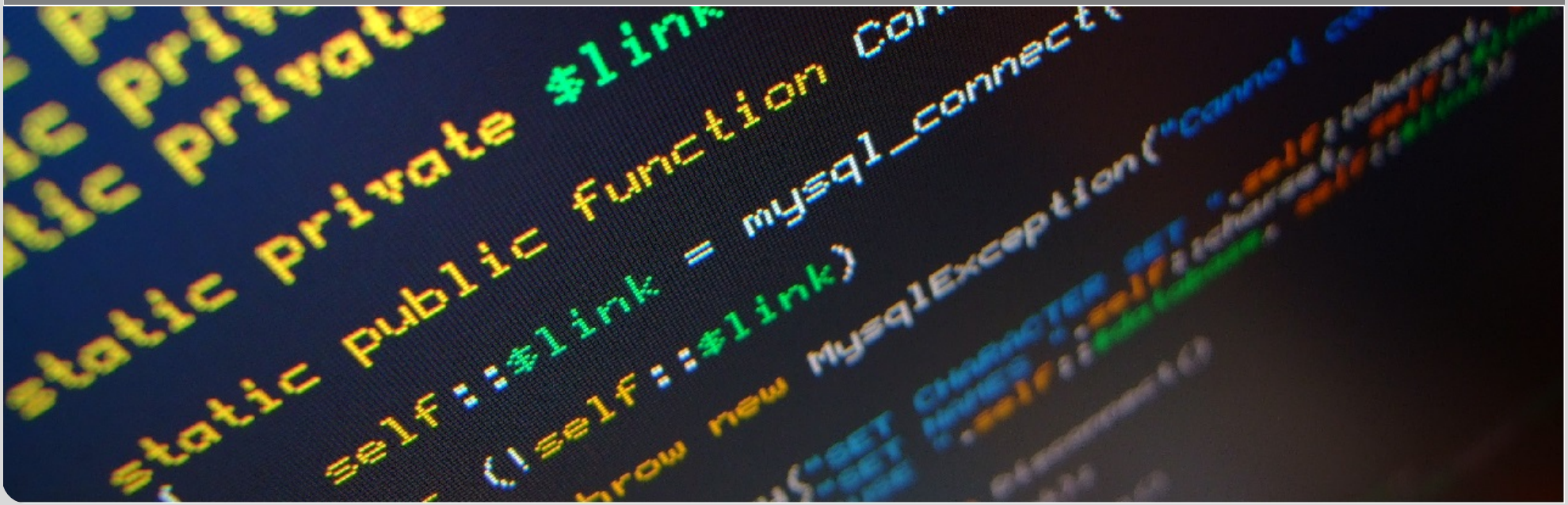
Karlsruher Institut für Technologie

Tutorium Programmieren

Tut Nr.11: Exceptions, java.util

Michael Friedrich | 21. / 23.11.2013

INSTITUT FÜR THEORETISCHE INFORMATIK



Outline/Gliederung

Fehlerbehandlung

Wichtig bei:

- Öffnen einer Datei, welche nicht existiert
- Zugriff auf Array-Element mit ungültigem Index
- Division durch Null
- ...

Fehlerbehandlung

Wichtig bei:

- Öffnen einer Datei, welche nicht existiert
- Zugriff auf Array-Element mit ungültigem Index
- Division durch Null
- ...

Probleme:

- Fehler abfangen, weiterleiten, behandeln?
- Wie Programmfluss dadurch nicht stören?

Fehlerbehandlung

Eine Möglichkeit: Prüfen mittels if-Abfragen

```
1 public int positivMultiply(int x, int y) {
2   if ((x>0 && y<0) || (x<0 && y>0)) {
3     System.out.println("Fehler: ein Argument ist negativ");
4     return -1;
5   } else {
6     return x*y;
7   }
8 }
9
```

Fehlerbehandlung

Eine Möglichkeit: Prüfen mittels if-Abfragen

```
1 public int positivMultiply(int x, int y) {
2   if ((x>0 && y<0) || (x<0 && y>0)) {
3     System.out.println("Fehler: ein Argument ist negativ");
4     return -1;
5   } else {
6     return x*y;
7   }
8 }
9
```

Funktioniert, aber:

- Stört Kontrollfluss
- evtl. Ausgaben, wo man sie nicht haben will bzw. sollte!

