HO
GENT

OOSDII

Exception handling - Oefeningen

Table of Contents

1. DivideByZeroWithExceptionHandling voorbeeld

Gegeven de classe `DivideByZeroNoExceptionHandling` uit de theorie.

In de functionaliteit is nog geen rekenening gehouden met het opvangen en afhandelen van exceptions, waardoor het programma soms bruusk tot een einde komt.

We passen de code aan om de fouten op te vangen en af te handelen.

1.1. Aanpassing programma delen door 0

Bij een foute invoer van de noemer (0 of geen integer getal) treedt een exception op. We gaan deze exceptions als volgt afhandelen:

- Uitschrijven van de fout (System.err)
- Duidelijke foutmelding voor de gebruiker uitschrijven (System.out)
- Nieuwe invoer vragen aan de gebruiker

We hebben twee mogelijkheden om de exceptions af te handelen:

- afhandeling op de plaats waar de fout optreedt (duidelijk en vlugger) → try-catch-finally-statement
- afhandeling op een andere plaats → impliciete of expliciete exception propagation

1.2. Afhandeling via try-catch-finally

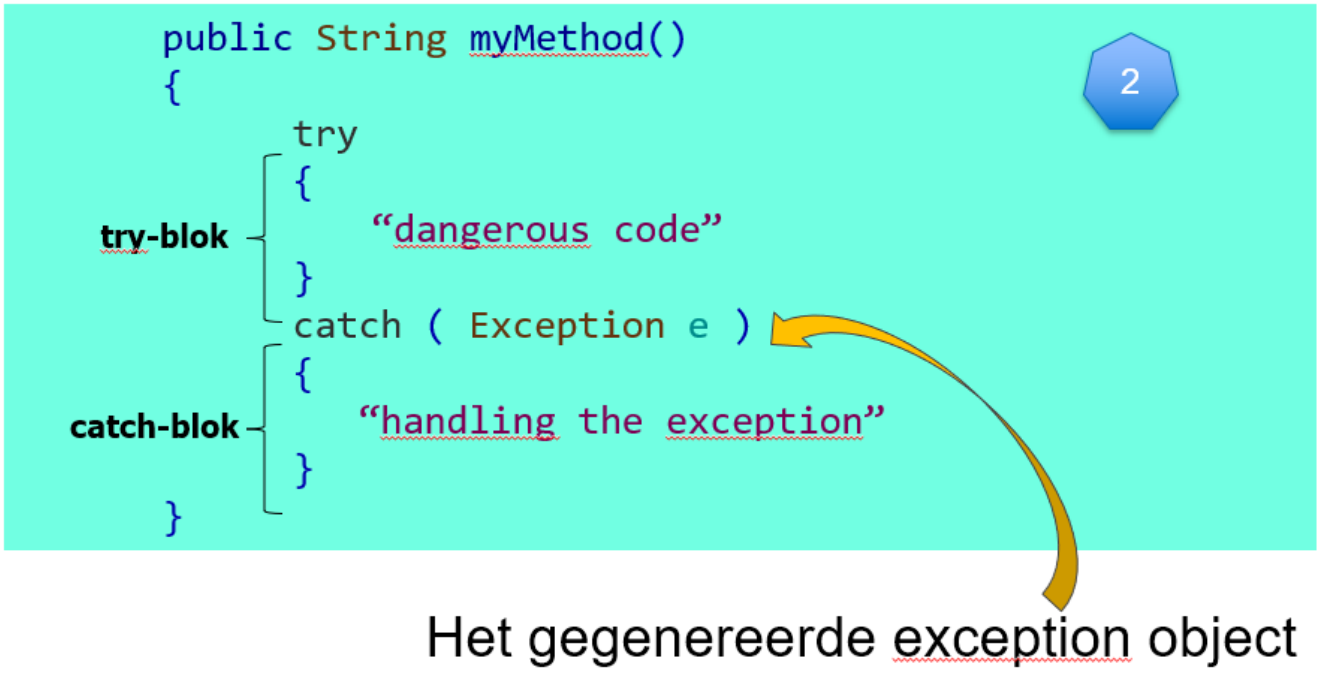
- Alle instructies waarin zich fouten kunnen voordoen en alle instructies die niet mogen uitgevoerd worden wanneer een exception optreedt komen in een try-blok
- als er een fout optreedt in het try-blok, dan wordt er een instantie van een exception-class gemaakt en deze kan opgevangen worden in één van de catch-blokken; elk catch blok verwerkt een bepaald soort fout; als er geen geschikte catcher is in het programma zal dit bruusk eindigen
- code die hoe dan ook moet uitgevoerd worden (bv. vrijgeven van resources): finally-blok [optioneel]

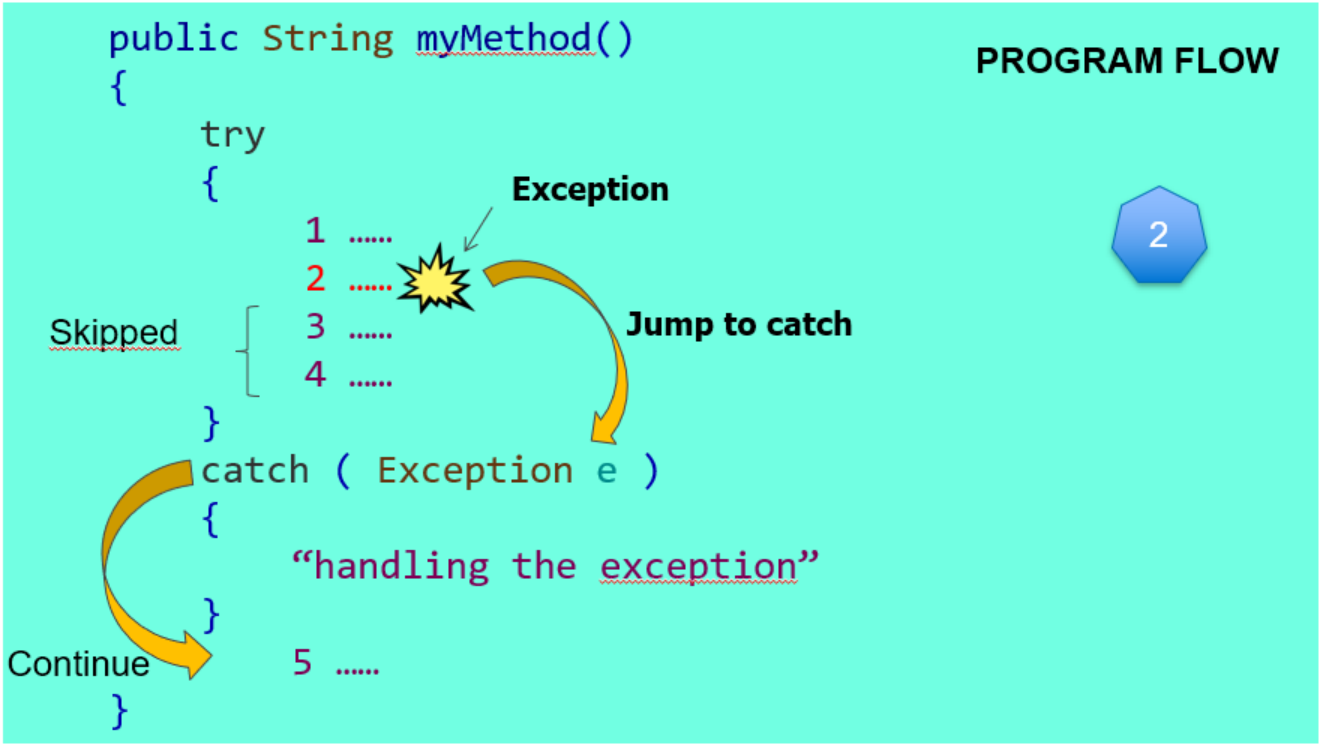
Het try-catch-finally statement:

```
try
{
    // code die mag falen
}
catch (Exception e)
{
    // afhandelen Exception
}
// meerdere catchers mogelijk
[finally
{
    // code die altijd uitgevoerd wordt
}]
```

Let op:

