Neues in Java 5

- Collections
  - In früheren Versionen von Java gab es oft bei der Verwendung von Collections Probleme:
    - Collections sind typunsicher, da Referenzen in einer Collection vom Typ **`object`** sind.
    - Elemente müssen explizit zum gewünschten Typ typecasted werden.
    - Variablen mit elementaren Datentypen können nicht ohne sog. „Boxing“ in Collections eingefügt werden.
    - Auch umgekehrt konnte Variablen mit elementaren Datentypen nicht direkt Elemente aus einer Collection zugewiesen werden.

Neues in Java 5

- Boxing
  - In Java wird zwischen Klassen und elementaren (primitiven) Datentypen unterschieden.
  - Um elementare Datentypen in Programmier-situationen die Klassen erfordern verwenden zu können, waren bisher manuelle Konvertierungen in Referenzklassen erforderlich (Boxing).
    - Beispiel: Konvertierung Datentyp **`int`** → Klasse **`Integer`**
  - Die entsprechenden Referenzklassen bieten keinerlei arithmetische Methoden, sie sind daher sehr unflexibel.

Neues in Java 5

- Boxing (bisher in Java <5)

Quelltext

```
Collection collection = new ArrayList();
int zahl = 5;
Collection.add(zahl);           // Fehler, int ist
                                // inkompatibel zu object.

Collection.add(new Integer(zahl));          // Korrekt
```

```
zahl = collection.elementAt(0); // Fehler, object ist
                                // inkompatibel zu int.

zahl = ((Integer)collection.elementAt(0)).intValue();
// Korrekt
```


Neues in Java 5

- Generics und Autoboxing
  - Ein Rückgriff auf die Templates aus C++ (in Java sog. „Generics“) ermöglicht es, eine Collection explizit auf einen bestimmten Typ zu beschränken.
  - Boxing, d.h. die Konvertierung von Datentyp zu Klasse wird nun automatisch durchgeführt.
  - Es erleichtert den Umgang mit elementaren Datentypen im Kontext mit Referenztypen.
  - Es gibt weniger redundante Programmierarbeit.
  - Dies führt zu wesentlich übersichtlicherem und kürzerem Programmcode. (Siehe nachfolgendes Beispiel)

Neues in Java 5

- **for**
  - Um die Arbeit mit Collections und Arrays weiter zu vereinfachen, wurde eine weitere Implementation der **for**-Schleife implementiert.
  - Eine Kurzschreibweise ermöglicht es, automatisch alle Elemente einer Collection oder eines Arrays zu durchlaufen, ein Überlauf ist nicht möglich.
  - Dieses Konstrukt ist aus anderen Programmier-sprachen unter dem Namen „**foreach**“ bekannt.

Neues in Java 5

- Gegenüberstellung

Quelltext (Java <5)

```
Collection collection = new ArrayList();
int[] werte = { 5, 13, 12, 40, 30 };
                  // Werte werden in Collection eingefügt.
for (int i = 0; i < werte.length; i++)
   collection.add(new Integer(werte[i]));
                  // Summe aus Elementen der Collection
                  // wird gebildet und ausgegeben.
int summe = 0;
for (Iterator it = collection.iterator(); it.hasNext();)
   summe += ((Integer)it.next()).intValue();
System.out.println(summe);
```


Neues in Java 5

- Gegenüberstellung

Quelltext (Java 5)

```
Collection<Integer> collection = new ArrayList<Integer>();
int[] werte = { 5, 13, 12, 40, 30 };
                  // Werte werden in Collection eingefügt.
for (int wert : werte)
   collection.add(wert);
                  // Summe aus Elementen der Collection
                  // wird gebildet und ausgegeben.
int summe = 0;
for (int wert : collection)
   summe += wert;
System.out.println(summe);
```


Neues in Java 5

- Risiken beim Autoboxing
  - Ein **Integer** kann „**null**“ sein, ein **int** nicht.
    - Wird an dieser Stelle ein Autoboxing durchgeführt, wird eine Exception ausgelöst.

Quelltext

```
Integer i1 = null;
int i2 = i1;  // Erzeugt Exception, keinen Compilerfehler!
```

  - Eine automatische Typerweiterung findet nicht statt.
    - Datentypen werden erweitert, Referenzklassen nicht.

Quelltext

```
long l = 5;  // funktioniert, 5 wird zu 5L erweitert
Long l = 5;  // Fehler: 5=>new Integer(5), Integer != Long
Long l = 5L; // funktioniert
```

Neues in Java 5

- Umgang mit Referenzklassen
  - Der **==**-Operator überprüft bei Objekten die Referenzgleichheit, nicht die Wertegleichheit.
  - Um dieses Problem zu umgehen, existieren für die Zahlen -128 bis +127 eindeutige Referenzen in der Java VM.

Quelltext

```
Integer i1 = 127;  Integer i2 = 127;
System.out.println(i1 == i2);  // Ausgabe: true

i1 = 128;  i2 = 128;
System.out.println(i1 == i2);  // Ausgabe: false
```


Neues in Java 5

- Ausblick
  - Es ist geplant, diesen Bereich auf Systemen mit großem Speicher auf -32768 bis +32767 zu erweitern.
  - Dies bedeutet einen wesentlichen Eingriff in ein Grundprinzip von Java: Die Plattformunabhängigkeit, egal ob das Programm auf einem Handy oder einem Server läuft.

Neues in Java 5

- Fazit
  - Sowohl die Vorteile, als auch die Nachteile sind gravierend.
  - Java 5 ist noch zu neu, um jetzt schon ein endgültiges Urteil darüber fällen zu können, ob die Vorteile die Nachteile aufwiegen.
  - Bis dahin muss man sich beiden bewusst sein, und seine Entscheidung nach Problemstellung und nach persönlichem Geschmack fällen.