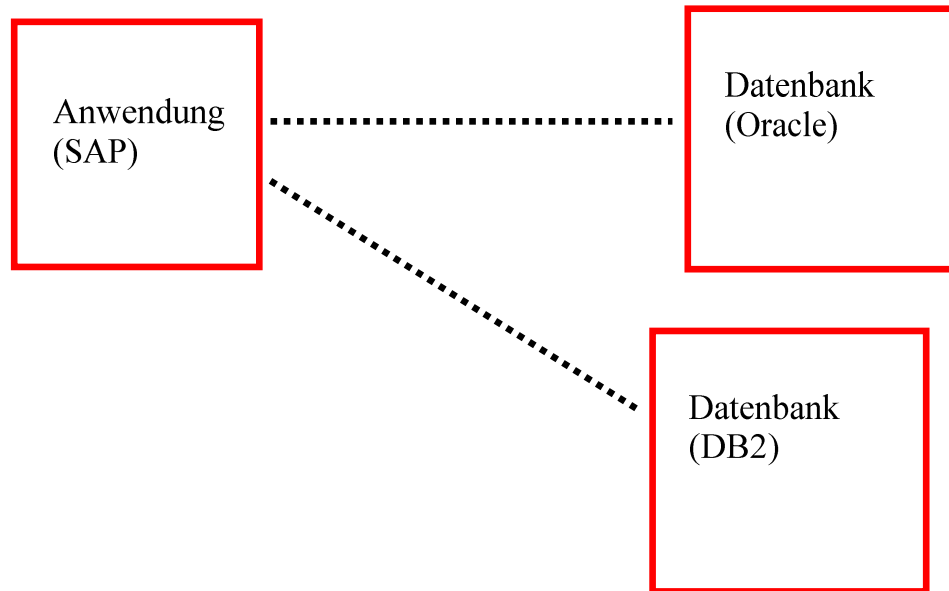


# Nachtrag Interfaces



Es ist eine schwache Kopplung zwischen "SAP" und der Datenbank gewünscht, um die Datenbank austauschen zu können.

# Fehlerbehandlung

Java erzwingt die "Behandlung " von Fehlern.

Es gibt zwei Möglichkeiten, Fehler zu behandeln

## 1. Fehler fangen

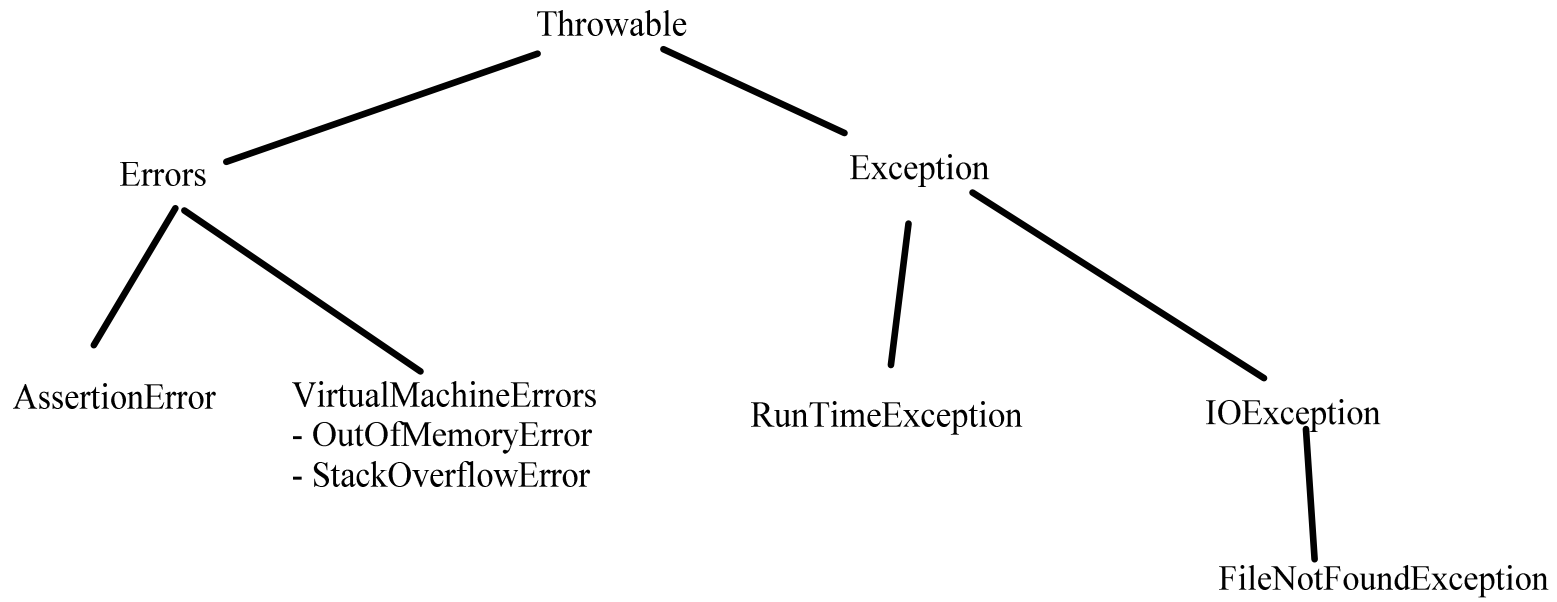
```
try {  
    FileWriter writer = new FileWriter(dateiname);  
} catch (IOException e) {  
    e.printStackTrace();  
}
```