- Wanneer een exception optreedt in een try-blok, dan wordt dat blok verlaten (zelfs als het throw punt in een geneste blok zit) en wordt verdergegaan met het corresponderende catch-blok.
- In een catch-blok kan je geen gebruik maken van objecten lokaal gedeclareerd in het corresponderende try-blok.
- Tussen de catchers en het try-blok kan geen andere code staan.






```
6   public class DivideByZeroWithExceptionHandling {
7
8       public static int berekenQuotient(int teller, int noemer){
9                   return teller / noemer;
10      }
11
12      public static void main(String[] args) {
13          Scanner scanner = new Scanner(System.in);
14          boolean blijvenHerhalenFlag = true;
15          do {
16              try {
17                  System.out.print("Geef een integere waarde voor de teller: ");
18                  int teller = scanner.nextInt();
19                  System.out.print("Geef een integere waarde voor de noemer: ");
20                  int noemer = scanner.nextInt();
21
22                  int result = berekenQuotient(teller, noemer);
23                  System.out.printf("%nResultaat: %d / %d = %d%n", teller,
24                          noemer, result);
25                  blijvenHerhalenFlag = false;
26              } catch (InputMismatchException inputMismatchException) {
27                  System.err.printf("%nException: %s%n",
28                          inputMismatchException);
29                  scanner.nextLine();
30                  System.out.printf(
31                          "You must enter integers. Please try again.%n%n");
32              } catch (ArithmeticException arithmeticException) {
33                  System.err.printf("%nException: %s%n", arithmeticException);
34                  System.out.printf(
35                          "Zero is an invalid denominator. Please try again.%n%n");
36              }
37          } while (blijvenHerhalenFlag);
38      }
39  }
```

3

GEEL

Code kan falen

ORANJE

Niet uitvoeren als er zich een fout voordoet

1.3. Stap voor stap

Volgende methodes kunnen verkeerd lopen. Beide methodes propageren impliciet een Exception.

```
1     int teller = scanner.nextInt();
2     int noemer = scanner.nextInt();
3     int result = berekenQuotient(teller, noemer);
```

- **Propageren**: Dit wil zeggen dat ze de fout niet afhandelen, maar gewoon “verder gooien” naar de methode die hen opgeroepen heeft.
- **Impliciet**: Aan de signatuur van de methode zie je niet dat er een Exception kan optreden.

1.3.1. Impliciet propageren

nextInt

```
public int nextInt()
```

Scans the next token of the input as an `int`.

An invocation of this method of the form `nextInt()` behaves in exactly the same way as the invocation `nextInt(radix)`, where `radix` is the default radix of this scanner.

Returns:
the int scanned from the input

Throws:
`InputMismatchException` - if the next token does not match the *Integer* regular expression, or is out of range
`NoSuchElementException` - if input is exhausted
`IllegalStateException` - if this scanner is closed

1.3.2. Expliciet propageren

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.

We kunnen ook zelf een exception expliciet propageren, bv. bij berekenQuotient(...)

```
1     public static int berekenQuotient(int teller, int noemer) throws
  ArithmeticException
2     {
3         return teller / noemer;
4     }
```

In de signatuur van de methode leg je vast dat dit type Exception kan optreden in de body van de methode. Soms is het verplicht dit op te nemen.

1.4. Verschillende exceptions afhandelen

In het try-blok kunnen uiteindelijk 2 verschillende soorten Exceptions optreden:

- Een InputMismatchException gegooid door nextInt()

- Een ArithmeticException gegooid door berekenQuotient()

```
try {
   System.out.print("Geef een integere waarde voor de teller: ");
   int teller = scanner.nextInt();
   System.out.print("Geef een integere waarde voor de noemer: ");
   int noemer = scanner.nextInt();

   int result = berekenQuotient(teller, noemer);
   System.out.printf("%nResultaat: %d / %d = %d%n", teller,
         noemer, result);
   blijvenHerhalenFlag = false;
}
```

We voorzien voor elk type een verschillend catch-blok, de afhandeling is per type verschillend:

- Andere boodschap uitschrijven voor gebruiker.
- Scanner object klaar zetten voor volgende input (enkel nodig waar het inlezen zelf fout ging)

```
catch (InputMismatchException inputMismatchException) {
   System.err.printf("%nException: %s%n",
         inputMismatchException);
   scanner.nextLine();
   System.out.printf(
         "De invoer moeten integere getallen zijn. Probeer opnieuw.%n%n");
} catch (ArithmeticException arithmeticException) {
   System.err.printf("%nException: %s%n", arithmeticException);
   System.out.printf(
         "Het cijfer 0 kan geen noemer zijn. Probeer opnieuw.%n%n");
}
catch (Exception exception)// ALL-catcher
{
   System.err.printf("%nException: %s%n", exception);
}
```

We voorzien voor elk type een verschillend catch-blok en voor de volledigheid ook de ALL-catcher:

- Deze catcher vangt de fouten op die we eventueel niet voorzien hadden.

```
catch (Exception exception)// ALL-catcher
{
   System.err.printf("%nException: %s%n", exception);
}
```

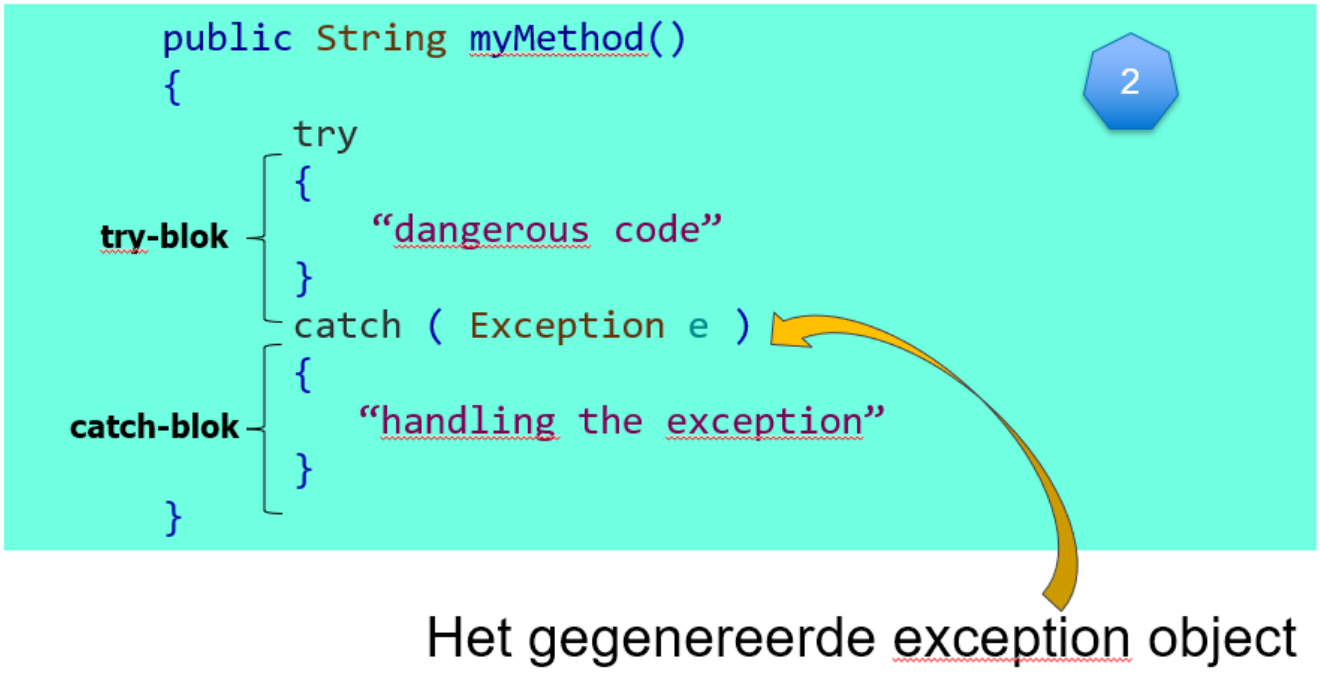
Denk goed na: een all-catcher is niet altijd een goede oplossing. Je moet heel goed weten of de afhandeling in alle gevallen dan wel ok is!

Elk catch-blok start met het keyword catch, gevolgd door een parameterlijst met één parameter, die aangeeft welk soort exceptie kan opgevangen worden meerdere parameters gescheiden door "|"

```
catch ( Type1 | Type2 | Type3 e) {….}
```

Het programma stopt als er geen geschikte catcher is.

- De eerste catcher na een try-blok die overeenkomt met het exception-type wordt uitgevoerd; alle andere worden genegeerd!
- Zet nooit een catcher van een superklasse object vóór een catcher van een subklasse object!
- Een exception kan op verschillende wijzen afgehandeld worden.
- Het is niet mogelijk om nog terug te keren naar het throw punt!