  $\Rightarrow$ Vermischung von Logik und Fehlerbehandlung

Exceptions

Exceptions

- Werden in Fehlersituationen zur Laufzeit ausgelöst („geworfen“)
- Können in einem übergeordneten Block abgefangen und behandelt werden
- Unterbrechen den Kontrollfluss

Vorteile:

+ Fehler muss nicht lokal behandelt werden, sondern kann weitergereicht werden
+ saubere Trennung von Logik und Fehlerbehandlung

Exceptions

- Sind Objekte (*Exception* und Unterklassen)

  $\Rightarrow$ werden mit `new` erzeugt

Exceptions

- Sind Objekte (*`Exception`* und Unterklassen)

  $\Rightarrow$ werden mit `new` erzeugt
- werden mit `throw` geworfen

Exceptions

- Sind Objekte (*Exception* und Unterklassen)

  $\Rightarrow$ werden mit `new` erzeugt
- werden mit `throw` geworfen
- Methode, welche den Fehler erzeugt (wirft), wird mit `throws` gekennzeichnet

```
public void methode(..) throws IllegalArgumentException
```

Exceptions

- Sind Objekte (*Exception* und Unterklassen)

  $\Rightarrow$ werden mit `new` erzeugt
- werden mit `throw` geworfen
- Methode, welche den Fehler erzeugt (wirft), wird mit `throws` gekennzeichnet

  ```
  public void methode(..) throws IllegalArgumentException
  ```
- Wird die Exception nicht in der Methode selbst gecatched, so wird sie an die aufrufende Methode weitergeleitet

Exceptions

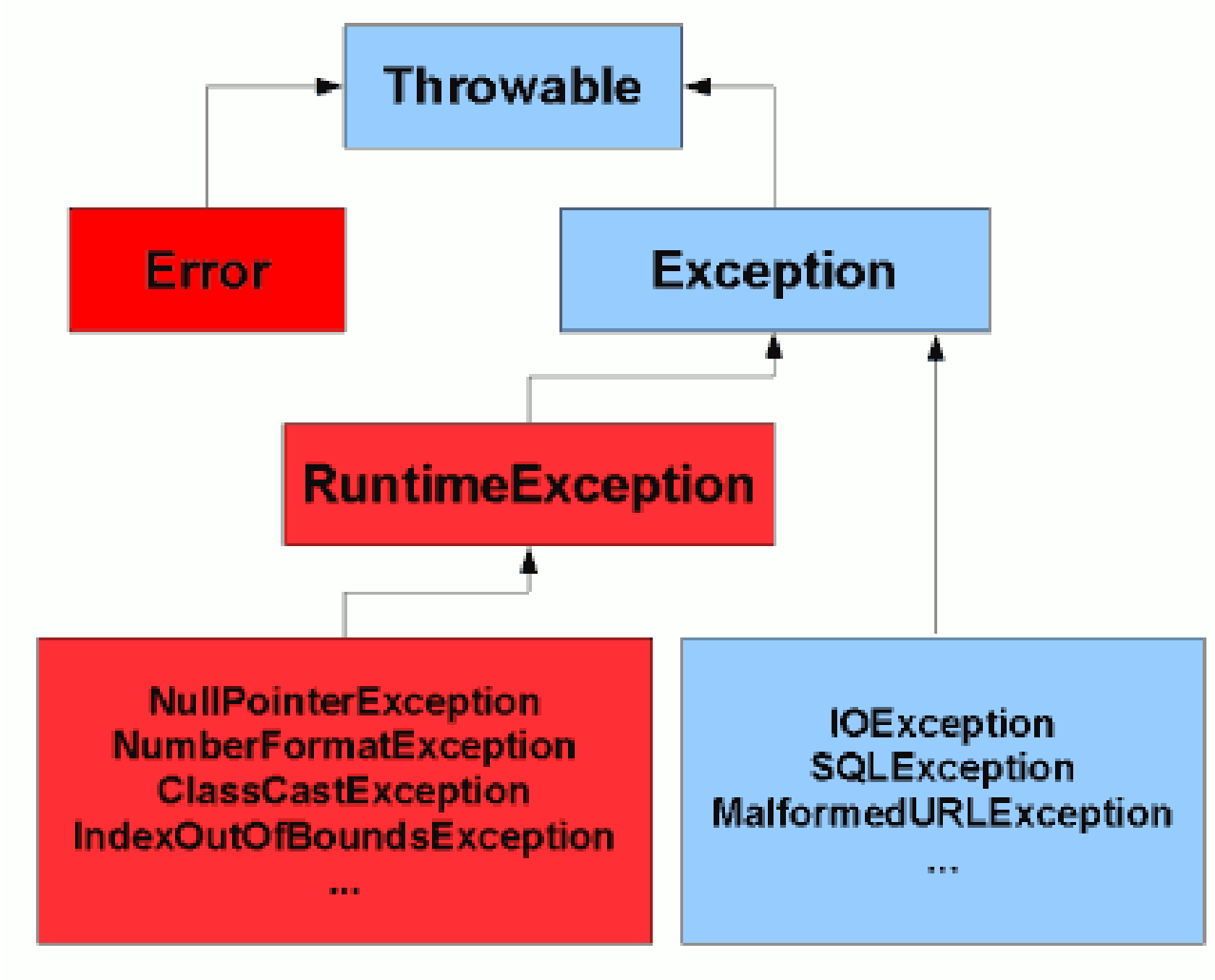
- Sind Objekte (*Exception* und Unterklassen)

