

Object-Oriented Programming

3. Lecture
LS 2025/2026
Author: Juraj Petrík

Methods

- Java is **pass by value (copy)**
- Any number and type of parameters
- We return max one "object"

Access modifiers

- Classes
 - Public – everywhere
 - *Default* – only from the same package
- Attributes, methods, constructors
 - Public – accessible to all classes
 - Private – only in own class
 - *Default* – only from the same package
 - Protected – only from the same package and subclasses

Other modifiers

- **Classes**
 - Final – inheritance not possible
 - Abstract – objects cannot be created
- **Attributes and methods**
 - Final – cannot be overridden or modified
 - Static – belong to the class, not the object
 - Abstract – only in abstract classes, only for methods, "enforced" override
 - Transient – omitted during serialization
 - Synchronized – "lock" for threads (mutual exclusion)
 - Volatile – not cached in threads

Static

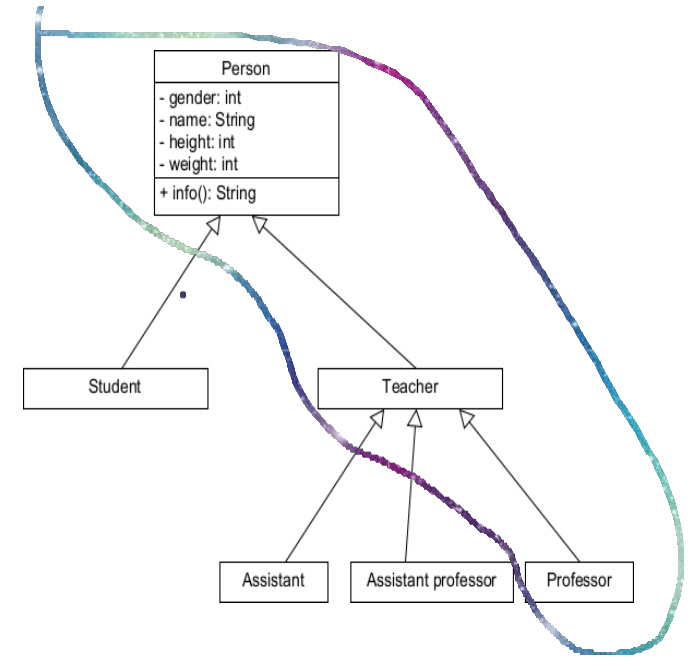
- Method – e.g., helper methods, only static content is accessed in the class
 - Block – executed when the class is "loaded" for the first time
 - Attribute – or otherwise class variable
 - *Class (only nested)*
-
- If you have a lot of "static" code, you are probably not programming object-oriented

Casting primitive variables

- Widening
- Narrowing
- <https://docs.oracle.com/javase/specs/jls/se21/html/jls-5.html>

Casting ~~Objects~~ References

- Upcasting (implicitly) - automatically
- Downcasting (explicitly)
- Only in the relevant subtree



- Instanceof – “test if an object is an instance of a class, an instance of a subclass, or an instance of a class that implements a particular interface”

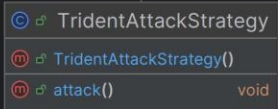
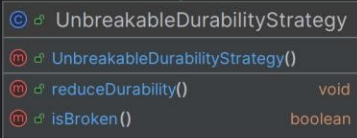
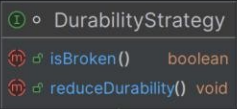
Java API

- So we don't have to reinvent the wheel over and over again
- <https://docs.oracle.com/en/java/javase/21/docs/api/index.html>
- `System.out.println()`

- Let's take a look at our weapon system

Strategy

- We define ways to do something; these ways are interchangeable
- Identify the aspects of your application that vary and separate them from what stays the same.
- Program to an interface, not an implementation
- Favor composition over inheritance



Styles and patterns

- **"Someone has already solved your problems"**
 - Architectural style (microservices)
 - Architectural pattern (client-server)
 - Design pattern (singleton)
-
- Gang of Four: Erich Gamma, Richard Helm, John Vlissides, Ralph Johnson, 23 classic patterns

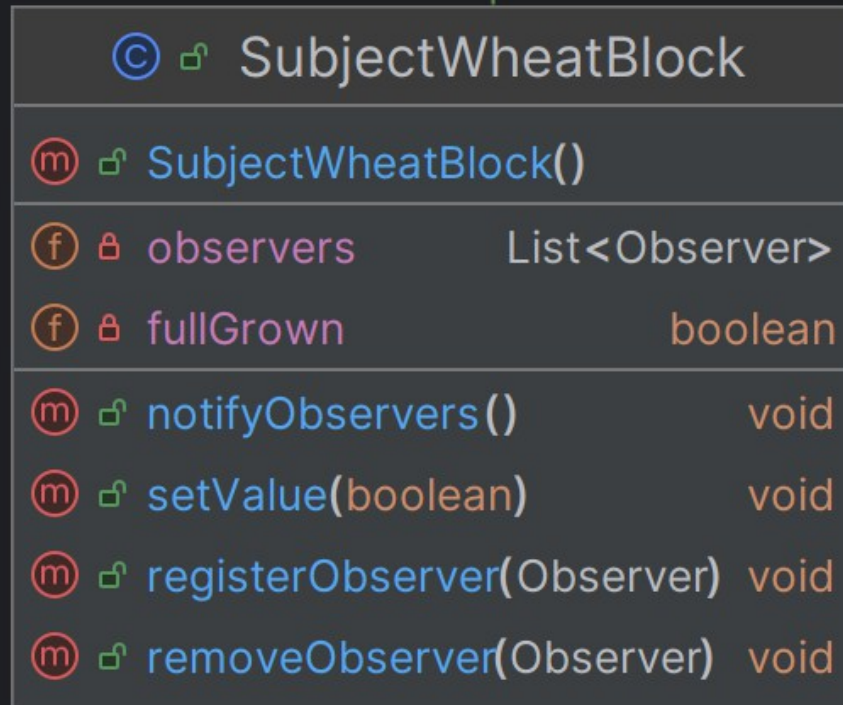
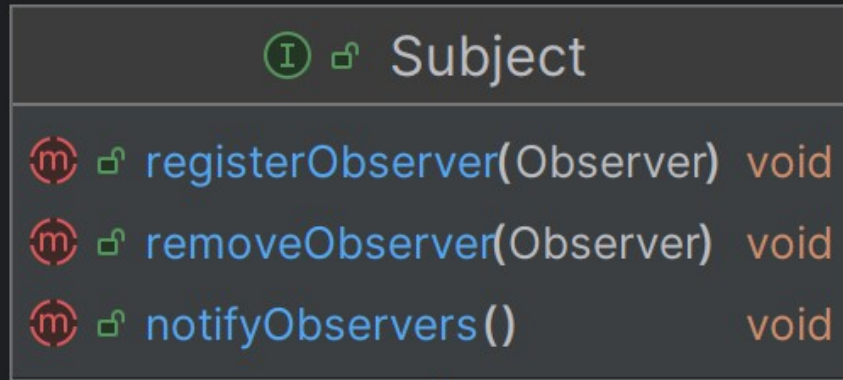
Design patterns

- Pattern catalogs
- Creational – creating objects
- Structural – composing objects and classes
- Behavioral – behavior and "responsibility" of objects

Observer

- Mechanism for notifying other objects of changes
- Loose coupling
- GUI – listeners





Quiz time