Merk op:

- Aan de hand van de boolean (vlag) blijvenHerhalenFlag zorgen we ervoor dat de do-while-lus enkel verlaten wordt als de try-blok geen enkel probleem ondervonden heeft. In dat geval wordt effectief het laatste statement van de try-blok uitgevoerd (`blijvenHerhalenFlag = false;`) en wordt bijgevolg de do-while-lus afgesloten. In alle andere (probleem)situaties blijven we in de do-while-lus!

- Er is geen finally-blok opgenomen. Er is geen code die sowieso uitgevoerd moet worden, zelfs al treedt er een exception op.

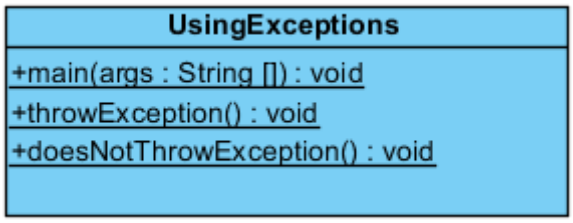
```
1  public class DivideByZeroWithExceptionHandling {
2
3      public static int berekenQuotient(int teller, int noemer){
4                      return teller / noemer;
5      }
6
7      public static void main(String[] args) {
8          Scanner scanner = new Scanner(System.in);
9          boolean blijvenHerhalenFlag = true;
10         do {
11             try {
12                 System.out.print("Geef een integere waarde voor de teller: ");
13                 int teller = scanner.nextInt();
14                 System.out.print("Geef een integere waarde voor de noemer: ");
15                 int noemer = scanner.nextInt();
16
17                 int result = berekenQuotient(teller, noemer);
18                 System.out.printf("%nResultaat: %d / %d = %d%n", teller,
19                         noemer, result);
20                 blijvenHerhalenFlag = false;
21             }
22             catch (InputMismatchException inputMismatchException) {
23                 System.err.printf("%nException: %s%n",
24                         inputMismatchException);
25                 scanner.nextLine();
26                 System.out.printf(
27                         "De invoer moeten integere getallen zijn. Probeer
   opnieuw.%n%n");
28             } catch (ArithmeticException arithmeticException) {
29                 System.err.printf("%nException: %s%n", arithmeticException);
30                 System.out.printf(
31                         "Het cijfer 0 kan geen noemer zijn. Probeer opnieuw.%n%n");
32             }
33             catch (Exception exception)// ALL-catcher
34             {
35                 System.err.printf("%nException: %s%n", exception);
36             }
37         } while (blijvenHerhalenFlag);
38     }
39 }
```

2. Even tussendoor

Wat wordt uitgeschreven op het scherm?

```
    public void myMethod()
    {
        try
        {
            int aNumber;
            aNumber = Integer.parseInt("this is not a number");
            System.out.println("Wrong input");
        } catch (NumberFormatException e)
        {
            System.out.println("The given number is not a number");
        }
        System.out.println("Continue");
    }
```

3. UsingExceptions voorbeeld

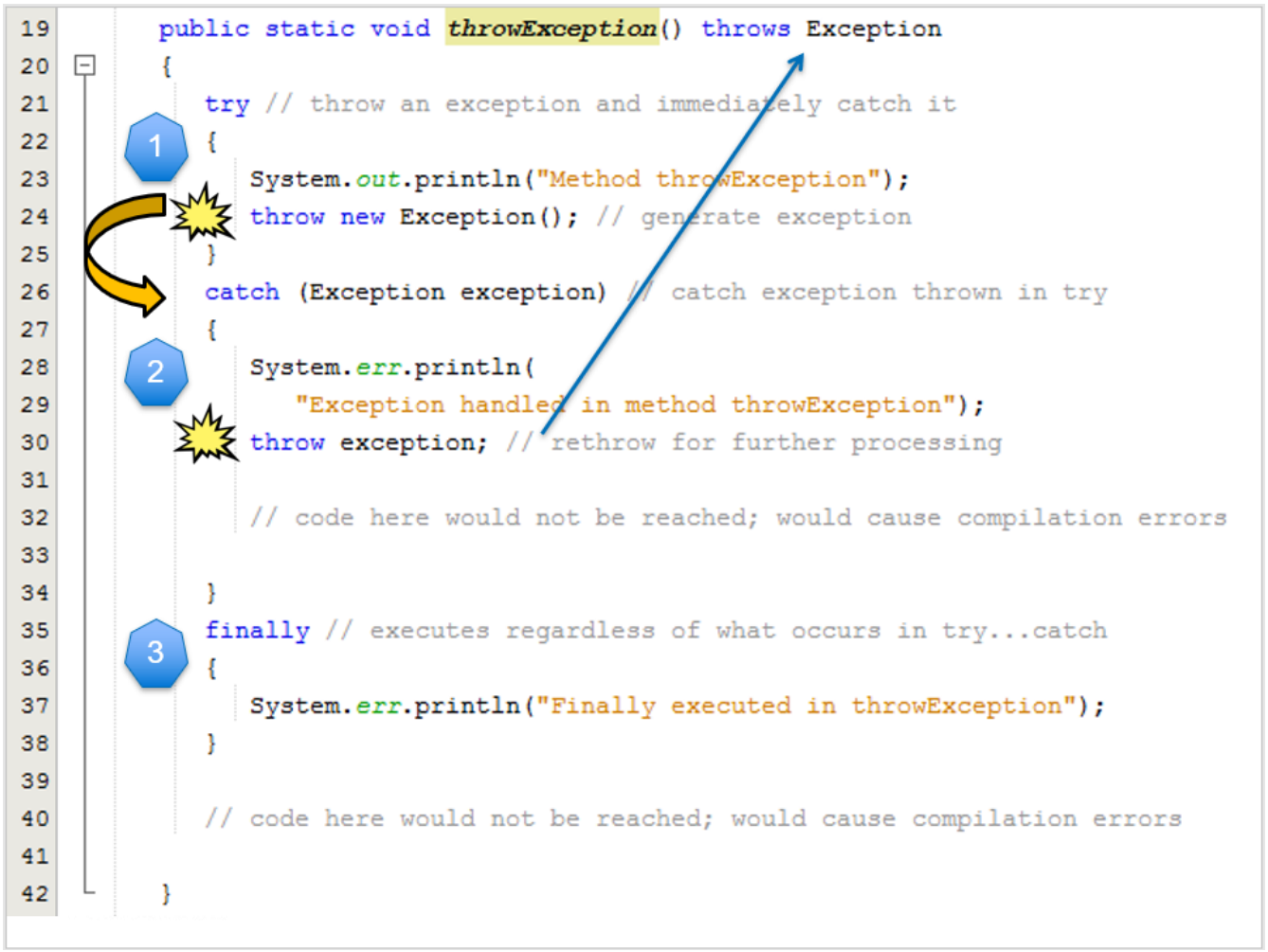


- Static main methode roept de twee andere static methodes aan;
- **throwException**: er kan een Exception optreden, die wordt deels afgehandeld en dan wordt de Exception verder doorgegooid
- **doesNotThrowException**: hier treden geen Exceptions op
- De laatste 2 methodes hebben elk een try-catch-finally- blok. Wat wordt precies uitgevoerd?

```
public static void main(String[] args)
{
    try
    {
        throwException();
    }
    catch (Exception exception) // exception thrown by throwException
    {
        System.err.println("Exception handled in main");
    }

    doesNotThrowException();
}
```

Oproepen van 2 static methodes uit dezelfde klasse.

- In de eerste methode kan een Exception optreden, dus deze aanroep moet in een try-blok staan.
- Bij de tweede methode kan er geen Exception optreden.




```
19  public static void throwException() throws Exception
20  {
21     try // throw an exception and immediately catch it
22     {
23        System.out.println("Method throwException");
24        throw new Exception(); // generate exception
25     }
26     catch (Exception exception) // catch exception thrown in try
27     {
28        System.err.println(
29           "Exception handled in method throwException");
30        throw exception; // rethrow for further processing
31
32        // code here would not be reached; would cause compilation errors
33
34     }
35     finally // executes regardless of what occurs in try...catch
36     {
37        System.err.println("Finally executed in throwException");
38     }
39
40     // code here would not be reached; would cause compilation errors
41
42  }
```

- Uitvoeren van try-blok
  - Er treedt een Exception op. Dit object wordt hier aangemaakt en gegooid via de instructie **throw**
- Catch-blok vangt de Exception op
  - Het afhandelen hier bestaat uit iets afdrukken in de err-stream. Verder afhandelen is niet mogelijk, datzelfde object wordt verder gegooid (via throw op lijn 30).
  - Exception is een checked Exception en wordt hier niet meer verder opgevangen moet dus vermeld worden in de signatuur van de methode op lijn 19 *Finally-blok wordt als laatste stap uitgevoerd.

```
44      public static void doesNotThrowException()
45      {
46         try // try block does not throw an exception
47         {
48            System.out.println("Method doesNotThrowException");
49         }
50         catch (Exception exception) // does not execute
51         {
52            System.err.println(exception);
53         }
54         finally // executes regardless of what occurs in try...catch
55         {
56            System.err.println(
57               "Finally executed in doesNotThrowException");
58         }
59
60         System.out.println("End of method doesNotThrowException");
61      }
```

- Uitvoeren van try-blok
  - Er treedt geen Exception op. Er wordt dus geen enkel catch-blok uitgevoerd.
  - Finally-blok wordt uitgevoerd.
- Daarna gaan we verder met de code onder het finally-blok.

