



# Java Essentials

## Hoofdstuk 14

### Exception Handling

**DE HOGESCHOOL  
MET HET NETWERK**

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt  
[www.pxl.be](http://www.pxl.be) - [www.pxl.be/facebook](http://www.pxl.be/facebook)



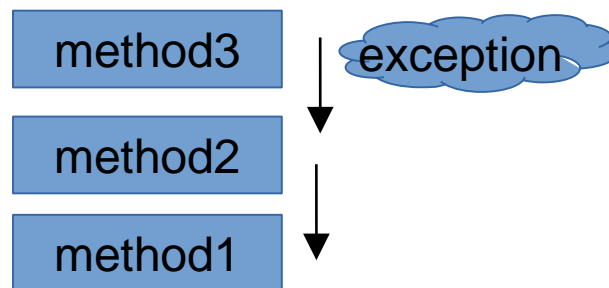
# Inhoud

1. Inleiding
2. Exceptions afhandelen
3. Exceptions genereren
4. Soorten exceptions
5. Zelf een exception-class maken



# 1. Inleiding

- exception = gebeurtenis die optreedt en de normale executie van het programma verstoort
- throwing an exception = creatie van een exception-object
- exception-handler behandelt de exception, indien geen exception-handler: uitvoer staakt
- call stack wordt doorlopen op zoek naar de juiste exception-handler (in omgekeerde volgorde)  
b.v. method1 calls method2 calls method3 (exception)



# 2. Exceptions afhandelen

## 2.1 Een exception veroorzaken

```
public static void main(String args[ ]) {  
    Scanner keyboard = new Scanner(System.in);  
    int numerator = Integer.parseInt(keyboard.next());  
    int denominator = Integer.parseInt(keyboard.next());  
    int division = numerator / denominator;  
    System.out.println(numerator + "/" + denominator +  
        "= " + division);  
    keyboard.close();  
}
```

Wat met invoer "a"?



## 2.1 Een exception veroorzaken

### → java.lang.NumberFormatException

<terminated> Opdracht1 (3) [Java Application] C:\Program Files (x86)\Java\jre1.8.0\_111\bin\javaw.exe (9 nov. 2016 11:10:44)

```
a
Exception in thread "main" java.lang.NumberFormatException: For input string: "a"
    at java.lang.NumberFormatException.forInputString(Unknown Source)
    at java.lang.Integer.parseInt(Unknown Source)
    at java.lang.Integer.parseInt(Unknown Source)
    at Opdracht1.main(Opdracht1.java:6)
```

stack trace



## 2.1 Een exception veroorzaken

### Opdracht 1:

- Test deze code uit met gehele, reële getallen en letters
- Zoek de methode `Integer.parseInt` op in de javadoc: welke exception kan deze methode genereren?

## parseInt

```
public static int parseInt(String s)
    throws NumberFormatException
```

Parses the string argument as a signed decimal integer. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' ('\u002D') to indicate a negative value or an ASCII plus sign '+' ('\u002B') to indicate a positive value. The resulting integer value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseInt(java.lang.String, int)` method.

