Faculty Civil Engineering
Chair of Intelligent Technical Design
Prof. Dr.-Ing. Christian Koch

Bauhaus-Universität Weimar

Object-oriented Modeling and Programming in Engineering (OOMPE)

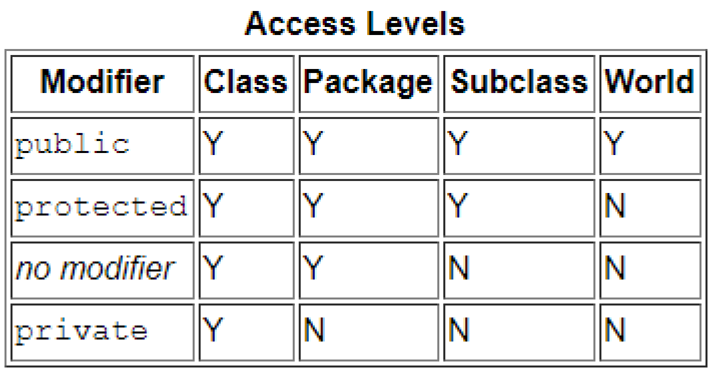
Winter semester 2018-19

07 – Inheritance

Encapsulation, inheritance and polymorphism

Bauhaus-Universität Weimar

- Encapsulation
  - Hide implementation (knowledge is money)



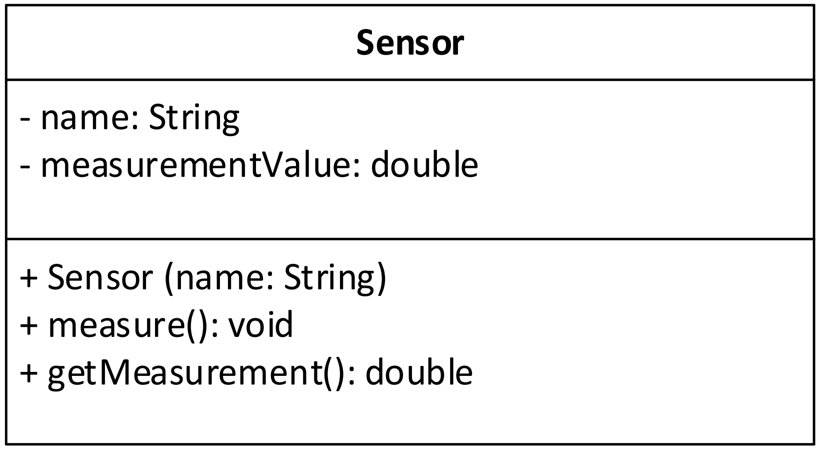
Access Levels

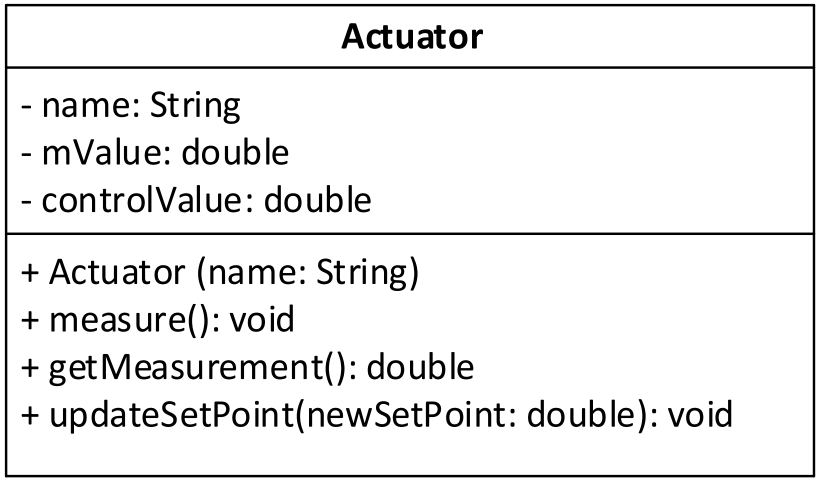
Modifier	Class	Package	Subclass	World
`public`	Y	Y	Y	Y
`protected`	Y	Y	Y	N
no modifier	Y	Y	N	N
`private`	Y	N	N	N

  - Most restrictive as possible and usage of getters and setters
- Inheritance
  - Build up classification
  - DRY concept (Don't repeat yourself)
- Polymorphism
  - Specialized implementations
  - Allows more dynamic within code vs. readability

Sensors and actuators

- Think about a sensor and an actuator
- Both can measure (mybe in different ways)
- The Actuator has the possibility tho control something




Actuator
- name: String
- mValue: double
- controlValue: double
+ Actuator (name: String)
+ measure(): void
+ getMeasurement(): double
+ updateSetPoint(newSetPoint: double): void

- Find a better way to implement it
  - Draw the UML class diagram
  - Implement it

