PM2 Aufgabenblatt A07

Youssef Benlemlih, Jonas Krukenberg 18.05.2020

Java Denksportaufgaben

1. Die ==-Relation in java.

Prüfung von `==` auf Äquivalenzeigenschaften:
Sei `x = (double) PI`, `y = (int) 3` und `z = (float) PI`

- **reflexiv**
  `x == x`, `y == y`, `z == z`, da primitive Typen gleichen Werts sogar die selbe Objektidentität haben.
- **symmetrisch**
  Sei `x == y`. Dann ist auch `y == x`, da auch `(int) 3 == (int) (double) PI` ist.
- **transitiv**
  Sei `x == y` und `y == z`. Dann ist auch `x == z`, also `(double) PI == (float) PI`. Das ist ein Widerspruch, da die Repräsentation von PI im double-Format mit 64 Bit eine doppelt so große Genauigkeit und somit auch mehr Nachkommastellen als `(float) PI` besitzt.
  Somit ist die `==`-Relation für primitive Datentypen **keine Äquivalenzrelation**.

2. Field Hiding

2.1 Der Code ist **nicht** kompilierbar.
Der Anwender würde vermutlich erwarten, dass "Derived" ausgegeben wird.

2.2 Dadurch dass eine Instanzvariable `private className` in der Unterklasse `Derived` den gleichen Namen hat, wird `public className` in der Superklasse `Base` "versteckt" und dadurch ist className dann private.
Der Code verliert durch ein solches "Field Hiding" deutlich an Übersichtlichtlichkeit.
Besser wäre es, den Zugriff auf Attribute über getter-Methoden zu realisieren, also `public String getClassName()` in `Base` zu schreiben, wobei nur `Base` das Attribut `private className` hat, welches im Defaultkonstruktor auf "Base" gesetzt wird bzw. in einem anderen Konstruktor per Parameter gesetzt werden kann. In `Derived` würde man dann im Defaultkonstruktor `super("Derived")` aufrufen. Möchte man das Lesen von `Derived.className` verhindern, könnte man dies durch Überschreiben von `getClassName()` in `Derived` regeln.

Quellen
Hiding Fields