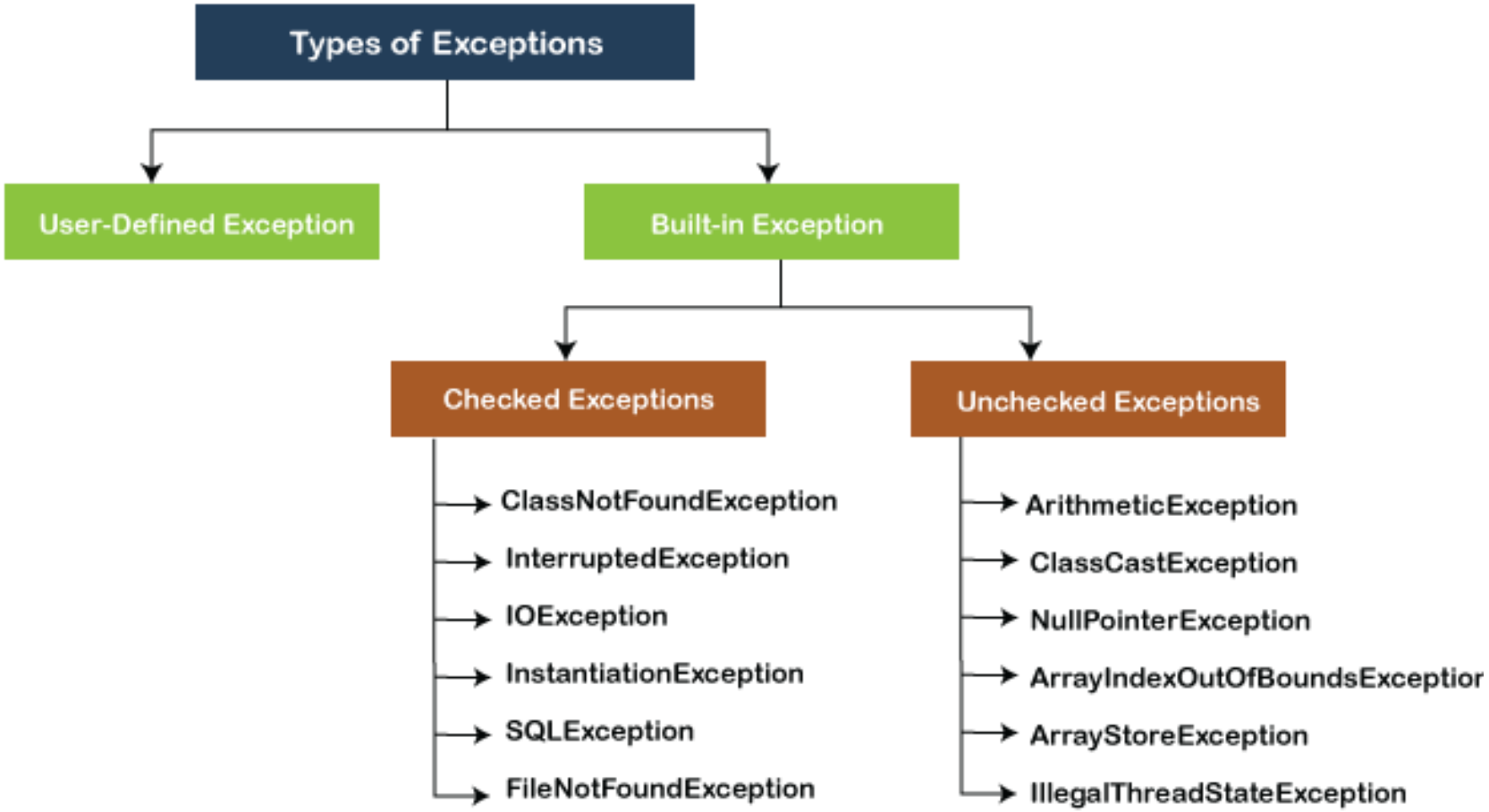
Exceptions

- Ongewenste of onverwacht event waar het object niet mee om kan
- Verstoord de normale flow van de applicatie
- Kan opgelost worden door de exception op te vangen en te verwerken
  - Try – catch
- Een exception is een object
  - Naam de exception
  - Beschrijving van het probleem
  - Status van het programma op het moment dat het misging

