Foutafhandeling

DE HOGESCHOOL
MET HET NETWERK

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

INHOUD

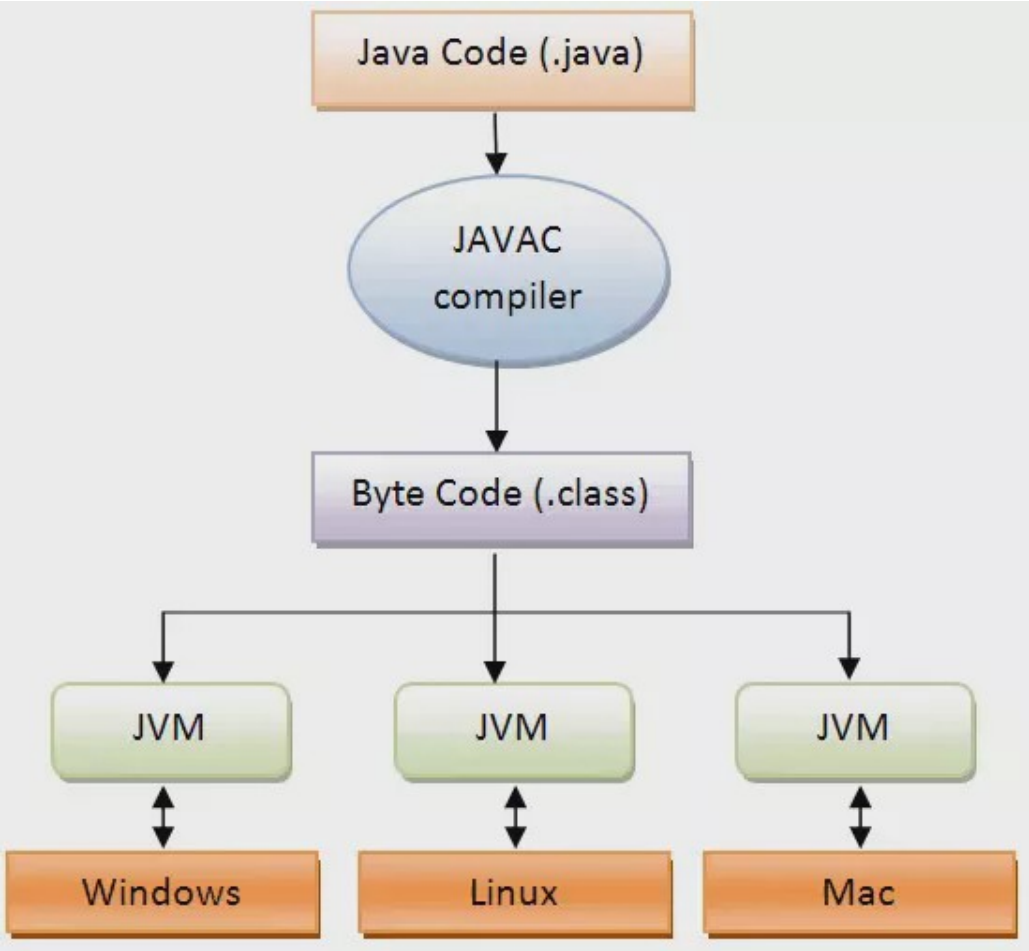
- Compile-time vs runtime
- Wat zijn exceptions?
- Exception handling
- Exception hiërarchie
  - Errors, checked en unchecked exceptions
- Multiple catches
- Finally
- Unit testen
- Throwing exceptions
- Eigen exceptions

Code op github:

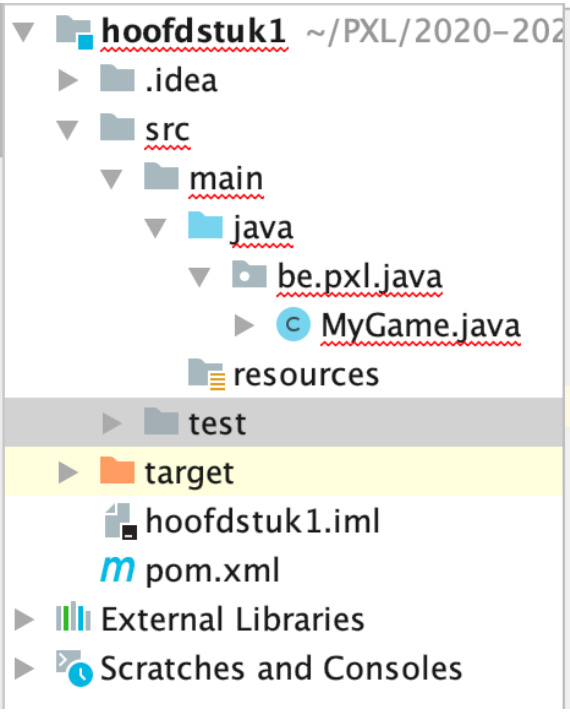
https://github.com/custersnele/

JavaAdv_H3_ExceptionHandling

Compile-time vs runtime



Compile-time vs runtime




```java
package be.pxl.java;

public class Game {

    public static Void main(String[] args) {
        System.out.println("If I choose Paper,");
        System.out.println("And you choose Scissors,")
        System.out.println("Then I win, and you lose!");
}

```

Wat zijn exceptions?

