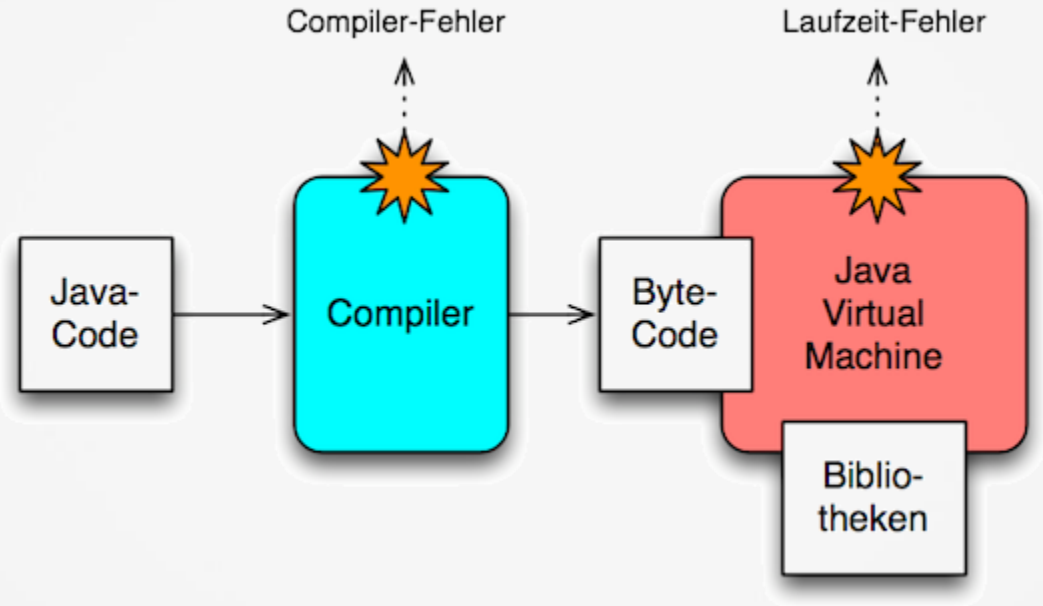
Exceptions

Prinzip Abfangen von Fehlern



Syntax von try-catch-finally

```
try
{
    ……….
}
catch( Exception e)
{
    ……….
}
finally
{
    ……….
}
```

Quellcode, wo Fehler auftreten können

Hier erfolgt die Fehlerbehandlung
catch ist optional, sofern ein finally-Block folgt
Es können mehrere catch-Blöcke hintereinander stehen, um
unterschiedliche Exceptions abzufangen

Wird immer durchlaufen, ist allerdings optional

Beispiel I: Abfangen von Fehlern

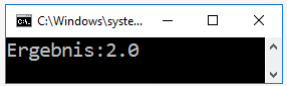
„Erzeugen“ des Fehlers

```
try {
    double result = divide(2,0);
     System.out.println("Ergebnis:"+result);
  } catch (Exception e) {
        System.out.println("Fehler beim Dividieren:"+e.getMessage());
  }
}
public static double divide(int numberToDivide, int numberToDivideBy) {
  return numberToDivide / numberToDivideBy;
}
```

Ausgabe der Fehlermeldung

Programmlauf mit Fehler:

Programmlauf ohne Fehler:



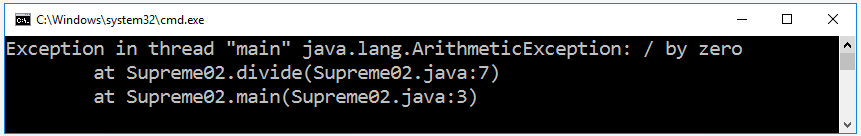
Beispiel II: Weiterleiten an den Aufrufer

