Komponenten durch Ableiten: Reflection

Woher kennt der GUI-Builder die Property MouthWidth?

Reflection

- **Die Klasse Class liefert Methoden, um eine Klasse zu inspizieren**
  - → Obj.getClass() liefert Class-Objekt cl
  - → cl.getMethods liefert z.B. Feld mit Methoden
- **Klassen in java.lang.reflect für Methoden (und mehr)**
- **Wenn es Methoden getXXX und setXXX gibt, dann erzeuge eine Property „XXX“**

Komponenten durch Ableiten: Reflection

RWTH AACHEN UNIVERSITY

Übung: Schreiben einer Klasse mit statischen Methoden

- **public static void printAllMethods(Class<?> cl)**

  → cl.getMethods()

- **public static void printDeclaredMethods(Class<?> cl)**

  → cl.getDeclaredMethods()

- **public static void printHierarchy(Class<?> cl)**

  → cl.getSuperclass()

- **Main-Methode, die einen Klassennamen als Parameter nimmt.**

```
public static void main(String[] args) {
        Class<?> c = Class.forName(args[0]);
        printAllMethods(c);
        printDeclaredMethods(c);
        printHierarchy(c);
}
```

Komponenten durch Ableiten: Reflection

- **Aufrufen einer Methode einer Klasse, wo nur die Namen als Strings gegeben sind.**

```
package de.rz;
public class TestReflect {
        private int id = 0;
        public TestReflect() {
        }
        public TestReflect(int newId) {
                id = newId;
        }
        public void doIt(String str) {
                System.out.println("TestReflect(" + id + ").doit(" + str + ")");
        }
}
```

Komponenten durch Ableiten: Reflection

- **Konstruieren eines Objekts mit dem Standard-Konstruktor und Aufruf der Methode doIt mit „Test“**
  - → Als Input nur „de.rz.TestReflect“ und „doIt“.
- **Konstruieren eines Objekts mit dem Konstruktor, der ein int als Parameter nimmt, und Aufruf der Methode doIt mit „Test“**