- Gebeurtenis tijdens uitvoeren van code
- Verstoort normale *flow*

- Exception wordt *gegooid* (throw)
- ... en kan opgevangen worden (*catch*)

- Indien niet opgevangen: programma stopt met uitvoeren

Wat zijn exceptions?

Code:

```
public class DivisionByZero {

  public static void main(String[] args) {
    int a = (1 + 1) % 2;
    int b = 5;
    int c = b / a;
    System.out.println("Het resultaat is " + c);
  }
}
```

Wat zijn exceptions?

Console toont stack trace & info:

```
Exception in thread "main"
java.lang.ArithmeticException: / by zero
   at
be.pxl.ja.DivisionByZero.main(DivisionByZero.java:8)
```

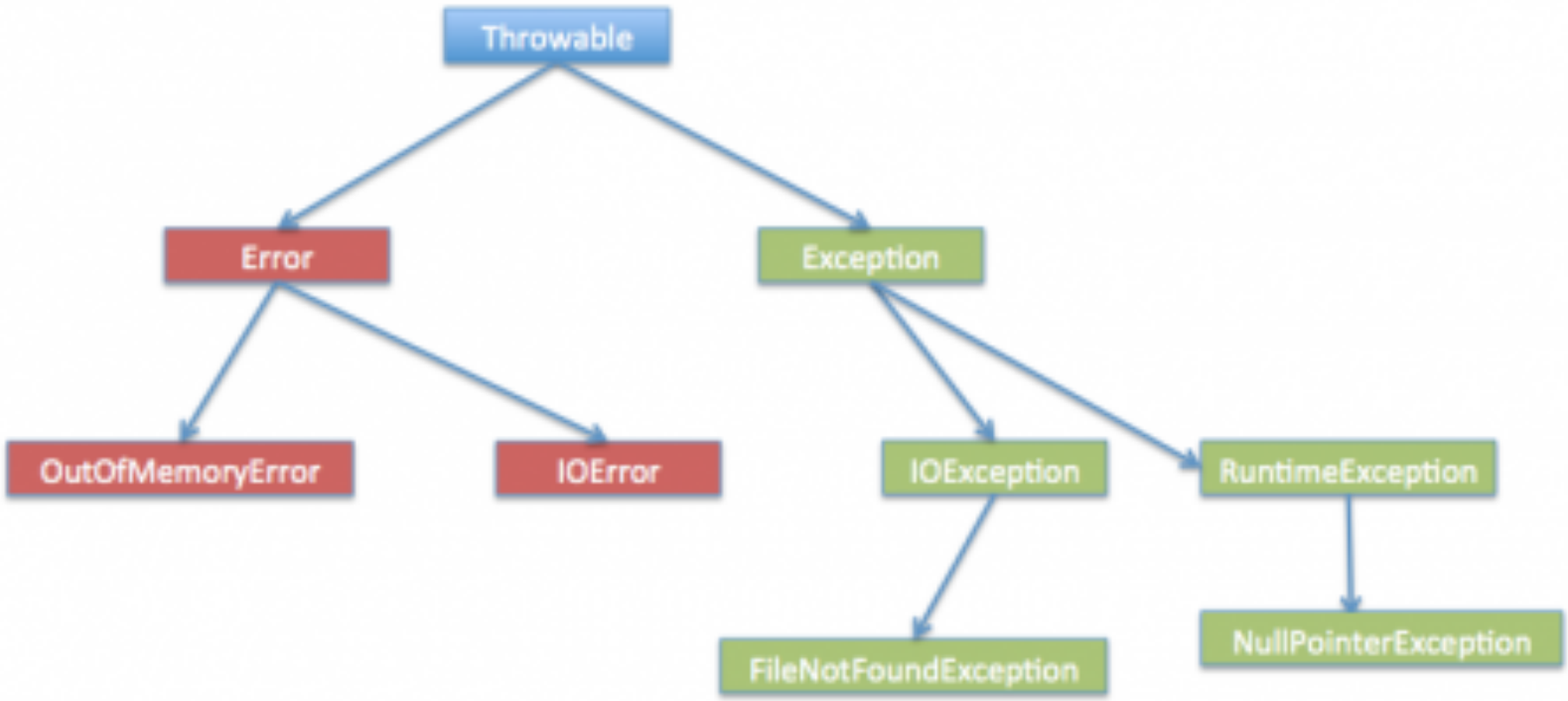
→ programma gestopt

Exception handling

```
public class DivisionByZero {

  public static void main(String[] args) {
    int a = (1 + 1) % 2;
    int b = 5;
    try {
      int c = b / a;
      System.out.println("Het resultaat is " + c);
    } catch (ArithmeticException e) {
      System.out.println("You should not divide a number by zero");
    }
    System.out.println("First catch completed!");
  }
}
```

