Java-Versionen und Vergleich zwischen Java 7 und Java 8

Übersicht früherer Java-Versionen

Java 6

- **Release Date:** Dezember 2006
- **Schlüsselmerkmale:**
  - Verbesserte Performance und neue Monitoring-Tools.
  - Einführung des Compiler API (JSR 199).
  - Erweiterte Desktop-APIs, wie das System Tray API.
  - Verbesserte Unterstützung für Web Services und Scripting.

Java 7

- **Release Date:** Juli 2011
- **Schlüsselmerkmale:**
  - **Try-with-resources:** Automatische Schließung von Ressourcen wie Dateien und Datenbanken.
  - **Switch mit Strings:** Nutzung von Strings in `switch`-Anweisungen.
  - **Dynamische Typisierung:** Unterstützung dynamischer Sprachen auf der JVM mit `invokedynamic`.
  - **Diamond Operator (`<>`):** Vereinfachte Generics-Deklarationen.
  - **NIO 2 API:** Verbesserte Dateioperationen und Unterstützung für asynchrone Aufgaben.
  - **Mehrere Ausnahmen in einer `catch`-Anweisung:** Mehrere Ausnahmetypen können in einem Block behandelt werden.

Java 8

- **Release Date:** März 2014
- **Schlüsselmerkmale:**
  - **Lambdas und funktionale Programmierung:** Einführung von Lambda-Ausdrücken für eine kompakte und deklarative Programmierung.
  - **Streams API:** Datenströme können mit Methoden wie `filter`, `map` und `reduce` verarbeitet werden.
  - **Default Methods:** Ermöglicht Standardimplementierungen in Interfaces.
  - **Neue Date and Time API:** Bessere Handhabung von Datums- und Zeitoperationen.
  - **Optional-Klasse:** Umgang mit Null-Werten ohne `NullPointerException`.
  - **Nashorn JavaScript Engine:** Ersetzt Rhino für JavaScript-Unterstützung.
  - **Parallel Streams:** Unterstützung paralleler Datenverarbeitung.
  - **Annotations on Types:** Verbesserung der Typannotationen.

Vergleich zwischen Java 7 und Java 8

Feature	Java 7	Java 8

Feature	Java 7	Java 8
Lambda Expressions	Nicht verfügbar	Unterstützt kompakte und deklarative funktionale Programmierung.
Streams API	Nicht verfügbar	Verarbeitung von Datenströmen mit deklarativen Methoden.
Default Methods	Nicht verfügbar	Standardmethoden in Interfaces für Abwärtskompatibilität.
Try-with-resources	Verfügbar	Auch verfügbar.
Switch mit Strings	Verfügbar	Auch verfügbar.
Annotations on Types	Nicht verfügbar	Verfügbar, um Typannotationen zu verbessern.
Neue Date and Time API	Nicht verfügbar	Verfügbar mit besseren Möglichkeiten für Zeitoperationen.
Optional-Klasse	Nicht verfügbar	Verfügbar zur Vermeidung von Nullwert-Problemen.

Fazit

Java 7 brachte wesentliche Verbesserungen in Bezug auf die Sprachkonstrukte und APIs, während Java 8 einen Paradigmenwechsel hin zur funktionalen Programmierung und modernen APIs einleitete. Java 8 wird als eine der bedeutendsten Versionen in der Java-Geschichte betrachtet.