Übungsstunde 1

Programm für heute

- Vorstellen
- Assignment1 „besprechen“
- Wiederholung Exceptions
- Beispiele zu Exceptions

Kurz über mich

- Sara Kilcher
- 4. Semester
- Mail: sakilche
- Webseite: www.mymountains.ch/pp.html
  - Slides
  - Zusatzübungen
  - Links

Kurz über euch

- Name
- Erfahrungen mit Java
- Sonstiges

Assignment 1

- Wer hat die Serie schon gelöst?
- Hatte jemand Probleme mit dem SVN?

- Bitte ladet eure Serie bis Morgen hoch

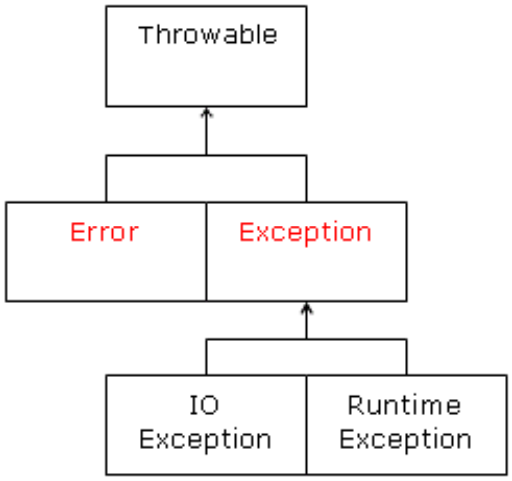
- Mulö „besprechen“

Exceptions

Eine Exception auslösen [„werfen“]

- Verwende das keyword throw
  - `throw new Throwable();`
- Nur Throwable-Objekte oder Objekte von Klassen, die von Throwable ableiten, können geworfen werden.

Hierarchie



- Kurzer Blick aufs API
- Exceptions sind Objekte in Java
- Errors: z.Bsp OutOfMemoryError
  - (von der Java virtual Machine geworfen)
- Euer Code sollte nur Exceptions werfen
- Runtime Exception: ausgelöst durch fehlerhafte Programme, z.Bsp:
  - fehlerhafte Typkonvertierung
  - Zugriff über die Arraygrenze hinaus
  - Zugriff auf einen leeren Zeiger

Syntax von try, catch und finally

```
try
{  // Anweisungen
}
catch(Typ1 Objekt)
{  // Anweisungen
}
catch(Typ2 Objekt)
{  // Anweisungen
}
finally
{  //Anweisung, die immer
   ausgeführt wird
}
```

try-Block markiert einen Bereich, in dem Exceptions abgefangen werden sollen.

catch-Block wird ausgeführt, wenn ein entsprechender Fehler auftritt.

Mehrere catch Blöcke sind möglich:

Erster Block, dessen **Typ** das geworfene Objekt aufnehmen kann, wird ausgeführt.

finally-Block wird immer ausgeführt, also:

- wenn keine Exception geworfen wird
- wenn eine Exception gefangen wurde
- wenn eine Exception nicht gefangen wurde (und vielleicht weiter oben im Stack dann gefangen wird)

Kurzes Beispiel

```
public class P01 {
   public static void main(String[] args) {
        try {   for (int i = 0; i <= args.length; i++) {
                    System.out.println(Integer.parseInt(args[i]) + 1);
                }
        } catch (ArrayIndexOutOfBoundsException e) {
              System.err.println("Out of Bounds: " + e.getMessage());
        } catch (NumberFormatException e) {
                System.out.println("You have to enter Integers as
                arguments. „ + e.getMessage());
        } finally {
                System.out.println("finally wird ausgeführt.");
        }
   }
}
```