Java exception hiërarchie

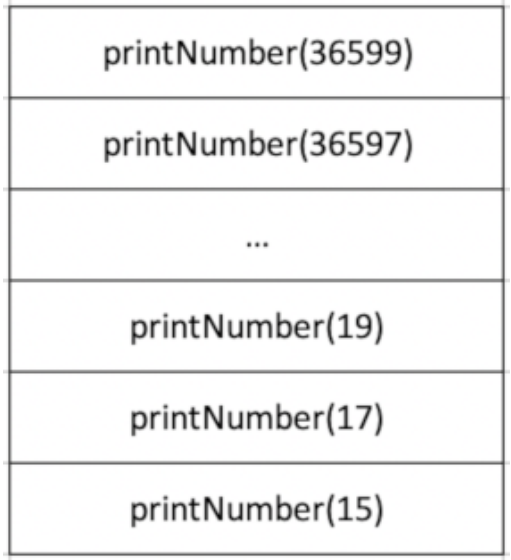


Java exception hiërarchie: Errors

```
public class DemoStackOverflow {

  private static void printNumber(int x) {
    System.out.println(x);
    printNumber(x + 2);
  }

  public static void main(String[] args) {
    printNumber(15);
  }
}
```



Java exception hiërarchie: Exceptions

- Unchecked exception
  - Runtime exception
  - Mogen opgevangen worden
  - bv. NullPointerException, IllegalArgumentException

- Checked exception
  - Compile time exception
  - Grote kans op fout
  - Moéten opgevangen worden
  - bv. IOException, …

Runtime exceptions

```
public class DemoInvalidArgumentException {

  public static void main(String[] args) {
    String tekst = "abc";
    System.out.println(tekst.repeat(-5));
  }
}
```

Checked exceptions

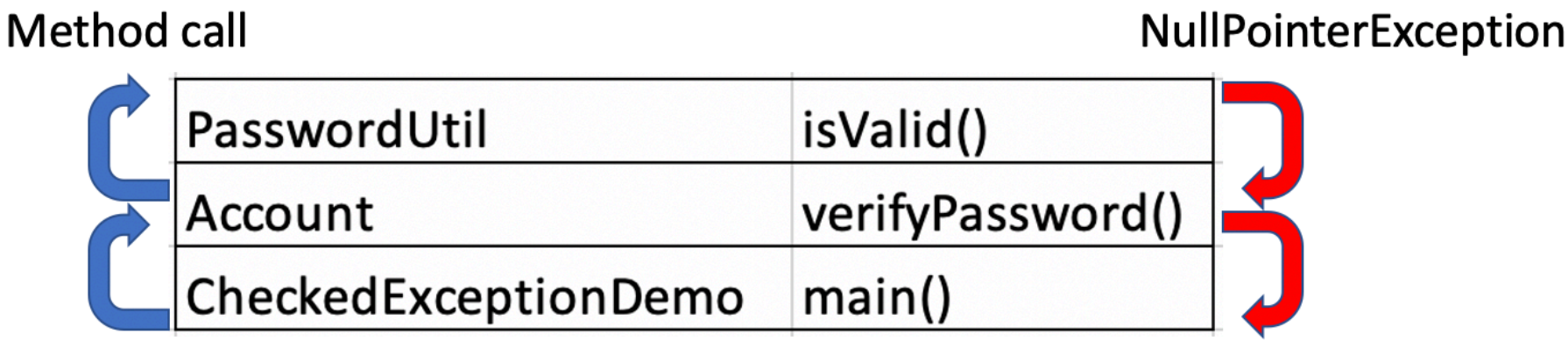
```
public class PasswordUtil {

  private static final String SPECIAL_CHARACTERS = "~!@#$%^&*()_-";
  private static final String ALGORITHM = "MD4";

  public static String encodePassword(String password)  {
    MessageDigest messageDigest = null;
    try {
      messageDigest = MessageDigest.getInstance(ALGORITHM);
    } catch (NoSuchAlgorithmException e) {
      // this is not ok!
      return null;
    }
    messageDigest.update(password.getBytes(), 0, password.length());
    return new BigInteger(1, messageDigest.digest()).toString(16);
  }
```