### Parameters:

`s` - a `String` containing the `int` representation to be parsed

### Returns:

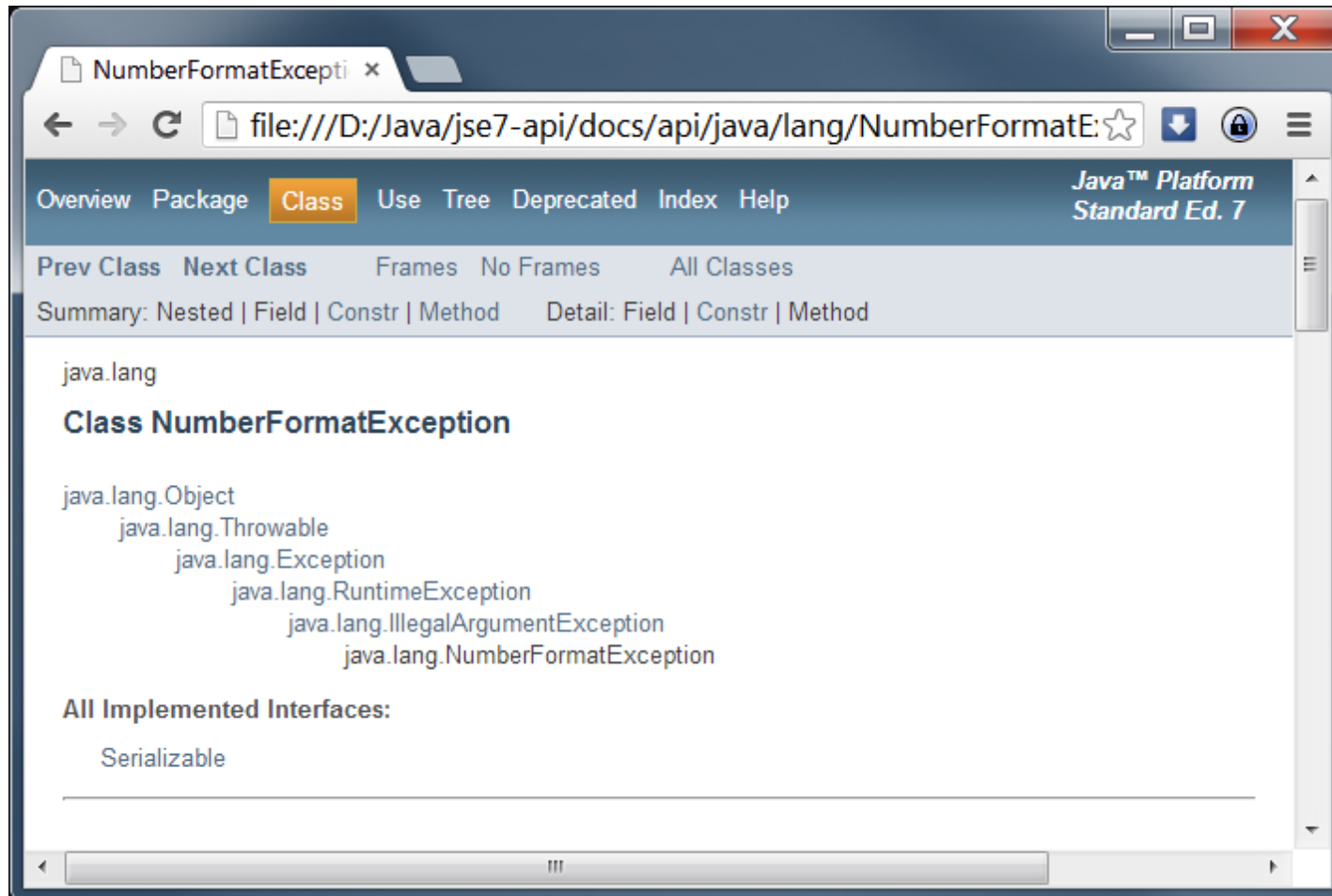
the integer value represented by the argument in decimal.

### Throws:

`NumberFormatException` - if the string does not contain a parsable integer.

## 2.2 Een exception opvangen

methode `Integer.parseInt()` kan een `NumberFormatException` exception-object opgooien





## 2.2 Een exception opvangen

### syntax

```
try {  
    // Code die "mogelijk" een exception genereert  
}  
catch (Throwable exceptionObject) {  
    // Code om de exception af te handelen  
    // Lijkt op een methode met 1 parameter  
    // Exception-object wordt doorgegeven aan catch-blok  
    // Exception-object moet afgeleid zijn van de class  
    // "Throwable"  
}
```



## 2.2 Een exception opvangen

```
public class Division {  
    public static void main(String args[]) {  
        try {  
            Scanner keyboard = new Scanner(System.in);  
            int numerator = Integer.parseInt(keyboard.next());  
            int denominator = Integer.parseInt(keyboard.next());  
            int division = numerator / denominator;  
            System.out.println(numerator + "/" + denominator +  
                               "=" + division);  
        }  
        ...  
    }  
}
```