  $\Rightarrow$ werden mit `new` erzeugt
- werden mit `throw` geworfen
- Methode, welche den Fehler erzeugt (wirft), wird mit `throws` gekennzeichnet

```
public void methode(..) throws IllegalArgumentException
```

- Wird die Exception nicht in der Methode selbst gecatched, so wird sie an die aufrufende Methode weitergeleitet
- Viele Exceptions in Java-API vorhanden, eigene sind möglich
- Java-Doc dokumentation mittels *@throws*

Exceptions: Vererbungshierarchie



Exceptions: Verwendung

```
1 public int positivMultiply(int x, int y) throws
      IllegalArgumentException {
2   if ((x>0 && y<0) || (x<0 && y>0)) {
3     throw new IllegalArgumentException();
4   }
5   return x * y;
6 }
7
```

Exceptions: Verwendung

```
1 public int positivMultiply(int x, int y) throws
      IllegalArgumentException {
2   if ((x>0 && y<0) || (x<0 && y>0)) {
3     throw new IllegalArgumentException();
4   }
5   return x * y;
6 }
7
```

An **anderer (aufrufender) Stelle**, wo der Fehler behandelt werden soll:

```
1   try {
2     int z = positivMultiply(x,y);
3   } catch (IllegalArgumentException e) {
4     System.out.println("Fehler: ein Argument ist negativ");
5   }
6
```

For when you just Gotta Catch 'Em All

```
1  try {
2     ...
3  } catch (Exception e) {
4     // Gotcha!
5  }
6
```

Exceptions vs Errors

RuntimeException

- Programmierfehler, der zur Laufzeit festgestellt wird, teilweise behandelbar
- *NullPointerException*, *ArrayIndexOutOfBoundsException*

Errors

- kritische Fehler bei Programmausführung
- Nicht behandelbar!
- *VirtualMachineError*, *OutOfMemoryError*

sonstige Exceptions

- behandelbare Programmfehler, sollte man auf jeden Fall abfangen

Exceptions: finally

optionaler Block, welcher auf jeden Fall ausgeführt wird.

```
1 try {
2   openFile();
3 } catch (IOException e) {
4   printError("File couldn't be read!");
5 } finally {
6   closeFile();
7 }
8
```

Exceptions

```
1  public static void main(String[] args) {
2     String input = "";
3     BufferedReader buf = new BufferedReader( new
       InputStreamReader(System.in));
4     while(!input.equals("quit")) {
5       try {
6         input = buf.readLine();
7       } catch (IOException e1) {
8         System.out.println("unable to read - shutting down...");
9   input = "quit";
10      }
11      try {
12        int i = Integer.parseInt(input);
13      } catch (NumberFormatException e) {
14        System.out.println("Eingabe war keine Zahl!");
15      }
16    }
17    System.out.println("shutting down...");
18 }
```

Was war bei dem Beispiel wichtig?!