Exception handling – Call stack

```
Exception in thread "main" java.lang.NullPointerException
        at be.pxl.ja.streamingservice.util.PasswordUtil.isValid(PasswordUtil.java:25)
        at be.pxl.ja.streamingservice.model.Account.verifyPassword(Account.java:40)
        at be.pxl.ja.CheckedExceptionDemo.main(CheckedExceptionDemo.java:11)
```



Checked exceptions

```
public class PasswordUtil {

  private static final String ALGORITHM = "MD4";

  public static String encodePassword(String password)  throws
                                                        NoSuchAlgorithmException {
    MessageDigest messageDigest = MessageDigest.getInstance(ALGORITHM);
    messageDigest.update(password.getBytes(), 0, password.length());
    return new BigInteger(1, messageDigest.digest()).toString(16);
  }
```

Checked exceptions

```
public class PasswordUtil {

  private static final String ALGORITHM = "MD4";

  public static String encodePassword(String password)  {
    MessageDigest messageDigest = null;
    try {
      messageDigest = MessageDigest.getInstance(ALGORITHM);
    } catch (NoSuchAlgorithmException e) {
      throw new IllegalArgumentException(e);
    }
    messageDigest.update(password.getBytes(), 0, password.length());
    return new BigInteger(1, messageDigest.digest()).toString(16);
  }
```

Multiple catches en finally

```
public class MultipleCatches {

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Kies een positie: ");
    int positie = scanner.nextInt();
    System.out.println("Kies een deler: ");
    int deler = scanner.nextInt();
    try {
      int getallen[] = new int[10];
      getallen[positie] = 30 / deler;
    } catch (ArrayIndexOutOfBoundsException e) {
      System.out.println("Je moet een positie kiezen tussen 0 en 9.");
    } catch (Exception e) {
      System.out.println(e.getMessage());
    } finally {
      System.out.println("Je koos positie " + positie);
    }
    System.out.println("Start je het programma nog een keer.");
  }
}
```

Finally

Wordt **altijd** uitgevoerd, met of zonder exception.

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

Unit testen

```
@Test
void testExpectedException() {
  Assertions.assertThrows(NumberFormatException.class, () -> {
    Integer.parseInt("One");
  });
}
```

Throwing exceptions

```
public class CreditCardNumber {
  private static final int LENGTH = 16;

  private CreditCardType type;
  private String number;
  private String cvc;

  public CreditCardNumber(String number, String cvc) {
    number = removeBlanks(number);
    if (!isNumeric(number) || number.length() != LENGTH) {
      throw new IllegalArgumentException(”Must have " + LENGTH + " digits.");
    }
    this.number = number;
    type = getCreditCardType(number);
    if (type == null) {
      throw new IllegalArgumentException("This is not a valid credit card.");
    }
    this.cvc = removeBlanks(cvc);
  }
  …
```

Unit testen

```
@Test
public void validMasterCardWithBlanks() {
  CreditCardNumber creditCardNumber = new CreditCardNumber(
                                                  "  53218 76532 1476 54  ", " 1 2 3 ");

  assertEquals(CreditCardType.MASTERCARD, creditCardNumber.getType());
  assertEquals("123", creditCardNumber.getCvc());
  assertEquals("5321876532147654", creditCardNumber.getNumber());
}

@Test
public void throwsInvalidArgumentExceptionWhenNumberTooShort() {
  assertThrows(IllegalArgumentException.class, () -> {
    new CreditCardNumber("  53218 76532 1476  ", " 1 2 3 ");
  });
}
```

Oefening

- Gooi ook een IllegalArgumentException indien de cvc niet bestaat uit 3 digits.
- Voeg ook een unit test toe.

Eigen exceptions

```
public class InvalidDateException extends RuntimeException {

  public static final DateTimeFormatter FORMATTER =
                                        DateTimeFormatter.ofPattern("dd/MM/yyyy");

  public InvalidDateException(LocalDate incorrectDate, String type, String description) {
    super(FORMATTER.format(incorrectDate) + " is not a valid " + type + ". " +
                                                                    description);
  }
}
```

Eigen exceptions

```
public class PaymentInfo {

  private String firstName;
  private String lastName;
  private CreditCardNumber cardNumer;
  private LocalDate expirationDate;

  …

  public void setExpirationDate(LocalDate expirationDate) {
    if (LocalDate.now().plusMonths(1).isAfter(expirationDate)) {
      throw new InvalidDateException(expirationDate,
                                     "expirationDate", "Must be valid for at least 1 month.");
    }
    this.expirationDate = expirationDate;
  }
}
```

Oefening

- In de klasse Profile mag je geen geboortedatum in de toekomst meegeven in de methode setDateOfBirth(). Gooi een InvalidDateException indien dit wel gebeurt.
- Schrijf unit testen.