Opgelet De uitvoer van dit programma ziet er niet altijd identiek uit.

```
Method throwException
Exception handled in method throwException
Finally executed in throwException
Exception handled in main
Finally executed in doesNotThrowException
Method doesNotThrowException
End of method doesNotThrowException
```

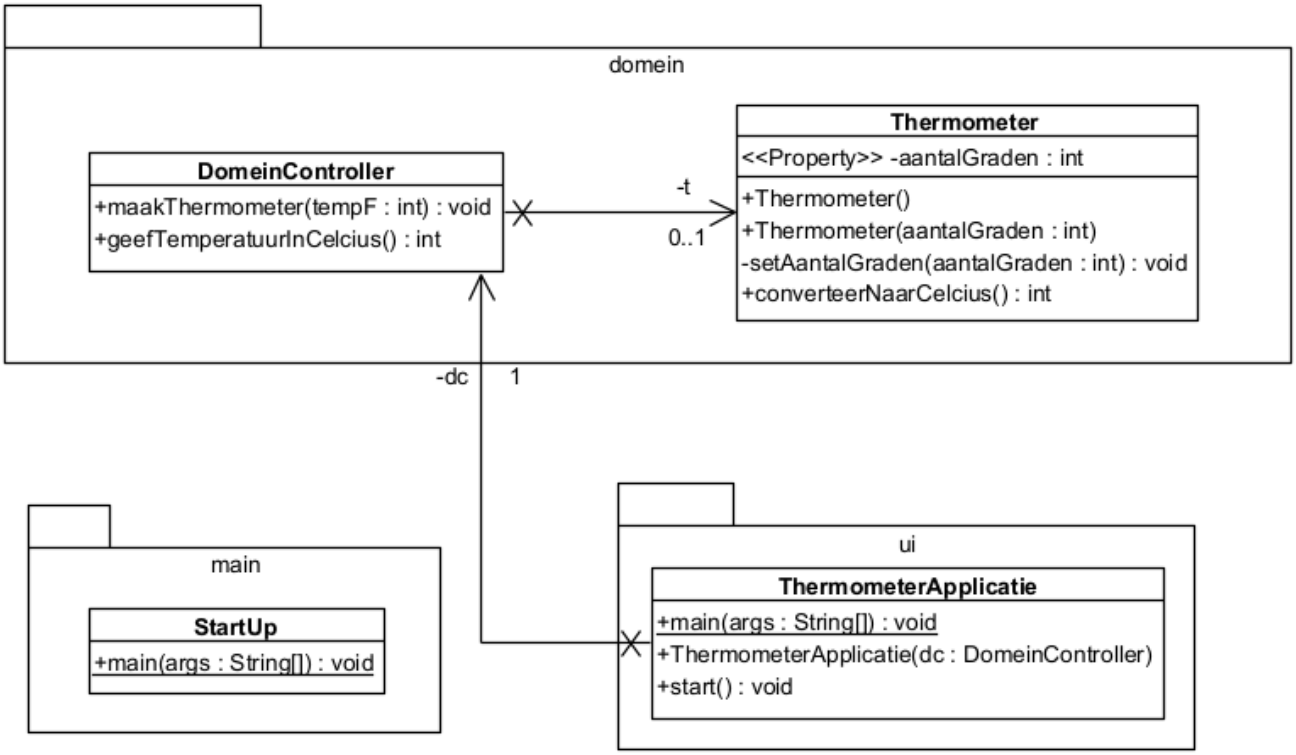
```
Exception handled in method throwExceptionMethod throwException

Finally executed in throwException
Exception handled in main
Finally executed in doesNotThrowException
Method doesNotThrowException
End of method doesNotThrowException
```

- **Zwart**: System.out (standard output stream)
- **Rood**: System.err (standard error stream)

Volgorde zwarte boodschappen en volgorde rode boodschappen ligt vast, maar de volgorde onderling kan altijd anders zijn.

4. Oefening - Afhandelen van een exception

4.1. Deel 1 - Fouten gooien en afhandelen in applicatie



Schrijf een consoleapplicatie die een temperatuur in Fahrenheit omzet in een temperatuur in Celsius.

Gebruik de domeinklasse Thermometer om de temperatuur om te zetten! (°C = 5/9 * (°F – 32)). De defaultconstructor maakt een thermometer aan waarbij de temperatuur ingesteld is op 32°F.

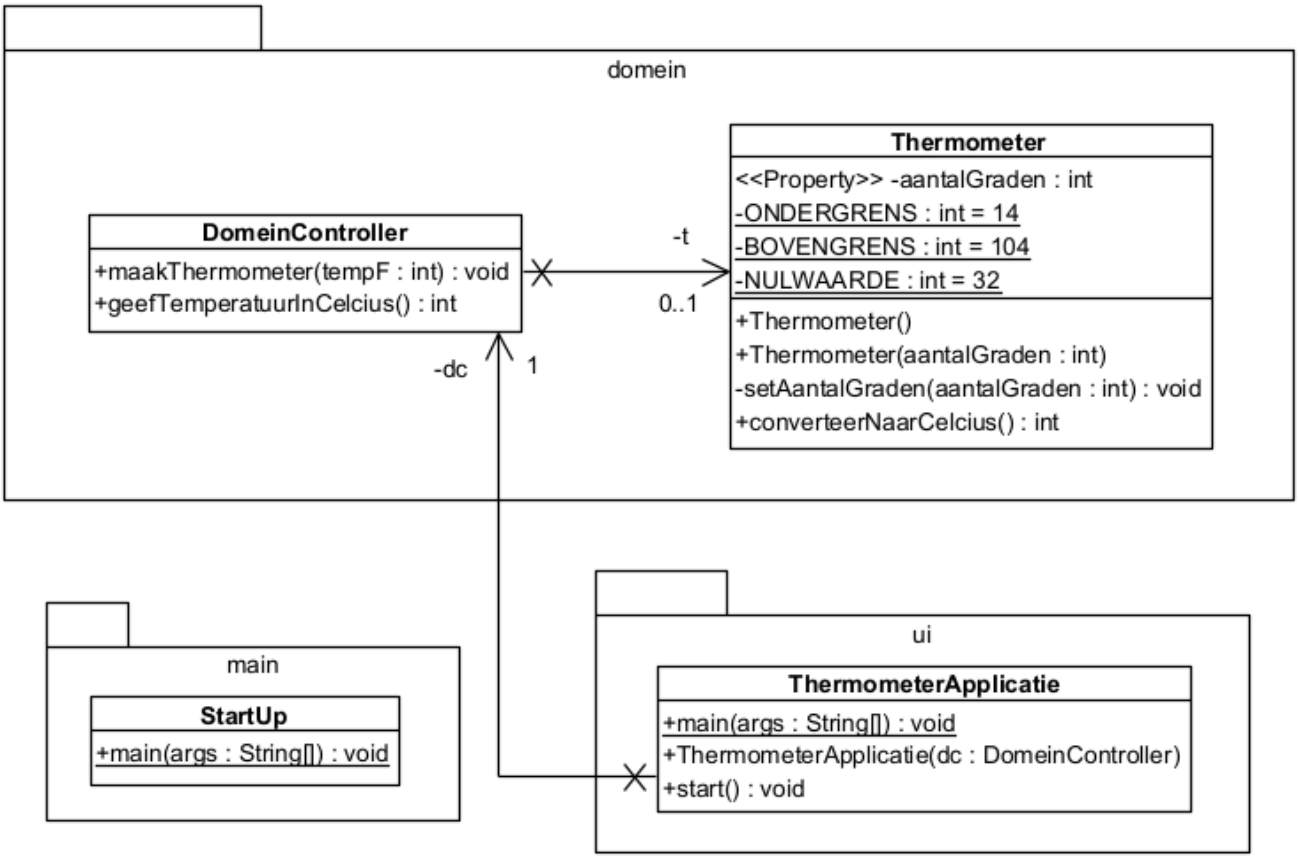
Zorg ervoor dat een foutieve input (= niet numeriek, buiten de grenzen van het interval [14°F,104°F]) wordt gemeld. Handel alle fouten ter plaatse (= in de applicatie) af!