Der Konstruktor von FileWriter wirft einen Fehler (sowohl Konstruktoren als auch Methoden können Fehler werfen)

## 2. Fehler weiterwerfen

```
public void speichern(String text, String dateiname) throws IOException {  
    FileWriter writer = new FileWriter(dateiname);  
}
```

IOException - java.io
Exception - java.lang
Throwable - java.lang



## Merkregeln zur Fehlerbehandlung

1. Errors werden nicht gefangen
2. Fehler so spezifisch wie möglich fangen (z.B. IOException statt Exception)
3. Fehler so nah am Ort des Auftretens wie möglich fangen
4. Niemals einen leeren catch-Block schreiben
5. Bevor eigene Exceptions geschrieben werden, bitte bestehende (200+) prüfen
6. Erben Sie nicht von Error
7. Das Erzeugen von Fehlern ist aufwendig
8. Beachte: Auch in catch-Blöcken können Fehler auftreten

Man kann selbst  
Exception werfen!

```
public String laden(String dateiname) throws ThomasException {  
    if(dateiname == null){  
        ThomasException texception = new ThomasException();  
        texception.setDatum(new Date());  
        throw texception;  
    }  
    return "";  
}
```

## Noch mehr zur Fehlerbehandlung

Ein try kann von mehreren catch-Blöcken gefolgt sein. Der Fehler, der in einem catch-Block gefangen werden soll, darf nicht bereits in einem vorausgegangen catch-Block behandelt worden sein:

```
try {
    datenbank.speichern("Hallo", "c:\\temp\\datei.txt");
} catch (IOException e) {
    e.printStackTrace();
} catch (RuntimeException e){
    e.printStackTrace();
}|
```

← korrekt!

```
try {
    datenbank.speichern("Hallo", "c:\\temp\\datei.txt");
} catch (IOException e) {
    e.printStackTrace();
} catch (FileNotFoundException e){
    e.printStackTrace();
}|
```

← nicht möglich, das IOException alle FileNotFoundExceptions bereits enthält

## Mögliche Kombinationen

- > try-catch
- > try-catch-catch
- > try-catch-finally
- > try-catch-catch-finally
- > try-finally

Finally-Blöcke werden immer ausgeführt, unabhängig davon, ob ein Fehler auftritt oder nicht.

Finally-Blöcke werden beispielsweise eingesetzt, um an Ende einer Methode Ressourcen (z.B. Datenbankverbindung) freizugeben.