Errors

- Onherstelbare fouten
  - Out of memory, memory leaks, stack overflows, infinite recursion, …
- Vaak buiten de controle van de programmeur
  - Best geen try-catch gebruik om het op te lossen

Types of exceptions



Built-in Exceptions

- Aanwezig in de standaard java libraries
- Checked exceptions
  - Compile-time exceptions
- Unchecked exceptions
  - Runtime exceptions

User-Defined Exceptions

- Voor exceptions specifiek voor je applicatie
- Voordelen:
  - Vermijden van voortbouwen van errors
  - Duidelijkere splitsing van programma code en error-handling code
  - Duidelijkere rapportering van fouten
  - Verscheidene types fouten gespecifieerd worden
  - de applicatie crasht niet

Hoe worden exceptions verwerkt?

- Een functie werpt (throws) een exception
  - Functie stopt onmiddellijk
  - Fout wordt doorgegeven naar hoger-gelegen functies
  - Indien ze opgevangen wordt in een try-catch structuur gaat de applicatie verder, de eerste catch die tegengekomen wordt vangt de exception op
  - Indien ze niet opgevangen wordt crasht de applicatie

Exceptions – Oefening – Wat wordt er uitgeprint?

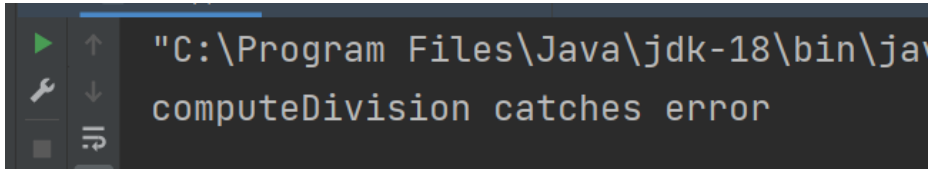
```java
public class Calculator {

    private int divide(int a, int b){
        return a/b;
    }

    public int computeDivision(int a, int b){
        try{
            return divide(a, b);
        } catch (ArithmeticException ex){
            System.out.println("computeDivision catches error");
            return 0;
        }
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Calculator calculator = new Calculator();

        try {
            calculator.computeDivision( a: 5,  b: 0);
        } catch (ArithmeticException ex){
            System.out.println("Main functie catches the error"
        }

    }
}
```

Exceptions – Oefening – Wat wordt er uitgeprint?

```java
public class Calculator {

    private int divide(int a, int b){
        return a/b;
    }

    public int computeDivision(int a, int b){
        try{
            return divide(a, b);
        } catch (ArithmeticException ex){
            System.out.println("computeDivision catches error");
            return 0;
        }
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Calculator calculator = new Calculator();

        try {
            calculator.computeDivision( a: 5,  b: 0);
        } catch (ArithmeticException ex){
            System.out.println("Main functie catches the error"
        }

    }
}
```



User-Defined Exceptions - Voorbeeld

```java
public class Person {

    private String name;
    private int age;

    public Person(String name, int age) throws InvalidNameException, InvalidAgeException {

        if(name == null || name.equals("")){
            throw new InvalidNameException();
        } else if(age < 0){
            throw new InvalidAgeException();
        }

        this.name = name;
        this.age = age;
    }

}
```

```java
public class InvalidNameException extends Exception{
}

 public class InvalidAgeException extends Exception{
 }
```

User-Defined Exceptions - Voorbeeld

```java
public class Person {

    private String name;
    private int age;

    public Person(String name, int age) throws InvalidNameException, InvalidAgeException {

        if(name == null || name.equals("")){
            throw new InvalidNameException();
        } else if(age < 0){
            throw new InvalidAgeException();
        }

        this.name = name;
        this.age = age;
    }

}
```

User-Defined Exceptions - Voorbeeld

```java
public class Main {

    public static void main(String[] args) {

        try {
            //Person person = new Person("", 18);
            //Person person = new Person("jens", -18);
            Person person = new Person( name: "",  age: -18);
        } catch (InvalidAgeException e) {
            System.out.println("Catch only invalid age exceptions");
        } catch (Exception e) {
            System.out.println("Catch all exceptions, including invalid name");
        } finally {
            System.out.println("This is optional");
            System.out.println("Something that will happen after try-catch block
        }
    }
}
```