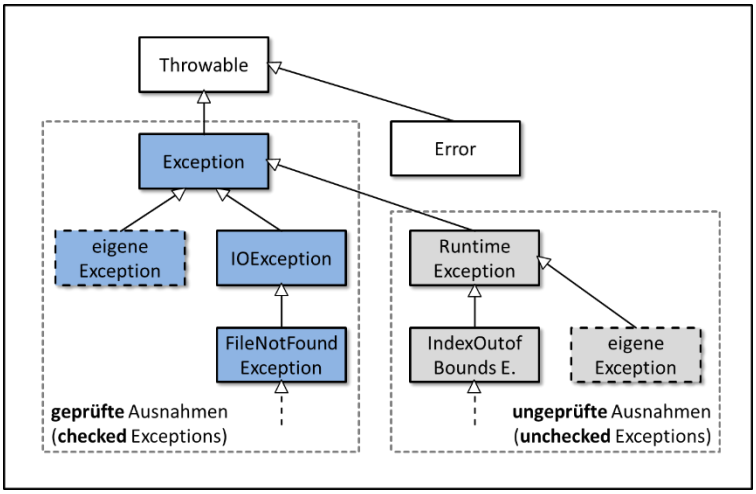

In Java dienen Exceptions dazu, ungewöhnliche oder fehlerhafte Situationen (Ausnahmen) während der Programmausführung kontrolliert zu behandeln, statt dass das Programm unkontrolliert abstürzt. Grundsätzlich wird zwischen zwei Hauptarten unterschieden:

- **Checked Exceptions:** sind Ausnahmen, die bereits zur Übersetzungszeit beachtet werden müssen. Sie treten typischerweise bei vorhersehbaren, aber nicht immer vermeidbaren Situationen auf (z. B. beim Zugriff auf externe Ressourcen). Der Code muss ausdrücklich damit umgehen oder sie weiterreichen.
- **Unchecked Exceptions:** Diese Ausnahmen entstehen meist durch Programmierfehler oder unlogische Zustände zur Laufzeit (z. B. ungültige Operationen). Der Compiler erzwingt hier keinen expliziten Umgang.
- **Errors:** Zusätzlich gibt es schwere Fehlerzustände auf Laufzeitebene (z. B. Ressourcenerschöpfung), die in der Regel nicht sinnvoll im Anwendungscode behandelt werden. Tritt eine Ausnahme ein, so wird das entsprechende Objekt erzeugt.

In Java haben alle Exceptions eine gemeinsame Basisklasse **Throwable** aus dem Paket java.lang.

Java erlaubt die Ausnahmebehandlung mit Hilfe von **try-catch-finally** Programmblöcken die aus drei Teilen bestehen. Alternativ kann eine Ausnahme auch an die aufrufende Instanz zur Behandlung weitergegeben werden.

Syntax	Beschreibung
`try { ... }`	Der **`try`** Block enthält den Code, der eine Exception auslösen („einen Fehler werfen“ 💣) könnte. Es wird "versucht" den Block auszuführen.
`catch( Exception e ) { ... }`	Ein Block **`catch`** wird nur ausgeführt, wenn die Exception e ausgelöst („der Fehler wird gefangen“ ☑). und ggf. behandelt wurde.
`[ finaly { } ]`	Der optionale **`finaly`** Block wird immer ausgeführt. Entweder nach dem regulären Durchlaufen des **`try`**-Blocks oder nach dem **`catch`**-Block.
`... throws Exception ...`	Mit der **`throws`** – Klausel in der Methodensignatur wird gekennzeichnet, dass die Exception nicht bearbeitet, sondern weitergeleitet 🖈 werden.

Aufbau einer Fehlermeldung (z.B.: durch Ausgabe von **`e.printStackTrace()`**)

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "A"
        at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
        at java.lang.Integer.parseInt(Integer.java:580)
        at Fehler.einlesenINT2(Fehler.java:48)
        at Fehler.main(Fehler.java:9)
```

Die erste Zeile zeigt die Fehlermeldung (`java.lang.NumberFormatException`) gefolgt von einer detaillierteren Beschreibung (`For input string: "A"`). Anschließend folgt der so genannte **StackTrace**, der die Aufrufreihenfolge wiedergibt. Hier also `NumberFormatException.forInputString` in der Zeile 65 aufgerufen von `java.lang.Integer.parseInt` in Zeile 580 → `Fehler.einlesenINT2` in Zeile 48 und → `Fehler.main` in Zeile 9.