Reiche an die Laufzeitumgebung von Java weiter

Reiche an main() weiter

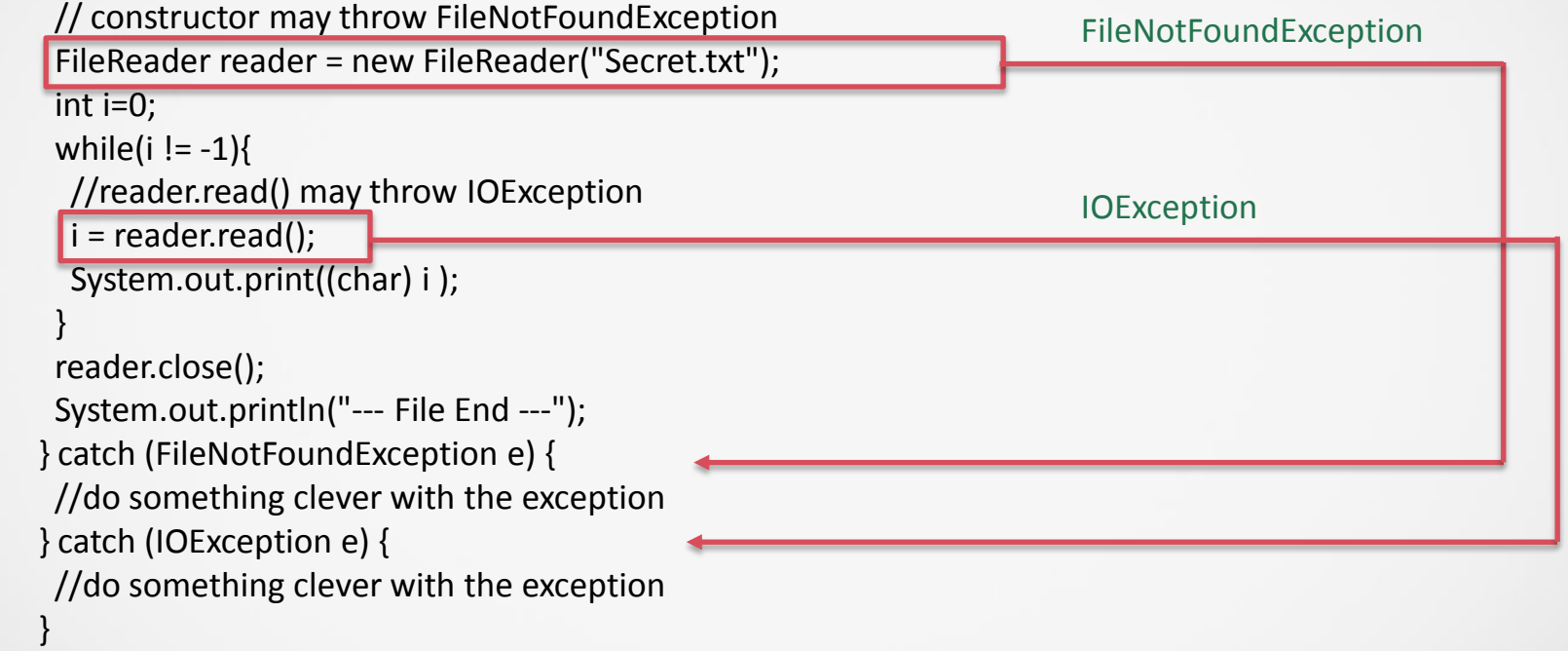
```
public class Supreme02 {
 public static void main(String[] args) throws Exception{
    double result = divide(2,0);
    System.out.print("Ergebnis:"+result);
 }
 public static double divide(int numberToDivide, int numberToDivideBy) throws Exception {
  return numberToDivide / numberToDivideBy;
 }
}
```

Programmlauf mit Fehler:



Beispiel III: Mehrere Catch-Blöcke

```
try {
    // constructor may throw FileNotFoundException
    FileReader reader = new FileReader("Secret.txt");
    int i=0;
    while(i != -1){
      //reader.read() may throw IOException
      i = reader.read();
      System.out.print((char) i );
    }
    reader.close();
    System.out.println("--- File End ---");
  } catch (FileNotFoundException e) {
    //do something clever with the exception
  } catch (IOException e) {
    //do something clever with the exception
  }
```



Wenn die Exception ausgelöst,werden die restlichen Zeilen des try-Blocks nicht mehr ausgeführt

Beispiel IV: Finally & Exceptions in finally

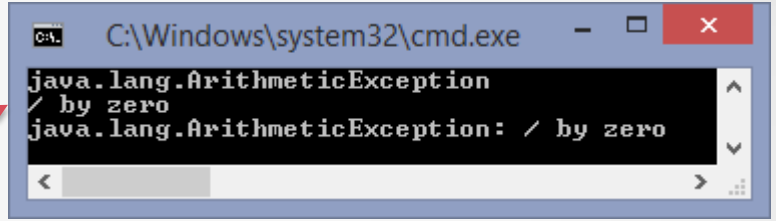
```
FileReader reader = null;
   try {
     reader = new FileReader("someFile");
     int i=0;
     while(i != -1){
       i = reader.read();
       System.out.println((char) i );
     }
   } catch (IOException e) {
     //do something clever with the exception
   } finally {
       if(reader != null){
         try {
           reader.close();
         } catch (IOException e) {
           //do something clever with the exception
         }
       }
}
```

finally ist optional

Exceptions in finally müssen abgefangen werden

Beispiel V: Fehlerdetails

```
public class Supreme05 {
  public static void main(String[] args) {
    try {
      double result = divide(2,0);
      System.out.println("Ergebnis:"+result);
    } catch (Exception e) {
      System.out.println( e.getClass().getName() );
      System.out.println( e.getMessage());
      System.out.println( e.toString() );    }
  }
  public static double divide(int numberToDivide, int numberToDivideBy) {
    return numberToDivide / numberToDivideBy;
  }
}
```



Exceptions - Übersicht