Was war bei dem Beispiel wichtig?!

- Programm stürzt nie unkontrolliert ab

Was war bei dem Beispiel wichtig?!

- Programm stürzt nie unkontrolliert ab
- User wird über seine Fehler informiert und kann diese verbessern

Was war bei dem Beispiel wichtig?!

- Programm stürzt nie unkontrolliert ab
- User wird über seine Fehler informiert und kann diese verbessern

Was war bei dem Beispiel wichtig?!

- Programm stürzt nie unkontrolliert ab
- User wird über seine Fehler informiert und kann diese verbessern

Wir als Programmierer müssen unfähigen User (und Kollegen...) entgegen arbeiten.
Beispiel?

Was war bei dem Beispiel wichtig?!

- Programm stürzt nie unkontrolliert ab
- User wird über seine Fehler informiert und kann diese verbessern

Wir als Programmierer müssen unfähigen User (und Kollegen...) entgegen arbeiten.
Beispiel? NullPointer abfangen, falsche Werte geliefert, falsche Formattierung...

java.util

Java bietet von sich aus schon sehr viel Funktionalität, z.Bsp LinkedList.

java.util

Java bietet von sich aus schon sehr viel Funktionalität, z.Bsp LinkedList.

Example

weitere eingebaute Datenstrukturen

java.util

Java bietet von sich aus schon sehr viel Funktionalität, z.Bsp LinkedList.

Example

weitere eingebaute Datenstrukturen

- Collections: ungeordneter Pool an Objekten
  - `Collection<Product> products;`

java.util

Java bietet von sich aus schon sehr viel Funktionalität, z.Bsp LinkedList.

Example

weitere eingebaute Datenstrukturen

- Collections: ungeordneter Pool an Objekten
  - `Collection<Product> products;`
- SortedSet: Menge mit totaler Ordnung
  - `SortedSet<Product> products;`

java.util

Java bietet von sich aus schon sehr viel Funktionalität, z.Bsp LinkedList.

Example

weitere eingebaute Datenstrukturen

- Collections: ungeordneter Pool an Objekten
  - `Collection<Product> products;`
- SortedSet: Menge mit totaler Ordnung
  - `SortedSet<Product> products;`
  - Product MUSS hier Comparable<Product> implementieren

java.util

Java bietet von sich aus schon sehr viel Funktionalität, z.Bsp LinkedList.

Example

weitere eingebaute Datenstrukturen

- Collections: ungeordneter Pool an Objekten
  - `Collection<Product> products;`
- SortedSet: Menge mit totaler Ordnung
  - `SortedSet<Product> products;`
  - Product MUSS hier Comparable<Product> implementieren
- ArrayList: ähnlich LinkedList, aber mit Index
  - `ArrayList<Product> products;`

java.util

Java bietet von sich aus schon sehr viel Funktionalität, z.Bsp LinkedList.

Example

weitere eingebaute Datenstrukturen

- Collections: ungeordneter Pool an Objekten
  - `Collection<Product> products;`
- SortedSet: Menge mit totaler Ordnung
  - `SortedSet<Product> products;`
  - Product MUSS hier Comparable<Product> implementieren
- ArrayList: ähnlich LinkedList, aber mit Index
  - `ArrayList<Product> products;`
- Maps: key-value Paare
  - `TreeMap<Product, Customer> orders;`
  - `HashMap<Product, Customer> orders;`

Nehmt die Hinweise auf dem Übungsblatt als Einstiegspunkt.