Respecteer bovenstaande UML en gebruik dus ook een DomeinController-klasse als doorgeefluik tussen de ui en het domein. De applicatie kan gestart worden via de klasse StartUp uit de package main.

Verwachte uitvoer:

```
run:
Geef een temperatuur (eenheid Fahrenheit) in het interval [14,104]: blablabla
Foutieve invoer! Moet numeriek zijn!
Geef een temperatuur (eenheid Fahrenheit) in het interval [14,104]: 500
De temperatuur ligt niet in het gevraagde interval.
Geef een temperatuur (eenheid Fahrenheit) in het interval [14,104]: 10
De temperatuur ligt niet in het gevraagde interval.
Geef een temperatuur (eenheid Fahrenheit) in het interval [14,104]: 20
De temperatuur is -6°C
BUILD SUCCESSFUL (total time: 23 seconds)
```

4.2. Deel 2 - Fouten gooien deels in domein en deels in applicatie, afhandelen in applicatie

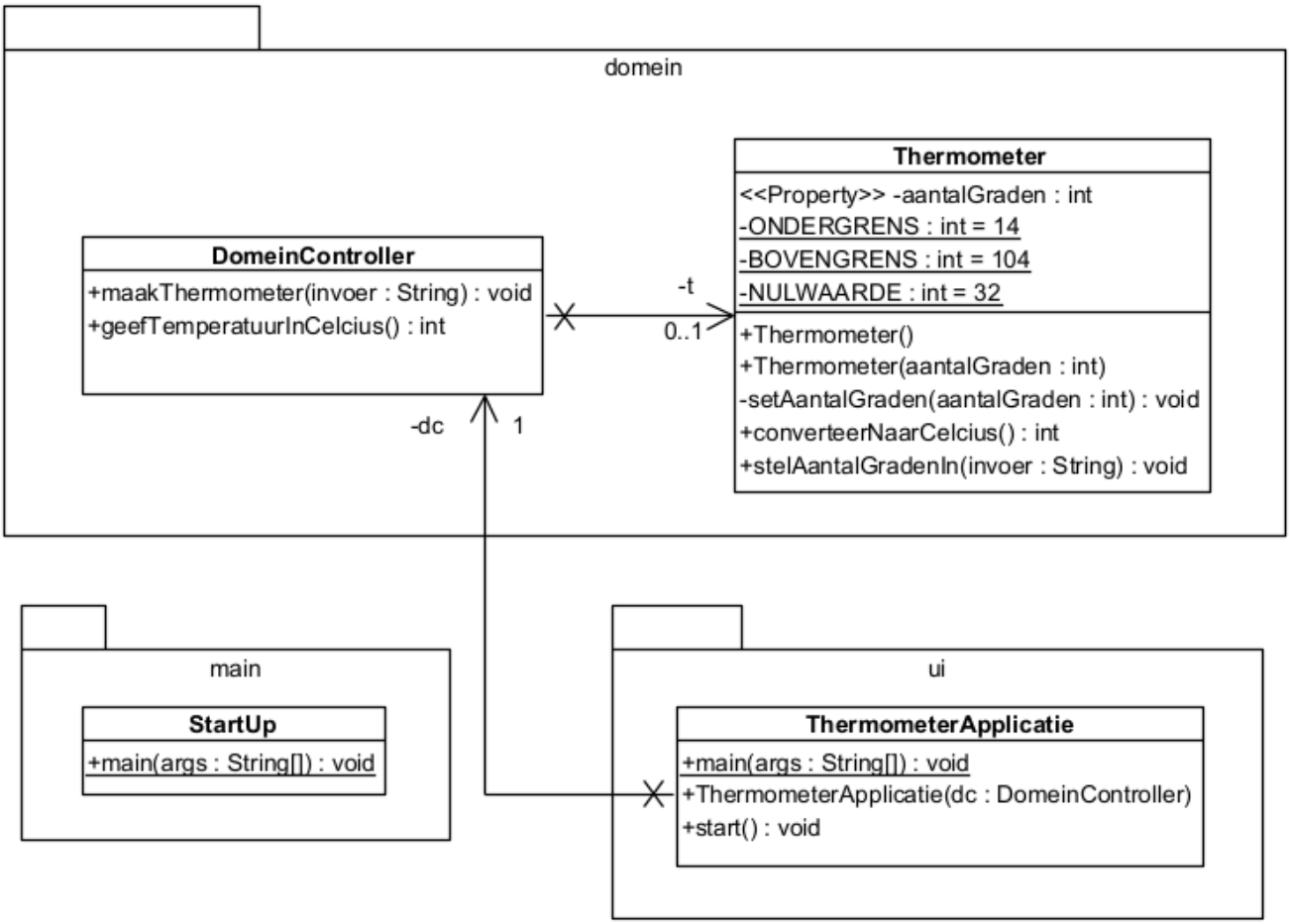


Wijzig deel 1 van de oefening

- Een niet-numerieke invoer handelen we nog altijd ter plaatse (= in de applicatie) af.
- Alle numerieke invoer wordt in eerste instantie aanvaard, maar in de klasse Thermometer moet op de waarde gecontroleerd worden. Blijkt de waarde buiten het interval te vallen, dan moet de (private) setter() een exceptie teruggooien. Deze wordt dan zoals in oefening1 afgehandeld.

De controle op het interval [14,104] is een domeinregel. Dit zijn de grenzen van de thermometer. Die controle MOET altijd in het domein zitten, het mag ook – extra bovenop de controle in het domein – opgenomen worden in de user interface.

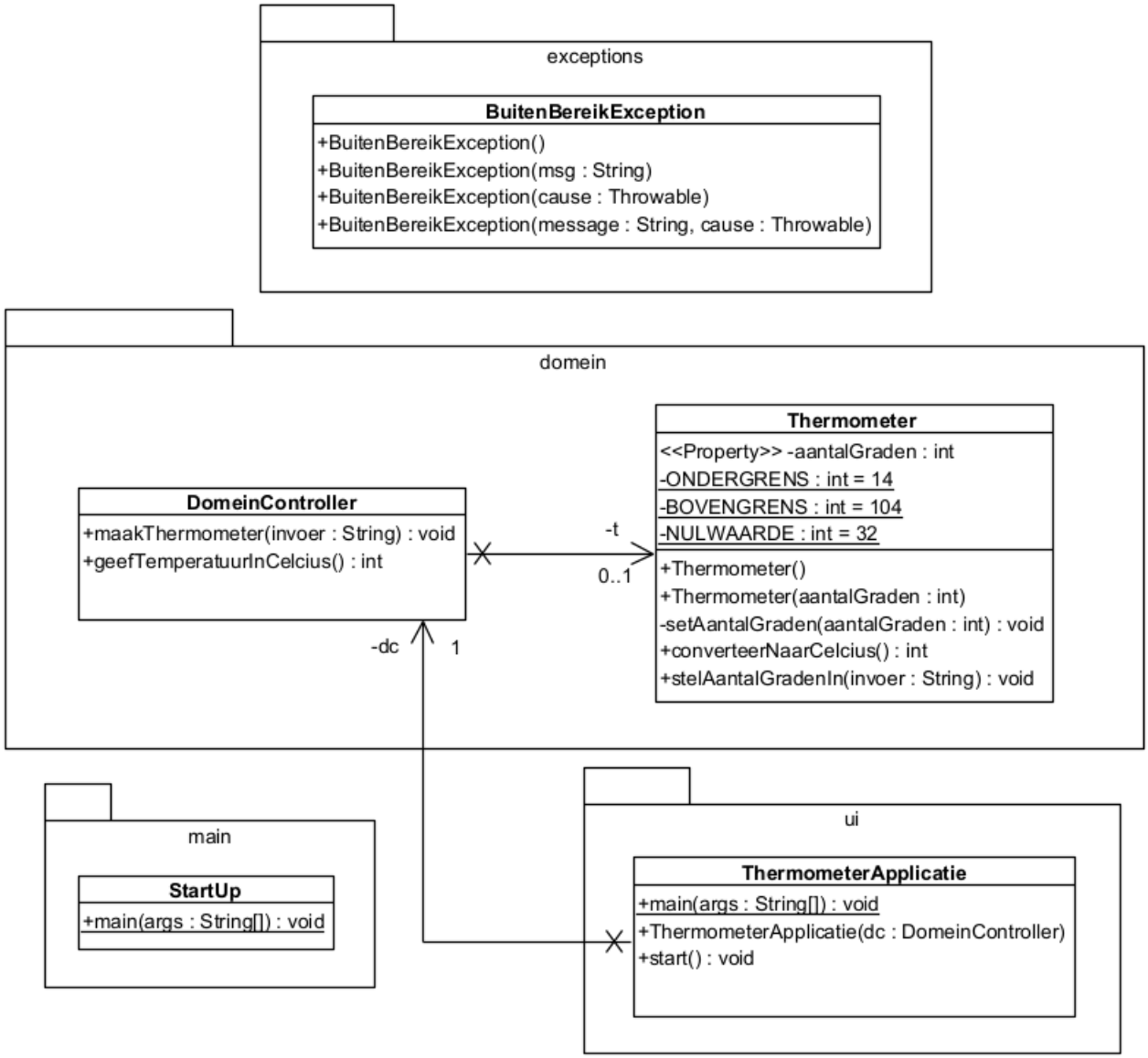
4.3. Deel 3 - Fouten gooien in domein, afhandelen in applicatie