→ code waarin een exception-object opgeworpen kan worden



## 2.2 Een exception opvangen

...

```
}  
    catch (NumberFormatException ex) {  
        System.out.println("Invalid number");  
    }  
    System.out.println("The End");  
}
```

- code waarin de exception afgehandeld wordt
- NumberFormatException is afgeleid van Throwable
- de referentie naar het exception-object krijgt de naam ex



## 2.2 Een exception opvangen

```
}  
catch (NumberFormatException ex) {  
    System.out.println("Invalid number");  
    System.out.println( ex.getMessage() );  
    ex.printStackTrace();  
}
```

- getMessage(): omschrijving van exception opvragen
- printStackTrace(): stack trace afdrukken



## 2.2 Een exception opvangen

### Opdracht 2:

- Zoek de klasse Throwable op in de javadoc
- Voeg een exception-handler toe, druk de foutboodschap en de stack trace op het scherm af
- Run het programma met deze exception-handling



## 2.3 Meerdere exceptions opvangen

- Er kunnen meerdere exceptions optreden in een codeblok
- Voor ieder soort uitzondering kan een afzonderlijk catch-blok voorzien worden
- Algemene syntax:

```
try {  
    // Code die “mogelijk” een exception genereert  
}  
catch (ThrowableClass1 exceptionObject) {  
    // afhandeling van exceptions van klasse ThrowableClass1  
}  
catch (ThrowableClass2 exceptionObject) {  
    // afhandeling van exceptions van klasse ThrowableClass2  
}  
catch (ThrowableClass3 exceptionObject) {  
    // afhandeling van exceptions van klasse ThrowableClass3  
}
```



## 2.3 Meerdere exceptions opvangen

Wat indien we proberen te delen door 0?

```
<terminated> Division (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_6
5
0
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at exercise15_02.Division.main(Division.java:12)
```



## 2.3 Meerdere exceptions opvangen

Een extra catch-blok specifiek voor deze uitzondering:

```
try {  
    Scanner keyboard = new Scanner(System.in);  
    int numerator = Integer.parseInt(keyboard.next());  
    int denominator = Integer.parseInt(keyboard.next());  
    int division = numerator / denominator;  
    System.out.println(numerator + "/" + denominator +  
                        "=" + division);  
}  
catch (NumberFormatException ex) {  
    System.out.println("Invalid number");  
}  
catch (ArithmeticException ex) {  
    System.out.println("Division by 0");  
}
```





