Objektovo Orientované programovanie

3. Prednáška
LS 2024/2025
Juraj Petrík

Metódy

- Java je **pass by value (copy)**
- Ľubovoľný počet a typ parametrov
- Vraciame max jeden „objekt“

Modifikátory prístupu

- Triedy
  - Public – odvšadiaľ
  - *Default* – iba z rovnakého balíku
- Atribúty, metódy, konštruktory
  - Public – prístupné pre všetky triedy
  - Private – iba vo vlastnej triede
  - *Default* – iba z rovnakého balíku
  - Protected – iba z rovnakého balíku a podtried

Ostatné modifikátory

- Triedy
  - Final – nie je možné dedenie
  - Abstract – nie je možné vytvárať objekty
- Atribúty a metódy
  - Final – nemôžu byť prekonané, modifikované
  - Static – patria k triede, nie objektu
  - Abstract – iba v abstraktných triedach, iba pri metódach, „vynútenie“ prekonania
  - Transient – vynechajú sa pri serializácii
  - Synchronized – „lock“ pri vláknach (mutual exclusion)
  - Volatile – nie je cachované pri vláknach

Static

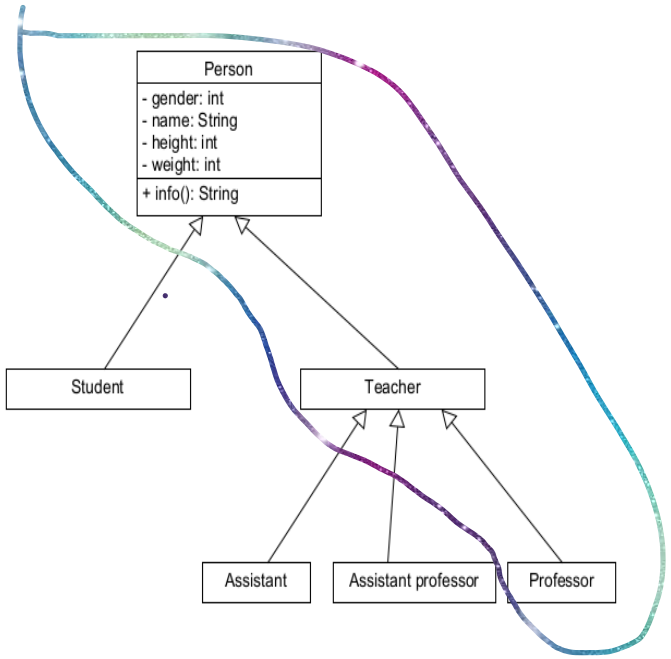
- Metóda – napr. helper methods, v triede sa dostane iba ku statickému obsahu
- Blok – vykoná sa keď sa prvý krát „načíta” trieda
- Atribút – alebo inak premenná triedy
- *Trieda (iba nested)*

- Pokiaľ veľa kódu máte “statického”, tak pravdepodobne neprogramujete objektovo

Casting primitive variables

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

Casting ~~Objects~~ References



- Upcasting (implicitne) - automaticky
- Downcasting (explicitne)
- Iba v príslušnom podstrome

- 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”