Herneem deel 2 van de oefening:

- Elke invoer van de gebruiker wordt doorgestuurd naar het domein. Daar wordt indien nodig een Exception gegooid als de invoer fout was.
- Maak hiervoor een nieuwe methode in de klasse Thermometer: `stelAantalGradenIn(invoer:String):void`

4.4. Deel 4 - Schrijf je eigen Exception klasse



Herneem deel 3 van de oefening.

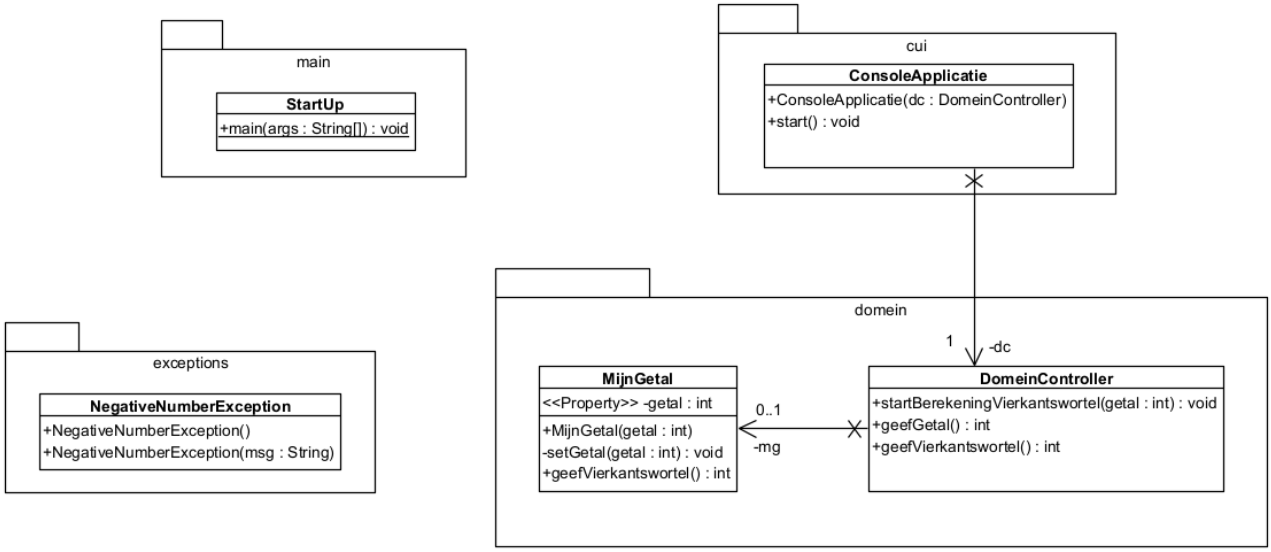
Schrijf je eigen exceptionklasse: BuitenBereikException, afgeleid van IllegalArgumentException

De exceptie wordt gegooid als de invoer niet voldoet aan de volgende voorwaarden:

- minimum 14°F
- maximum 104° F

Zorg ook voor een correcte afhandeling van deze fout.

5. Oefening - MijnGetal



Schrijf een eigen exceptieklasse NegativeNumberException, afgeleid van Exception

Gebruik deze klasse in een applicatie, die invoer krijgt via het toetsenbord.

De invoerlijn bevat normaal twee positieve gehele getallen. Splits de invoerlijn dus eerst op (zie tip). Krijg je geen twee elementen in de array, gooi dan een NoSuchElementException.

Probeer vervolgens de twee Strings om te zetten naar gehele getallen.

Als je zeker weet dat het om gehele getallen gaat, geef je ze door aan de DomeinController die er een MijnGetal-object van maakt. Hierbij controleer je of ze wel degelijk positief zijn (controle in klasse MijnGetal). Gooi een fout indien foutieve waarde.

Bepaal van deze beide getallen de gehele vierkantswortel.

```
run:
Geef twee gehele positieve getallen in (gescheiden door een spatie):
12
Twee getallen gescheiden door een spatie invoeren. Opnieuw!
Geef twee gehele positieve getallen in (gescheiden door een spatie):
a
Twee getallen gescheiden door een spatie invoeren. Opnieuw!
Geef twee gehele positieve getallen in (gescheiden door een spatie):
12.5 12.6
Zorg voor twee gehele getallen! Probeer opnieuw.
Geef twee gehele positieve getallen in (gescheiden door een spatie):
12 46
res1 = 3
res2 = 6
BUILD SUCCESSFUL (total time: 42 seconds)
```

Klasse String, methode split(...)

split

```
public String[] split(String regex)
```

Splits this string around matches of the given regular expression.

This method works as if by invoking the two-argument `split` method with the given expression and a limit argument of zero. Trailing empty strings are therefore not included in the resulting array.

The string `"boo:and:foo"`, for example, yields the following results with these expressions:

Regex	Result
:	{ "boo", "and", "foo" }
o	{ "b", "", ":and:f" }

Parameters:
`regex` - the delimiting regular expression

Returns:
the array of strings computed by splitting this string around matches of the given regular expression

Throws:
`PatternSyntaxException` - if the regular expression's syntax is invalid

Since:
1.4

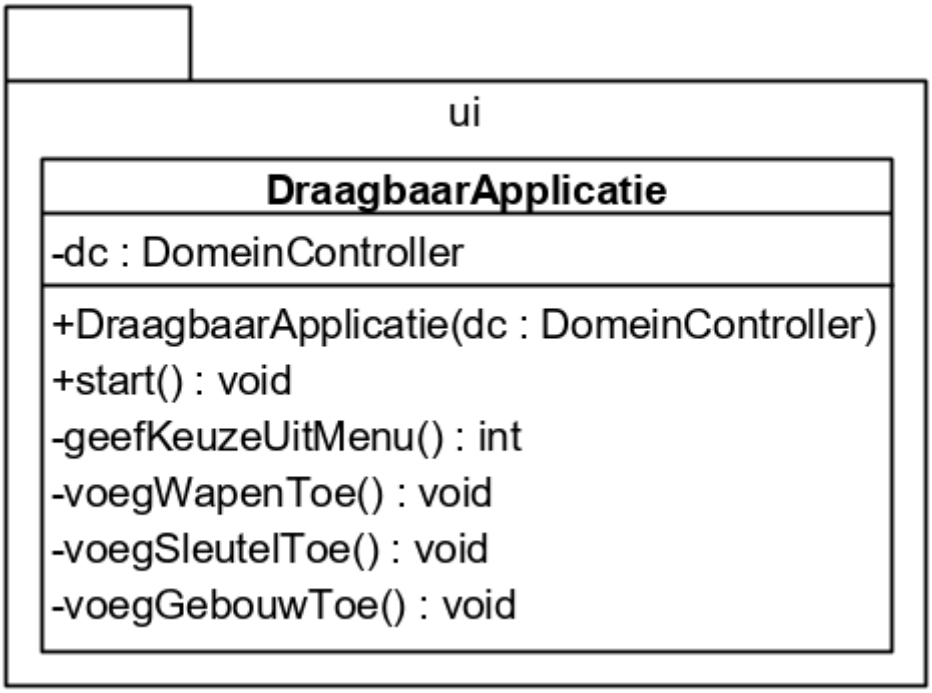
See Also:
`Pattern`

6. Oefening - Voorwerp

Werk verder op de oefening Voorwerp uit het hoofdstuk "Polymorfisme en interfaces". Vertrek van het project 'H04_Oef6_Voorwerp_Start'.