## 2.3 Meerdere exceptions opvangen

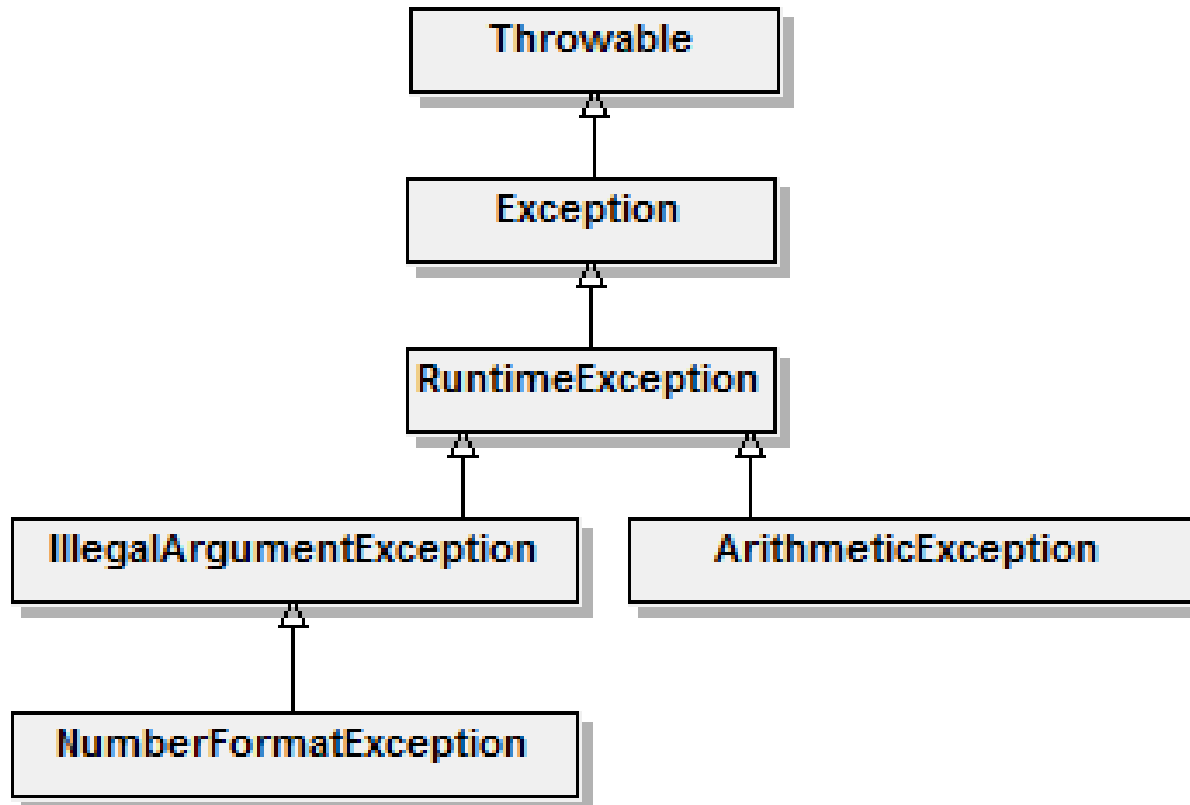
### Opdracht 3:

- Voer het programma uit met getallen 5 en 0: bekijk het resultaat
- Voeg de exception-handler voor `ArithmeticException` toe, voer het programma opnieuw uit



## 2.4 Gemeenschappelijke exception-handlers

Dankzij de klasse-hiërarchie kunnen exceptions gemeenschappelijk opgevangen worden:



## 2.4 Gemeenschappelijke exception-handlers

```
try {  
    Scanner keyboard = new Scanner(System.in);  
    int numerator = Integer.parseInt(keyboard.next());  
    int denominator = Integer.parseInt(keyboard.next());  
    int division = numerator / denominator;  
    System.out.println(numerator + "/" + denominator  
                        + "=" + division);  
}  
catch (RuntimeException ex) {  
    System.out.println("Invalid input !");  
}
```

Welke techniek komt hier aan bod door RuntimeException te gebruiken?

## 2.4 Gemeenschappelijke exception handlers

Exception handler die alle exceptions opvangt:

```
try {  
    ...  
}  
catch (Exception ex) {  
    ...  
}
```

Niet aan te raden want men weet niet precies welke  
→fout is opgetreden !

## 2.4 Gemeenschappelijke exception handlers

Sinds Java 7: exceptions combineren met | mogelijk

```
try {  
    ...  
}  
catch (NumberFormatException | IndexOutOfBoundsException ex){  
    System.out.println("Invalid input!");  
}
```



## 2.4 Gemeenschappelijke exception handlers

### Opdracht 4:

- Test het voorbeeld uit met de Runtime exception
- Test het voorbeeld uit met de gecombineerde exceptions via een |



## 2.4 Gemeenschappelijke exception handlers

Eerst catch-block subclass dan catch-block superclass

```
try {  
    ...  
}  
catch (ExceptionSubClass ex){  
    System.out.println(" invalid input!");  
}  
catch (ExceptionSuperClass ex){  
    System.out.println(" input error!");  
}
```

Opm: waarom eerst meest specifieke exception vermelden?