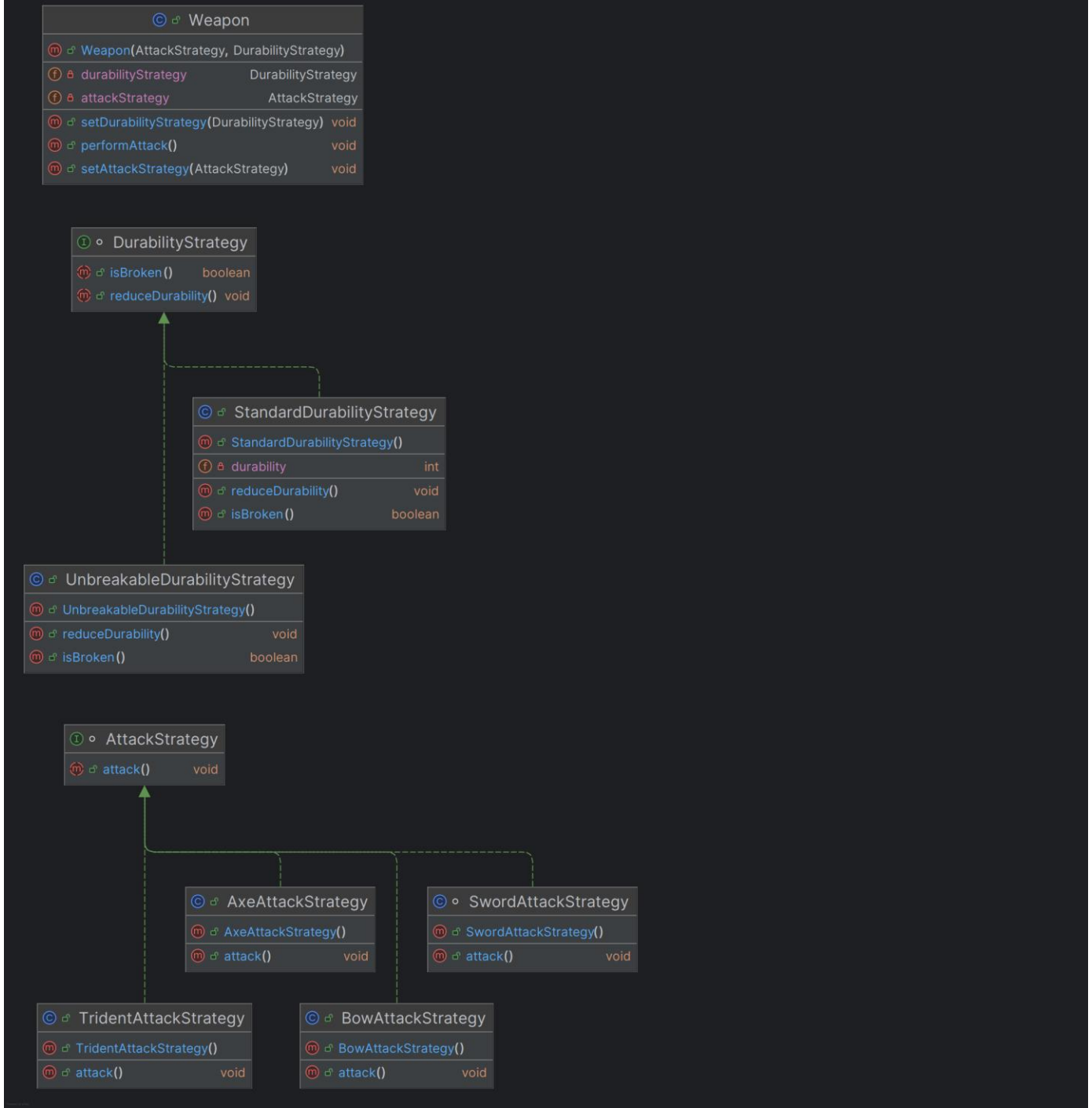
Java API

- Aby sme nevynašli koleso zas a znova
- https://docs.oracle.com/en/java/javase/21/docs/api/index.html
- System.out.println()

- Poďme sa pozrieť na náš systém zbraní

Strategy

- Definujeme spôsoby ako niečo spraviť, tieto spôsoby sú medzi sebou zameniteľné
- 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


Weapon
Weapon(AttackStrategy, DurabilityStrategy)
durabilityStrategy DurabilityStrategy
attackStrategy AttackStrategy
setDurabilityStrategy(DurabilityStrategy) void
performAttack() void
setAttackStrategy(AttackStrategy) void
DurabilityStrategy
isBroken() boolean
reduceDurability() void
StandardDurabilityStrategy
StandardDurabilityStrategy()
durability int
reduceDurability() void
isBroken() boolean
UnbreakableDurabilityStrategy
UnbreakableDurabilityStrategy()
reduceDurability() void
isBroken() boolean
AttackStrategy
attack() void
AxeAttackStrategy
AxeAttackStrategy()
attack() void
SwordAttackStrategy
SwordAttackStrategy()
attack() void
TridentAttackStrategy
TridentAttackStrategy()
attack() void
BowAttackStrategy
BowAttackStrategy()
attack() void

Štýly a vzory

- **„Niekto už vyriešil Vaše problémy“**
- Architektonický štýl (microservices)
- Architektonický vzor (client-server)
- Návrhový vzor (singleton)