In de domeinklassen Voorwerp, Wapen, Sleutel en Gebouw zijn de nodige Exceptions toegevoegd. We maken nu ook een robuuste applicatie die alle mogelijke foutmeldingen correct afhandelt.

Het ontwerp van de applicatie ziet er als volgt uit:



6.1. Methode start()

Zorgt ervoor dat je telkens een menu krijgt waaruit je kan kiezen en dat afhankelijk van de keuze de juiste opdracht wordt uitgevoerd. Het menu wordt altijd opnieuw getoond tot de gebruiker

aangeeft dat hij wil stoppen (keuze 5).

Vb uitvoer (effectief toevoegen van een wapen nog niet aangevuld in deze stap)

```
run:
Kies uit:
1. Voeg wapen toe
2. Voeg sleutel toe
3. Voeg gebouw toe
4. Toon huidig overzicht
5. Beëindig deze applicatie
Je keuze is: erazr
Voer een geheel getal in.
Kies uit:
1. Voeg wapen toe
2. Voeg sleutel toe
3. Voeg gebouw toe
4. Toon huidig overzicht
5. Beëindig deze applicatie
Je keuze is: 6,2
Voer een geheel getal in.
Kies uit:
1. Voeg wapen toe
2. Voeg sleutel toe
3. Voeg gebouw toe
4. Toon huidig overzicht
5. Beëindig deze applicatie
Je keuze is: 1
Kies uit:
1. Voeg wapen toe
2. Voeg sleutel toe
3. Voeg gebouw toe
4. Toon huidig overzicht
5. Beëindig deze applicatie
Je keuze is: 5
BUILD SUCCESSFUL (total time: 2 minutes 54 seconds)
```

6.2. Methode geefKeuzeUitMenu()

Deze methode schrijft het keuzemenu uit. Zolang er fouten optreden in de invoer, wordt dit herhaald. Uiteindelijk heb je een correct cijfer (uit het interval [1-5]) en dat cijfer wordt teruggegeven.

6.3. Methodes om wapen, sleutel of gebouw toe te voegen

Vraag alle data op die nodig is om dat item aan te maken. Is er iets van de gegevens fout, dan stel je alle vragen voor dat item opnieuw.

```
run:
Kies uit:
1. Voeg wapen toe
2. Voeg sleutel toe
3. Voeg gebouw toe
4. Toon huidig overzicht
5. Beëindig deze applicatie
Je keuze is: 1
Geef een naam (zonder spaties): qfdsff
Geef het gewicht: 50,89
Geef het niveau:
4
Geef de kracht:
8
Werd het wapen reeds gebruikt (true/false)?
true
Kies uit:
1. Voeg wapen toe
2. Voeg sleutel toe
3. Voeg gebouw toe
4. Toon huidig overzicht
5. Beëindig deze applicatie
Je keuze is:
```

```
Kies uit:
1. Voeg wapen toe
2. Voeg sleutel toe
3. Voeg gebouw toe
4. Toon huidig overzicht
5. Beëindig deze applicatie
Je keuze is: 2
Geef een naam (zonder spaties): qsfdsf
Geef het gewicht: 2
Geef het niveau:
5
Geef het nummer van de deur:
103
Kies uit:
1. Voeg wapen toe
2. Voeg sleutel toe
3. Voeg gebouw toe
4. Toon huidig overzicht
5. Beëindig deze applicatie
```

```
Kies uit:
1. Voeg wapen toe
2. Voeg sleutel toe
3. Voeg gebouw toe
4. Toon huidig overzicht
5. Beëindig deze applicatie
Je keuze is: 3
Geef een naam (zonder spaties): qfdfsf
Geef de hoogte: 100
Kies uit:
1. Voeg wapen toe
2. Voeg sleutel toe
3. Voeg gebouw toe
4. Toon huidig overzicht
5. Beëindig deze applicatie
Je keuze is:
```

```
Kies uit:
1. Voeg wapen toe
2. Voeg sleutel toe
3. Voeg gebouw toe
4. Toon huidig overzicht
5. Beëindig deze applicatie
Je keuze is: 3
Geef een naam (zonder spaties): qfdsffqsdf
Geef de hoogte: -100
De hoogte moet minstens 3m zijn!
Geef een naam (zonder spaties): qsdfsdf
Geef de hoogte: 16
Kies uit:
1. Voeg wapen toe
2. Voeg sleutel toe
3. Voeg gebouw toe
4. Toon huidig overzicht
5. Beëindig deze applicatie
Je keuze is:
```

Keuze 4 schrijft alle items uit die op dat moment toegevoegd zijn.

```
Kies uit:
1. Voeg wapen toe
2. Voeg sleutel toe
3. Voeg gebouw toe
4. Toon huidig overzicht
5. Beëindig deze applicatie
Je keuze is: 4
Wapen qfdsff met gewicht 50,890 kg uit niveau 4 en met kracht 8 al gebruikt.

Sleutel qsfdsf met gewicht 2,000 kg uit niveau 5 past op deur 103. Er zijn 1 sleutel(s) in omloop.

qfdfsf met hoogte 100,0 is niet draagbaar.

qsdfsdf met hoogte 16,0 is niet draagbaar.
```

7. Oefening - Verplaatsing

Werk verder op de oefening Verplaatsing uit het hoofdstuk "Polymorfisme en interfaces". Vertrek van het project 'H04_Oef7_Verplaatsing_Start'.

In de domeinklassen Verplaatsing, VerplaatsingPerAuto, VerplaatsingPerBusTram en Ticket zijn de nodige Exceptions toegevoegd. We maken nu ook een robuuste applicatie die alle mogelijke foutmeldingen correct afhandelt zoals in

```
1 new VerplaatsingPerAuto("Gent", "Kortrijk", 47.1, 1.649, 0.084));//foutieve waarde!
```

Mogelijke uitvoeren:

Foutmelding in console door foutieve waarde in laatste object:

```
run:
Het verbruik moet tussen de 0.02 en de 0.07 liter per km liggen
BUILD SUCCESSFUL (total time: 0 seconds)
```

0.084 wijzigen in 0.034 levert volgende output op:

```
Volgende documenten werden ingediend:
2 ticket(s), 3 verplaatsing(en) per auto en 2 verplaatsing(en) per bus/tram.

Overzicht gemaakte kosten:

                                                    Kostenpost          Bedrag
verplaatsing van Voskenslaan Gent naar Veldstraat Gent met stadstram 1    2,02
                                          Parkeerticket Flanders Expo      5,00
                          verplaatsing van Gent naar Oudenaarde per auto     3,98
                                                 Toegangsticket beurs     10,00
             verplaatsing van Laarne dorp naar Gent Sint-Pieters met bus 34     7,58
                           verplaatsing van Gent naar Brussel per auto     8,74
                          verplaatsing van Gent naar Kortrijk per auto     4,17


Totaal gedeclareerde kosten =   41,49
BUILD SUCCESSFUL (total time: 0 seconds)
```