## 2.5 Het finally blok

Code die uitgevoerd moet worden, ongeacht het al dan niet optreden van een exception, plaatsen we in het finally-blok.

```
try {  
    ...  
}  
catch (Exception ex){  
    ...  
}  
finally {  
    ...  
}
```

Nut? Bestanden of netwerkconnecties afsluiten





## 2.5 Het finally blok

```
try {  
    Scanner keyboard = new Scanner(System.in);  
    int numerator = Integer.parseInt(keyboard.next());  
    int denominator = Integer.parseInt(keyboard.next());  
    int division = numerator / denominator;  
    System.out.println(numerator + "/" + denominator +  
                        "=" + division);  
}  
catch (RuntimeException exception) {  
    System.out.println("Error");  
}  
finally{  
    System.out.println("The End");  
}
```

Met en zonder exception wordt het finally-gedeelte uitgevoerd



## 2.5 Het finally blok

```
try {  
    Scanner keyboard = new Scanner(System.in);  
    int numerator = Integer.parseInt(keyboard.next());  
    int denominator = Integer.parseInt(keyboard.next());  
    int division = numerator / denominator;  
    System.out.println(numerator + "/" + denominator +  
                        "=" + division);  
}  
finally{  
    System.out.println("The End (either good or bad");  
}
```

- Finally zonder catch is ook toegelaten
- De eventuele exception zal gewoon verder gegooid worden



## 2.5 Het finally blok

### Opdracht 5:

- Voeg een finally-blok toe
- Test het programma uit met en zonder fouten



# 3. Exceptions genereren

- Zelf exception aanmaken en **opwerpen**  
**throw** new ExceptionClass();
- optie1: opwerpen en lokaal behandelen in try-catch block

```
try{  
    ...  
    if(...){  
        throw new ExceptionClass();  
    }  
    ...  
}  
catch(ExceptionClass e){  
    ...  
}
```



### 3. Exceptions genereren

- Zelf exception aanmaken en **opwerpen**  
**throw** new ExceptionClass();
- optie2: opwerpen in methode, behandelen daar waar methode aangeroepen wordt  
**throws**: aanduiding deze methode kan ExceptionClass opwerpen

```
public void method throws ExceptionClass{  
    ...  
    if(...){  
        throw new ExceptionClass();  
    }  
    ...  
}
```



# 3. Exceptions genereren

- Meerdere `throws` mogelijk bij methode

```
public void method throws ExcClass1,ExcClass2{  
    ...  
    if(...){  
        throw new ExcClass1();  
    }  
    if(...){  
        throw new ExcClass2();  
    }  
    ...  
}
```



# Voorbeeld

```
class Rectangle {  
    private int height;  
    private int width;  
  
    public Rectangle(int width, int height) throws Exception {  
        setHeight(height);  
        setWidth(width);  
    }  
  
    public void setHeight(int height) throws Exception {  
        if (height < 0) {  
            throw new Exception("negative height");  
        }  
        this.height = height;  
    }  
  
    public void setWidth(int width) throws Exception {  
        if (width < 0) {  
            throw new Exception("negative width");  
        }  
        this.width = width;  
    }  
}
```

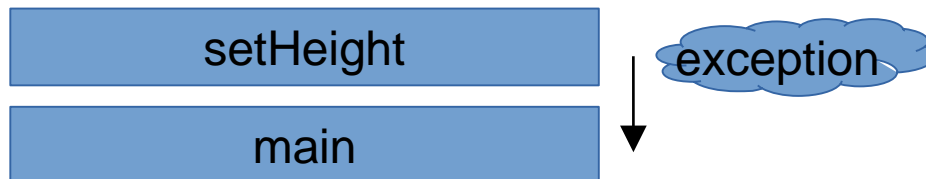


```
public static void main(String[] args){  
    try{  
        Rectangle rectangle = new Rectangle(-1,10);  
    }  
    catch(Exception e){  
        System.out.println(e.getMessage());  
    }  
}
```