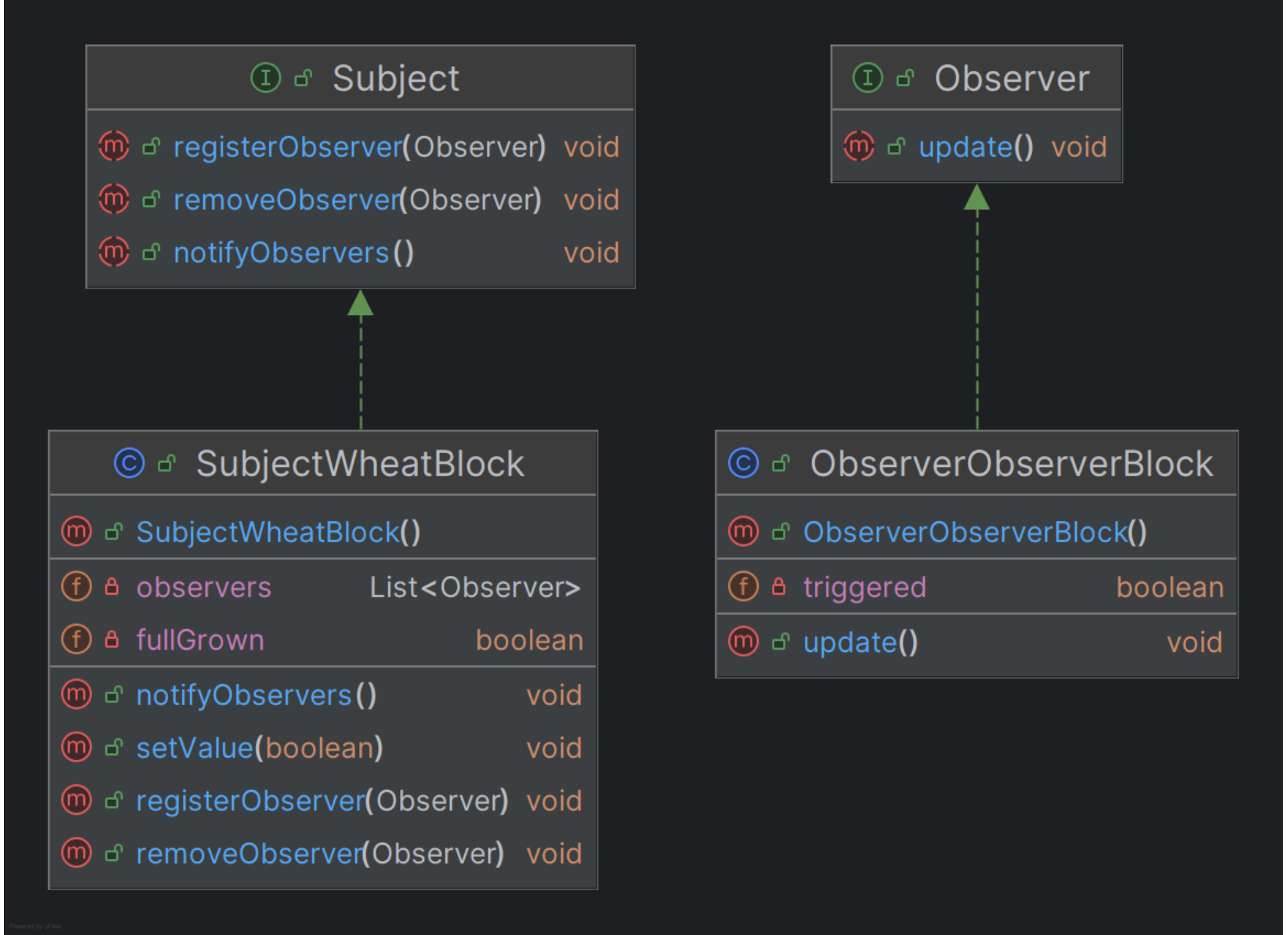
- Gang of Four: Erich Gamma, Richard Helm, John Vlissides, Ralph Johnson, 23 klasických vzorov

Návrhové vzory

- Katalógy vzorov
- Creational – vytváranie objektov
- Structural – skladanie objektov a tried
- Behavioral – správanie a „zodpovednost“ objektov

Observer

- Mechanizmus notifikovania o zmene iným objektom
- Loose coupling
- GUI – listeners


Subject
registerObserver(Observer) void
removeObserver(Observer) void
notifyObservers() void
Observer
update() void
SubjectWheatBlock
SubjectWheatBlock()
observers List<Observer>
fullGrown boolean
notifyObservers() void
setValue(boolean) void
registerObserver(Observer) void
removeObserver(Observer) void
ObserverObserverBlock
ObserverObserverBlock()
triggered boolean
update() void

Quiz time