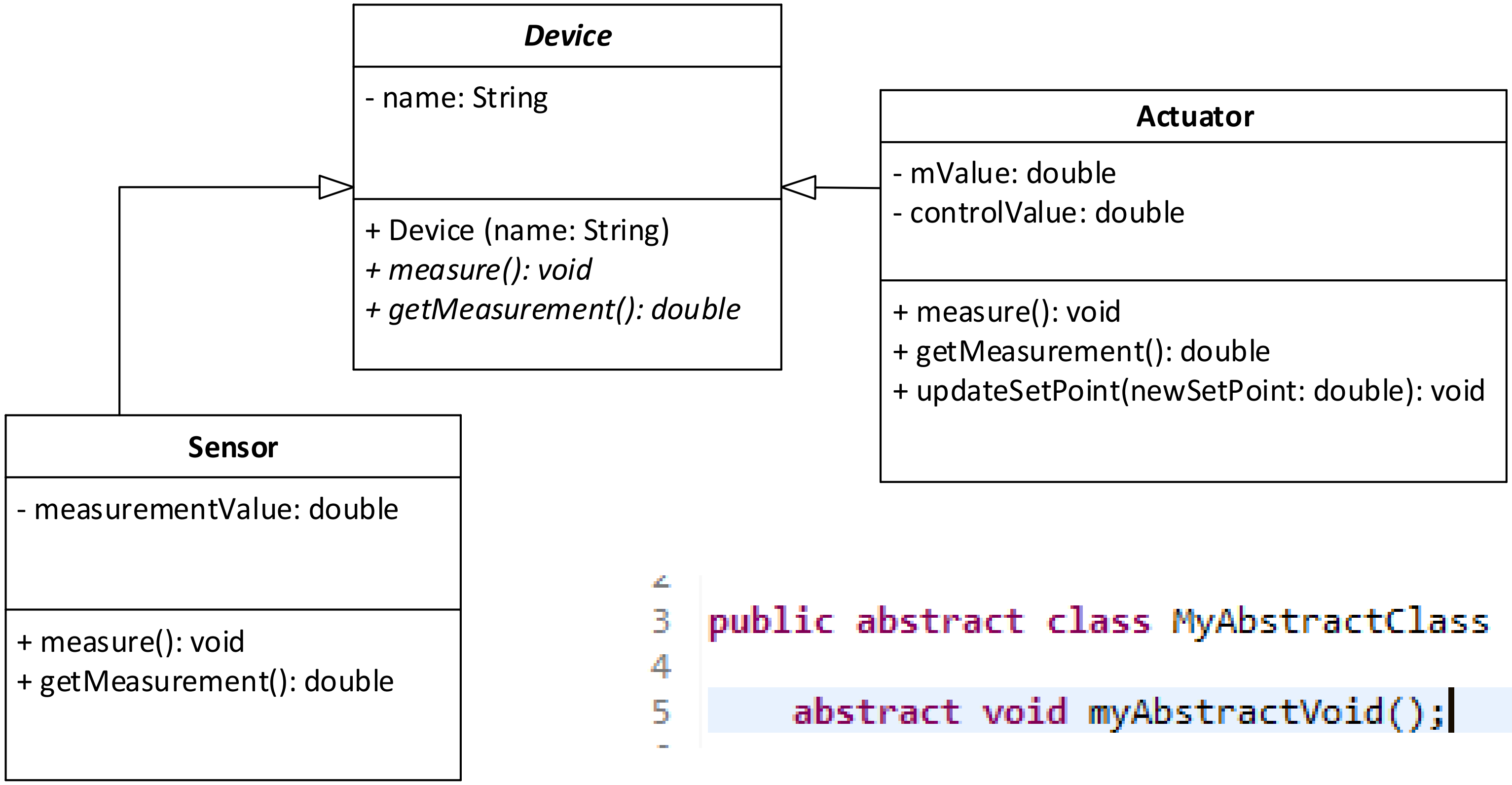
Abstract classes and interfaces

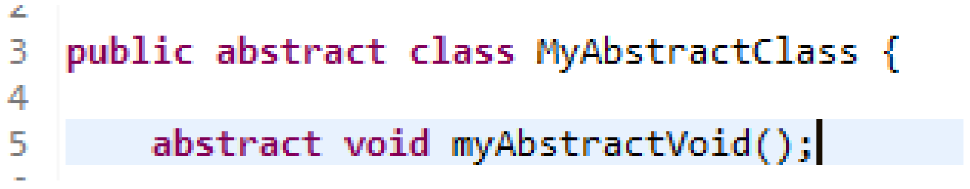
Bauhaus-Universität Weimar

- Abstract classes provide method definitions without implementation
- Interfaces nearly the same: They say: ‚There is a method with that signature‘ but don‘t tell something about the implementation
- The differences are:
  - An Abstract class can contain attributes
  - One class can implement several intefaces but have only one super class