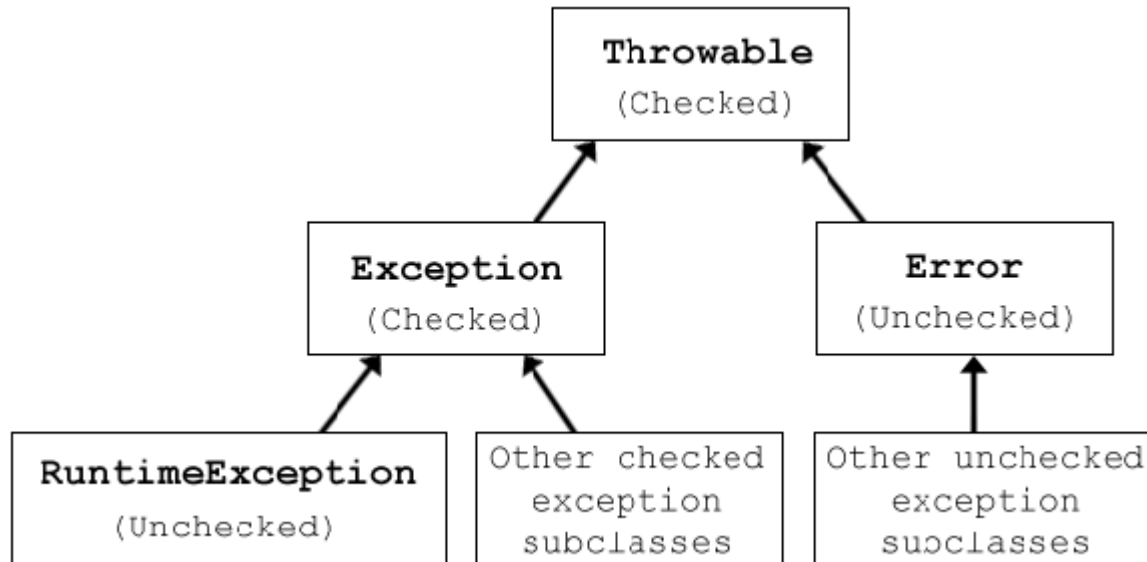




```
public static void main(String[] args){  
    try{  
        Rectangle rectangle = new Rectangle(10,10);  
        rectangle.setHeight(-1);  
    }  
    catch(Exception e){  
        System.out.println(e.getMessage());  
    }  
}
```



## 4. Soorten exceptions



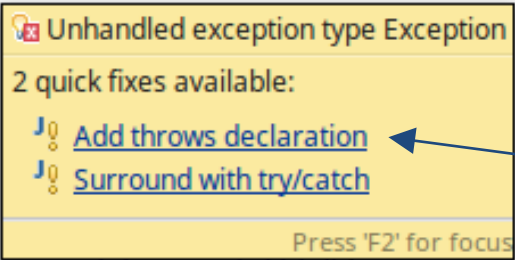
## 4. Soorten exceptions

- exception kan opgevangen worden in try... catch error niet
- checked (bv. Exception) vs unchecked (bv. RuntimeException)
  - (1) throw checked exception in methode:  
compiler controleert dat
    - ofwel try... catch in methode zelf
    - ofwel throws vermeld bij de definitie vd methode
  - (2) methode met throws checked exception  
compiler controleert dat
    - ofwel try... catch waar methode aangeroepen wordt
    - ofwel throws vermeld wordt in de methode die deze methode aanroept



# Exception is checked

```
public void setHeight(int height) {  
    if (height < 0) {  
        throw new Exception("negative height");  
    }  
    this.height = height;  
}  
  
public void setWidth(int width) {  
    if (width < 0) {  
        throw new Exception("negative width");  
    }  
}
```



- (1) throw Exception (checked) in methode:  
compiler controleert dat
- ofwel try catch in methode zelf
  - ofwel throws vermeld bij de definitie vd methode

```

1 class Rectangle {
2     private int height, width;
3
4     public Rectangle(int width, int height) | {
5         setHeight(height);
6
7     }
8
9     public void setHeight(int height) throws Exception {
10         if (height < 0) {
11             throw new Exception("negative height");
12         }
13         this.height = height;
14     }
15
16     public void setWidth(int width) throws Exception {
17         if (width < 0) {
18             throw new Exception("negative width");
19         }
20         this.width = width;
21     }
22 }
23

