



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

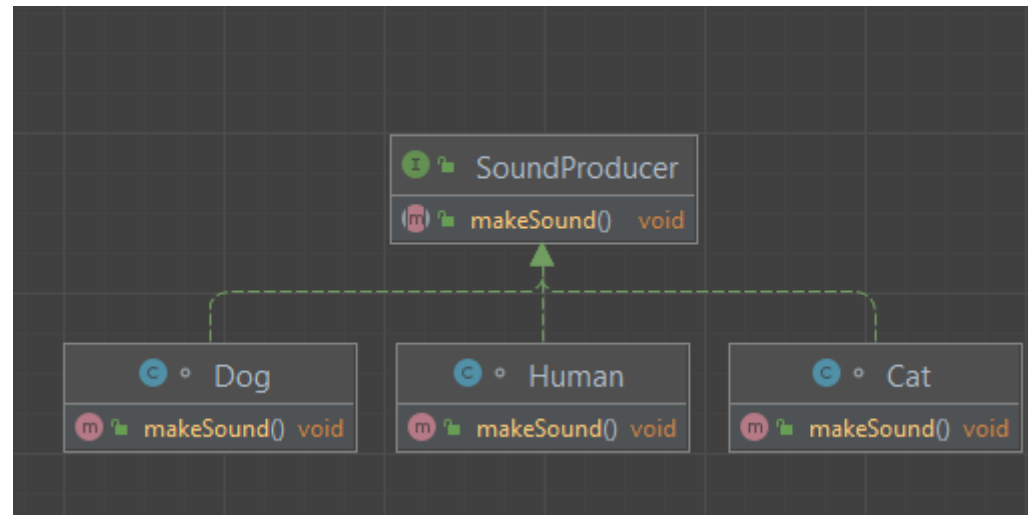
- Communicatie tussen twee klassen/objecten

- De interface van een klasse omvat alle publieke methoden van de klasse

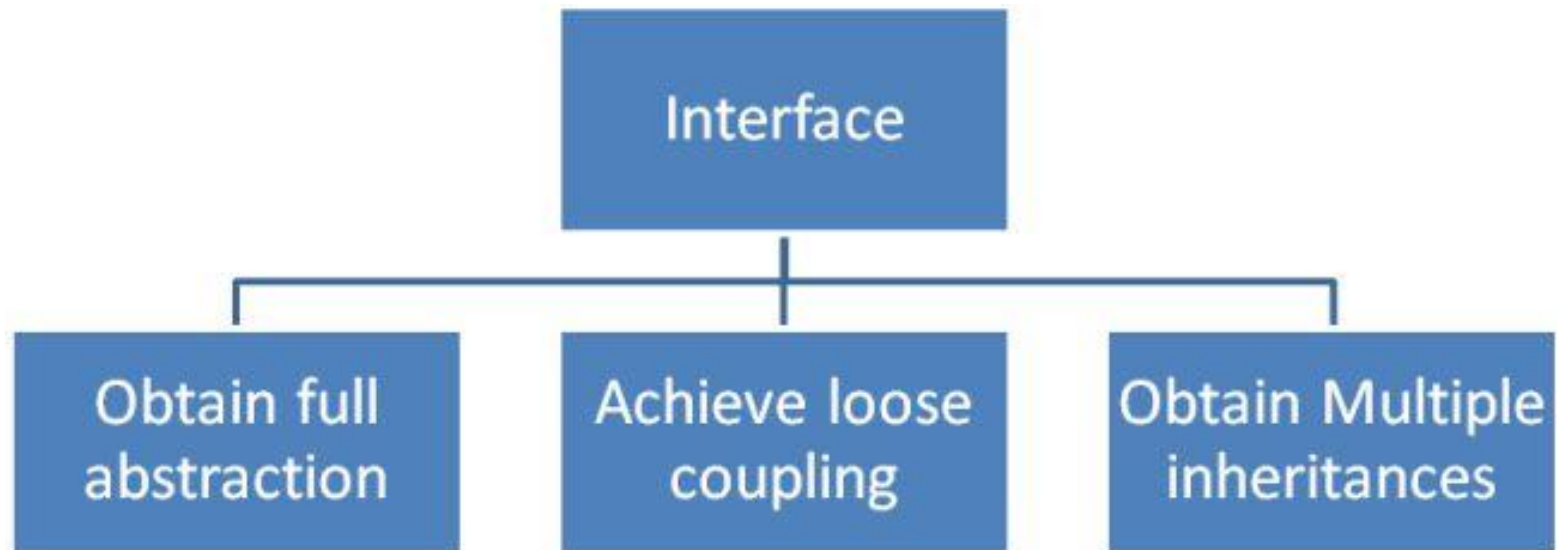


Interfaces

- ▣ 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



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
- ▣ Verstoord de normale flow van de applicatie
- ▣ Kan opgelost worden door de exception of 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