Implementing abstract classes

- For an abstract class



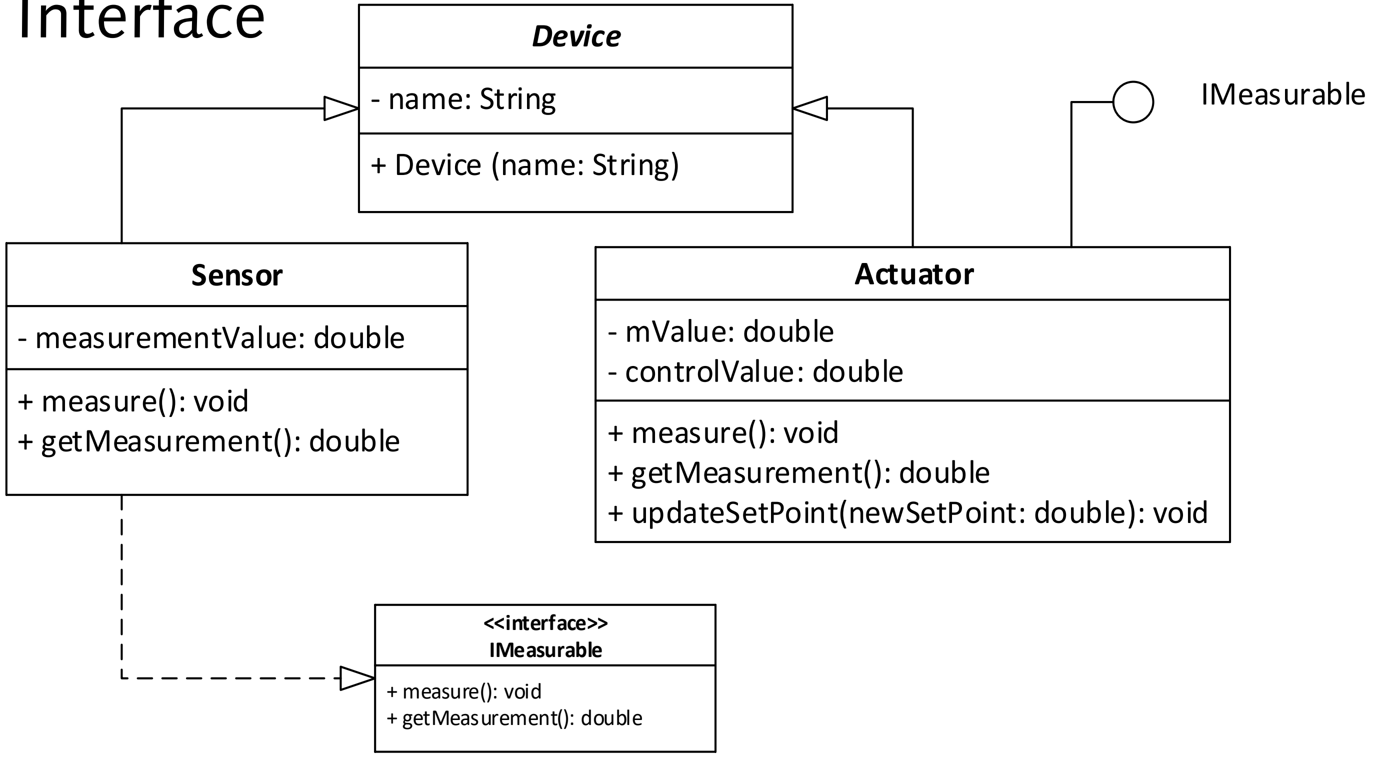


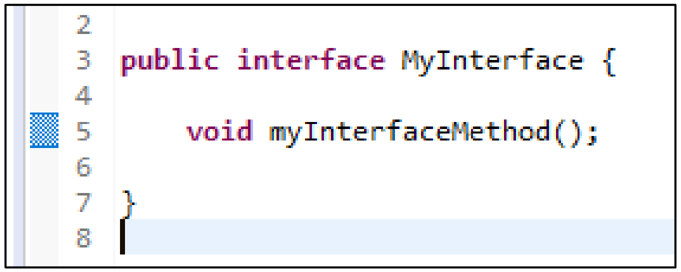
```
public abstract class MyAbstractClass {

    abstract void myAbstractVoid();
```

Implementation of Interfaces

- For an Interface





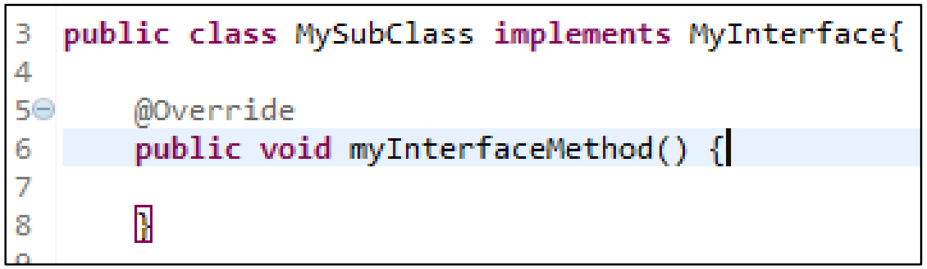
```java

public interface MyInterface {

    void myInterfaceMethod();

}

```



```java
public class MySubClass implements MyInterface{

    @Override
    public void myInterfaceMethod() {

    }
```

Alternatives

- Implement the sensor-actuator problem with an abstract class
- Use interfaces to implement the sensor-actuator problem
- Use a combination of abstract class and interface for the problem (e.g. let the abstract class implement an interface, whereas the interface methods in the abstract class are abstract