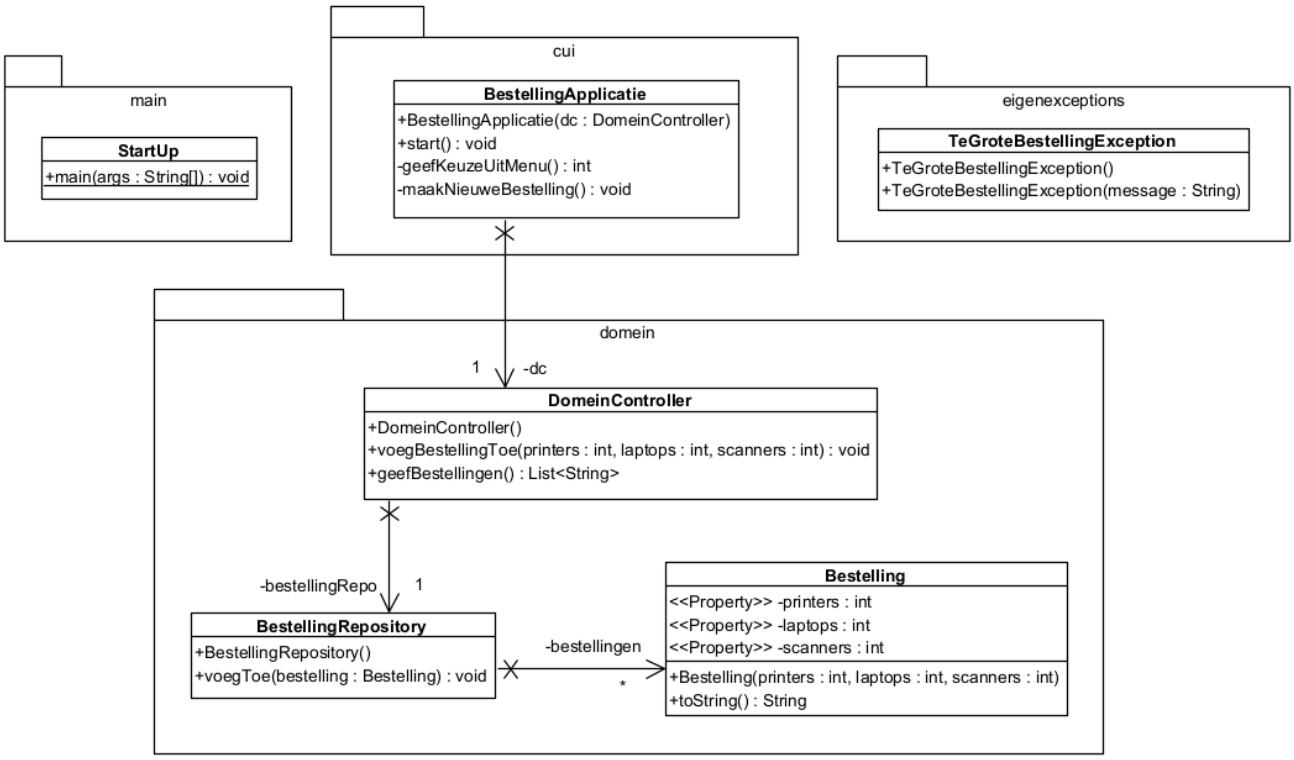
8. Oefening - Bestelling

Maak een robuuste applicatie waarbij je een maximum aantal printers, laptops en scanners kan bestellen. **Het totale aantal producten dat je per keer kan bestellen is 8.**

Let op: in 1 bestelling mogen **maximaal 4 printers** zitten. Als je laptops bestelt, kun je er **maximum 2** bestellen in één bestelling en bij **scanners** mag je **maximaal 3** exemplaren bestellen.

Indien je over de limieten gaat en/of een niet-numeriek karakter invoert, voorzie je de gepaste foutboodschappen. Maak een aparte klasse **TeGroteBestellingException** waaraan je de specifieke boodschappen doorgeeft. De superklasse van deze eigen gemaakte Exception is de klasse Exception.

Fouten worden in het domein gegenereerd en opgevangen in de applicatie.



Voorbeeld uitvoer:

```
run:
Kies uit:
1. Nieuwe bestelling
2. Overzicht bestellingen
3. Beëindig deze applicatie
Je keuze is: 1
Geef het aantal printers: 2
Geef het aantal laptops: 1
Geef het aantal scanners: 1
Kies uit:
1. Nieuwe bestelling
2. Overzicht bestellingen
3. Beëindig deze applicatie
Je keuze is: 2
Bestelling van 2 printers, 1 laptops en 1 scanners
Kies uit:
1. Nieuwe bestelling
2. Overzicht bestellingen
3. Beëindig deze applicatie
Je keuze is: 3
BUILD SUCCESSFUL (total time: 24 seconds)
```

```
run:
Kies uit:
1. Nieuwe bestelling
2. Overzicht bestellingen
3. Beëindig deze applicatie
Je keuze is: 1
Geef het aantal printers: 5
Geef het aantal laptops: 0
Geef het aantal scanners: 0
Aantal printers maximaal 4!
Kies uit:
1. Nieuwe bestelling
2. Overzicht bestellingen
3. Beëindig deze applicatie
Je keuze is: 3
BUILD SUCCESSFUL (total time: 36 seconds)
```

```
run:
Kies uit:
1. Nieuwe bestelling
2. Overzicht bestellingen
3. Beëindig deze applicatie
Je keuze is: 1
Geef het aantal printers: 3
Geef het aantal laptops: 3
Geef het aantal scanners: 3
Maximaal aantal te bestellen artikelen is 8!
Kies uit:
1. Nieuwe bestelling
2. Overzicht bestellingen
3. Beëindig deze applicatie
Je keuze is: 2
Kies uit:
1. Nieuwe bestelling
2. Overzicht bestellingen
3. Beëindig deze applicatie
Je keuze is: 3
BUILD SUCCESSFUL (total time: 35 seconds)
```

9. Oefening - Film

Pas de applicatie uit het hoofdstuk ivm polymorfisme aan zodat ze helemaal robuust is. We maken ook gebruik van een eigen exception klasse **LegeStringException**, die als superklasse de klasse Exception heeft.

Klasse LegeStringException Zorg ervoor dat een LegeStringException kan geworpen worden met als standaardboodschap "Alles moet ingevuld zijn", maar ook met een zelf gekozen foutboodschap.

Aanpassingen in de testklasses Als er geen Exception verwacht wordt, mag de fout gewoon verder gepropageerd worden. Wordt er wel een Exception verwacht, check dan of het juiste type gegooid wordt.

Aanpassingen in de domeinklasses Waar nodig dienen de juiste exceptions nog gepropageerd te worden.

Aanpassingen in de applicatieklasse Zorg ervoor dat de applicatie robuust is. Met andere woorden: alle exceptions moeten opgevangen worden. Zie ook onderstaande voorbeelduitvoer voor inspiratie.

Voorbeeld uitvoer:

```
1. Toon het overzicht van de voorstellingen
2. Voeg een voorstelling toe
3. Geef het aantal voorstellingen in een bepaalde zaal
4. Beëindig het programma
Geef je keuze: 2
1. Voeg een film toe
2. Voeg een concertregistratie toe
3. Keer terug naar hoofdmenu
Geef je keuze: 2
Geef de naam van de artiest:
Geef de benaming van het concert:
Alles moet ingevuld zijn
```

```
1. Toon het overzicht van de voorstellingen
2. Voeg een voorstelling toe
3. Geef het aantal voorstellingen in een bepaalde zaal
4. Beëindig het programma
Geef je keuze: 2
1. Voeg een film toe
2. Voeg een concertregistratie toe
3. Keer terug naar hoofdmenu
Geef je keuze: 1
Geef de naam van de film:
Geef het jaar waarin de film uitkwam: 2021
Hoeveel sterren verdient deze film: 2
Naam van de film mag niet leeg zijn.
```

```
1. Toon het overzicht van de voorstellingen
2. Voeg een voorstelling toe
3. Geef het aantal voorstellingen in een bepaalde zaal
4. Beëindig het programma
Geef je keuze: 2
1. Voeg een film toe
2. Voeg een concertregistratie toe
3. Keer terug naar hoofdmenu
Geef je keuze: 1
Geef de naam van de film: filmpje
Geef het jaar waarin de film uitkwam: 2030
Hoeveel sterren verdient deze film: 5
Het jaar waarin de film uitkomt moet in het interval [1900 - 2022] liggen
```

```
1. Toon het overzicht van de voorstellingen
2. Voeg een voorstelling toe
3. Geef het aantal voorstellingen in een bepaalde zaal
4. Beëindig het programma
Geef je keuze: 2
1. Voeg een film toe
2. Voeg een concertregistratie toe
3. Keer terug naar hoofdmenu
Geef je keuze: 1
Geef de naam van de film: filmpje
Geef het jaar waarin de film uitkwam: 2022
Hoeveel sterren verdient deze film: 7
Aantal sterren voor een film ligt in het interval [0,5]
```

```
1. Toon het overzicht van de voorstellingen
2. Voeg een voorstelling toe
3. Geef het aantal voorstellingen in een bepaalde zaal
4. Beëindig het programma
Geef je keuze: 2
1. Voeg een film toe
2. Voeg een concertregistratie toe
3. Keer terug naar hoofdmenu
Geef je keuze: 1
Geef de naam van de film: test
Geef het jaar waarin de film uitkwam: test
Hoeveel sterren verdient deze film: test
Aantal sterren en jaar moet een geheel getal zijn
```