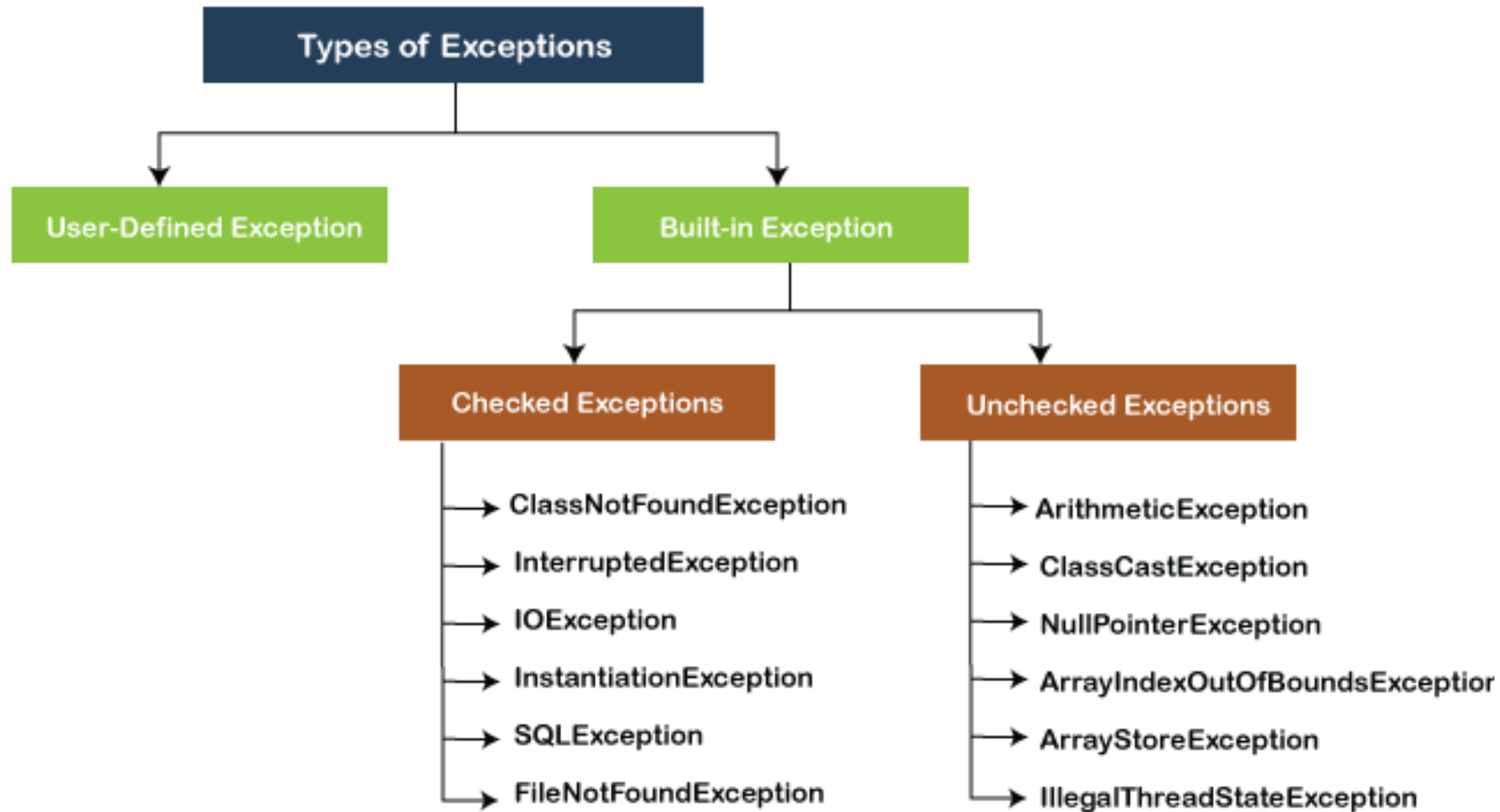
- ▣ Onherstelbare fouten

- 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 gespecificeerd worden
 - Vermeiden dat de applicatie crashed

Hoe worden exceptions verwerkt?

- Een functie werpt (throwed) 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?

```
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;  
        }  
    }  
}
```

```
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?

```
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;  
        }  
    }
```

```
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");  
        }  
    }
```

```
"C:\Program Files\Java\jdk-18\bin\jav  
computeDivision catches error
```

User-Defined Exceptions - Voorbeeld

```
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;  
    }  
}
```

```
public class InvalidNameException extends Exception{  
}  
  
public class InvalidAgeException extends Exception{  
}
```

User-Defined Exceptions - Voorbeeld

```
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

```
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");  
        }  
    }  
}
```