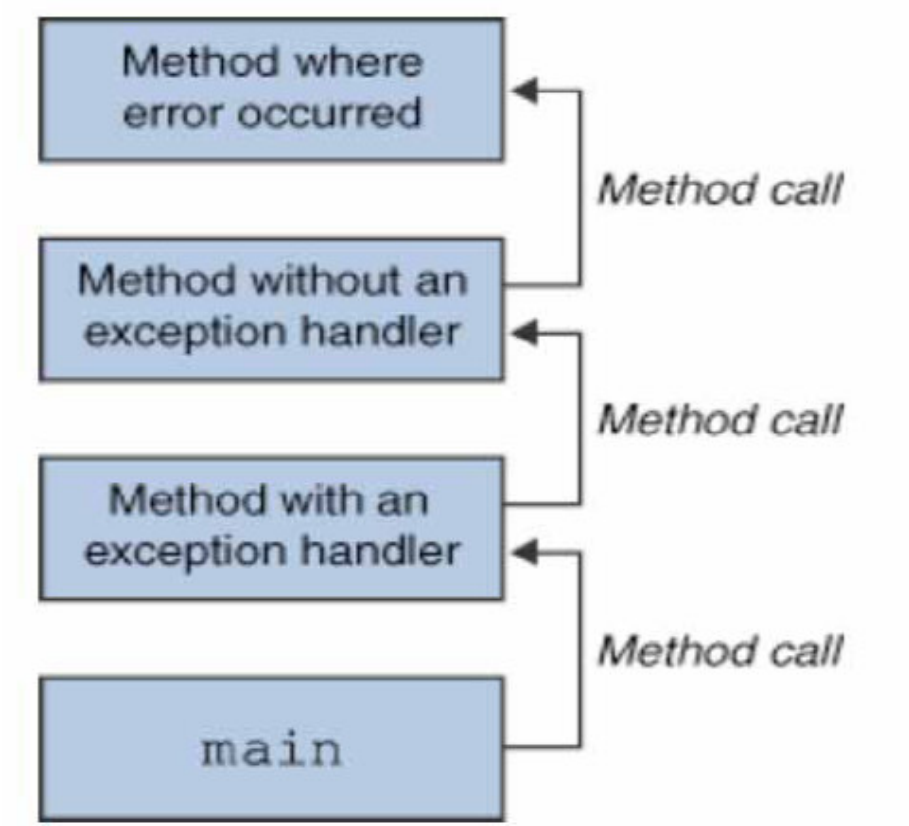
Methoden und Exceptions

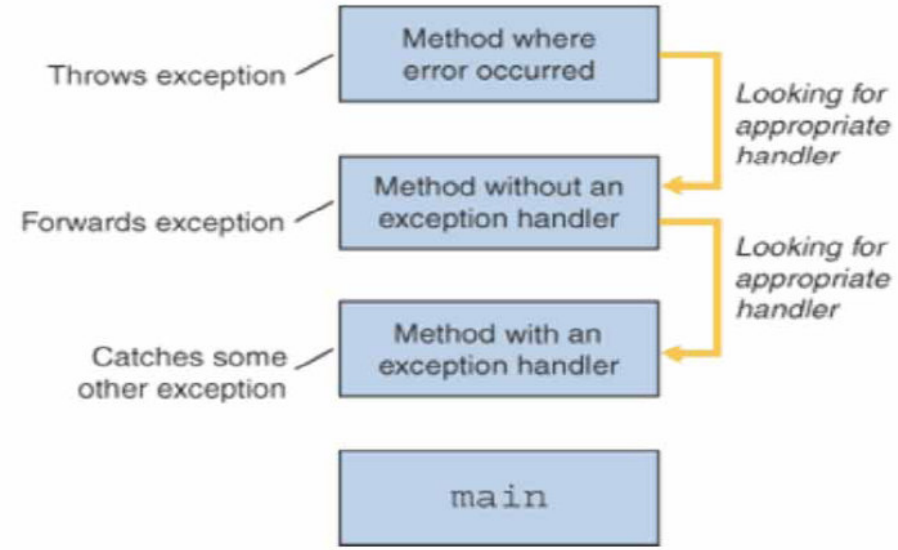
- Methoden sollten deklarieren, welche Exceptions sie auslösen können:

```
public void test() throws
   NumberFormatException, NullPointerException
{
   ...
}
```

- Ausnahme: sogenannte unchecked-Exceptions
  - RuntimeException, Error
  - Sowie Klassen, die von einer der beiden erben

Exception Handling





Corner Case 1

Was wird hier ausgegeben?

```
public void Foo() {
  try {
    throw new Exception();
  } catch (Exception e) {
    throw new Throwable();
  } catch (Throwable e) {
    System.out.println(„Ha, erwischt!");
  }
  System.out.println(„Und alles wieder normal.");
}
```

Lösung: Throwable wird nicht abgefangen!

Corner Case 2

Was wird hier ausgegeben?

```
public void foo() {
  try {
    try {
      System.out.println(„Langweilig.");
    } finally {
      throw new Exception();
      System.out.println("Ups.");
    }
  } catch (Exception e) {
    System.out.println("Das war knapp!");
  }
}
```

Lösung: Nur „Langweilig“ und „Das war knapp!

Corner Case 3

Was wird hier ausgegeben?

```
public void foo() {
  try {
    try {
      System.out.println("1");
    } finally {
      throw new Exception();
      System.out.println("2");
    }
  } finally {
    System.out.println("3");
  }
}
```

Lösung: 1 und 3… Exception wird aber nicht gefangen!

Beispiele

- Jetzt: Puzzle zusammensetzen → experimentieren mit Exceptions
- Danach solltet ihr Exceptions verstehen
- Wer sich bereits sicher fühlt darf gehen