JAVA

=====

- in Java sind alle Klassen direkt oder indirekt von der Klasse Object abgeleitet

- die Klasse Object bietet drei wichtige Methoden an

die ggf. in der eigenen Klasse überschrieben werden muss

Methoden: Typ Ergebniss

toString() : String liefert eine Zeichenkettenrepräsentation d, Objects

hashCode(): int liefert eine eindeutige Objekt-ID

equals(): Boolean liefert eine Aussage darüber, ob zwei Objecte gleich sind

AUTOBOXING: Alles ist ein Objekt, aber nur dann wenn es unbedingt sein muss

Int x 🡪 int x .toString 🡪 int x

Integer

* Falls Sie “individuelle” Ausgaben für Objekte Ihrer Klassen wünschen, dann überschreiben Sie die

Methode toString(): String

@Override

**public** String toString() {

**return** "" + **this**.meinWert; oder String.valueOf(**this**.meinWert);

}

* Die Vergleiche auf Gleichheit führen bei primitive Datentypen immer zu den “richtigen” Egebnisse
* Die Vergleiche auf Gleichheit für Objekte müssen differenziert betrachtet werden

== vergleicht die Objectreferenz

equals() vergleicht je nach Implementierung

Wie muss equals() implementiert werden? „equals contract“

1. Jedes Objekt liefert im Verglich mit sich selbst TRUE (Reflexivität)
2. Wenn x „==“ y TRUE ist, dann muss auch y „==“ x TRUE sein (Symnetrie)
3. Wenn x „==“ y TRUE ist und y „==“ z TRUE dann muss auch x „==“ z TRUE sein (Transitivtät)
4. X „==“ y muss auch bei wiederholter Ausführung TRUE/FALSE sein (Konsistenz)
5. Für alle x die verschieden sind von (ǂ)null 🡪 x“==“ null muss FALSE sein

* Der hashCode() Contract:

1. Egal, wie oft man hashCode() aufruft, die Methode liefert IMMER den gleich Wert (Konsistenz)
2. !!! WER EQUALS ÜBERSCHREIBT \*\*\*MUSS\*\*\* AUCH HASHCODE ÜBERSCHREIBEN if(x.equals(y))🡪x.hashcode==y.hashcode
3. Ungleiche Objekte **sollen** verschiedene HashCodes haben müssen aber nicht