```

Unhandled exception type Exception

2 quick fixes available:

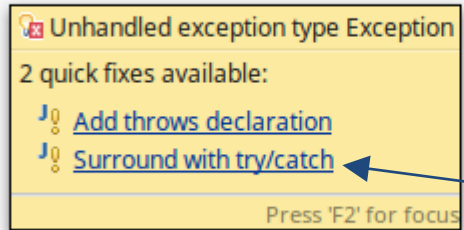
- Add throws declaration
- Surround with try/catch

Press 'F2' for focus

(2) methode (setHeight) met throws Exception (checked)  
compiler controleert dat

- ofwel try... catch waar methode aangeroepen wordt
- ofwel throws vermeld wordt in de methode die deze methode aanroept

```
public static void main(String[] args) {  
    new Rectangle(10, 10);  
}
```



- (2) methode (constructor) met throws Exception (checked)  
compiler controleert dat
- ofwel try... catch waar methode aangeroepen wordt
  - ofwel throws vermeld wordt in de methode die deze methode aanroept

## 4. Soorten exceptions

### Opdracht 6:

Vervang in de code van Rectangle Exception door RuntimeException.

Kijk of dezelfde verplichtingen als getoond op de voorgaande 3 slides gelden voor deze nieuwe situatie.



# 5. Zelf een exception-klasse maken

```
class NegSizeException extends RuntimeException{  
    public NegSizeException(){  
        super();  
    }  
  
    public NegSizeException(String message){  
        super(message);  
    }  
  
    public NegSizeException(String message, Throwable cause){  
        super(message, cause);  
    }  
  
    public NegSizeException(Throwable cause){  
        super(cause);  
    }  
}
```





## 5. Zelf een exception-klasse maken

### Waarom RuntimeException?

Runtime exceptions zijn exceptions die normaal door de programmeur die de code gebruikt behandeld moeten worden. De compiler verplicht de controle dat de grootte niet negatief mag zijn niet.

### Waarom cause in constructor NegSizeException?

```
public NegSizeException(String message, Throwable cause){  
    super(message, cause);  
}
```

Exception opvangen en opnieuw opwerpen. Info over de originele exception (cause) wordt meegegeven .



# Waarom cause in constructor NegSizeException?

```
public Rectangle(int width, int height) throws NegSizeException {  
    try {  
        setHeight(height);  
        setWidth(width);  
    } catch (NegSizeException ex) {  
        System.out.println(ex.getMessage());  
        throw new NegSizeException("negative");  
    }  
}
```

enkel message  
wordt meegegeven  
in de constructor

```
public static void main(String[] args) {  
    try{  
        Rectangle r=new Rectangle(-12, 2);  
    }  
    catch(Exception ex){  
        ex.printStackTrace();  
    }  
}
```

```
negative width  
NegSizeException: negative  
    at Rectangle.<init>(Rectangle.java:10)  
    at Rectangle.main(Rectangle.java:31)
```

bepaalde info



# Waarom cause in constructor NegSizeException?

```
public Rectangle(int width, int height) throws NegSizeException {  
    try {  
        setHeight(height);  
        setWidth(width);  
    } catch (NegSizeException ex) {  
        System.out.println(ex.getMessage());  
        throw new NegSizeException("negative", ex);  
    }  
}
```

message en  
oorspronkelijke  
exception worden  
meegegeven in de  
constructor

```
public static void main(String[] args) {  
    try{  
        Rectangle r=new Rectangle(-12, 2);  
    }  
    catch(Exception ex){  
        ex.printStackTrace();  
    }  
}
```

```
negative width  
NegSizeException: negative  
    at Rectangle.<init>(Rectangle.java:10)  
    at Rectangle.main(Rectangle.java:31)  
Caused by: NegSizeException: negative width  
    at Rectangle.setWidth(Rectangle.java:23)  
    at Rectangle.<init>(Rectangle.java:7)  
    ... 1 more
```

uitgebreide info

