Programovanie v jazyku Python

Návrhové vzory I
prednáška 10

Katedra kybernetiky a umelej inteligencie
Technická univerzita v Košiciach
Ing. Ján Magyar, PhD.

Návrhový vzor

- všeobecné znovupoužiteľné riešenie pre opakujúce sa problémy v softvérovom inžinierstve
- popis riešenia, nie samotné riešenie
- formalizovaný best practice
- nie sú nevyhnutné, ale zjednodušujú implementáciu

Štruktúra návrhového vzoru

- definuje komponenty a ich rolu
- definuje vzťah medzi komponentmi
- nešpecifikuje funkcionalitu (závisí od prípadu použitia)
- nešpecifikuje implementáciu (je to na programátorovi)

Typy návrhových vzorov

- kreačné - ako vytvoriť objekt?
- štrukturálne - ako realizovať vzťah medzi objektmi?
- behaviorálne - ako môžu komponenty komunikovať?
- konkurenčnosť - pre viacvláknové programy

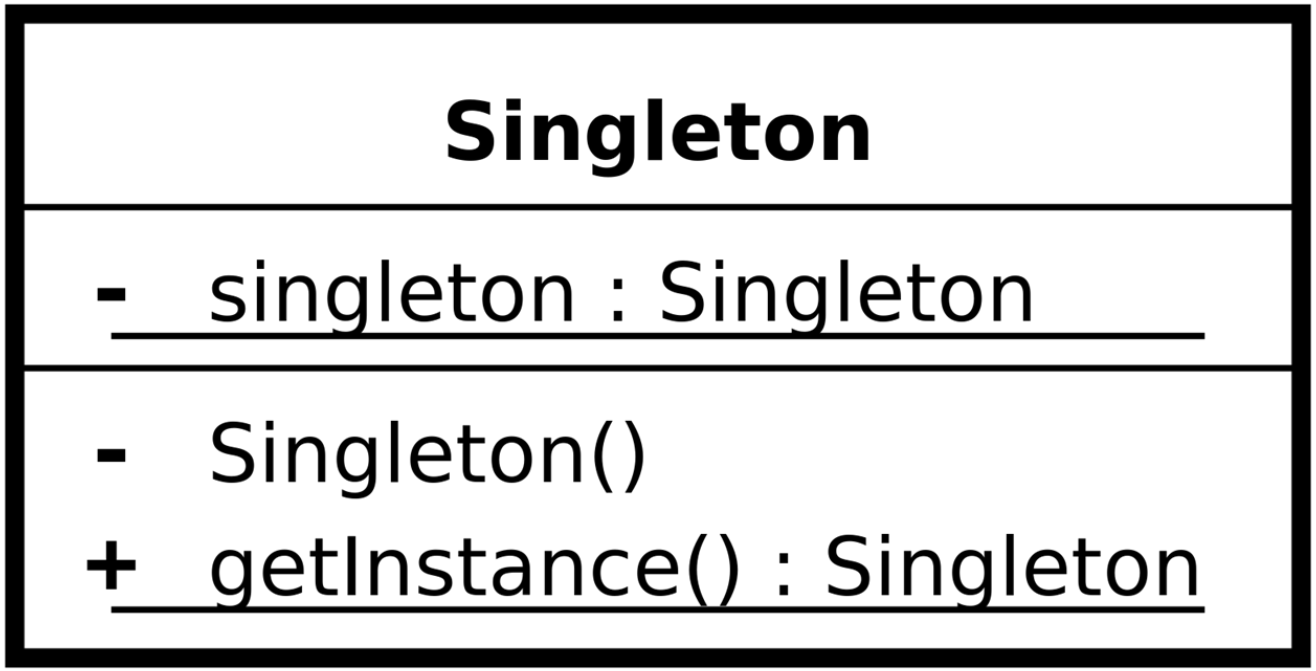
Kreačné návrhové vzory

- singleton
- abstract factory
- factory method
- builder
- prototype

Singleton

- zabezpečuje, že trieda má iba jednu inštanciu (alebo žiadnu)
- poskytuje prístup k jedinej inštancii
- môžeme kontrolovať vytvorenie inštancie
- možná lazy initialization
- globálny stav

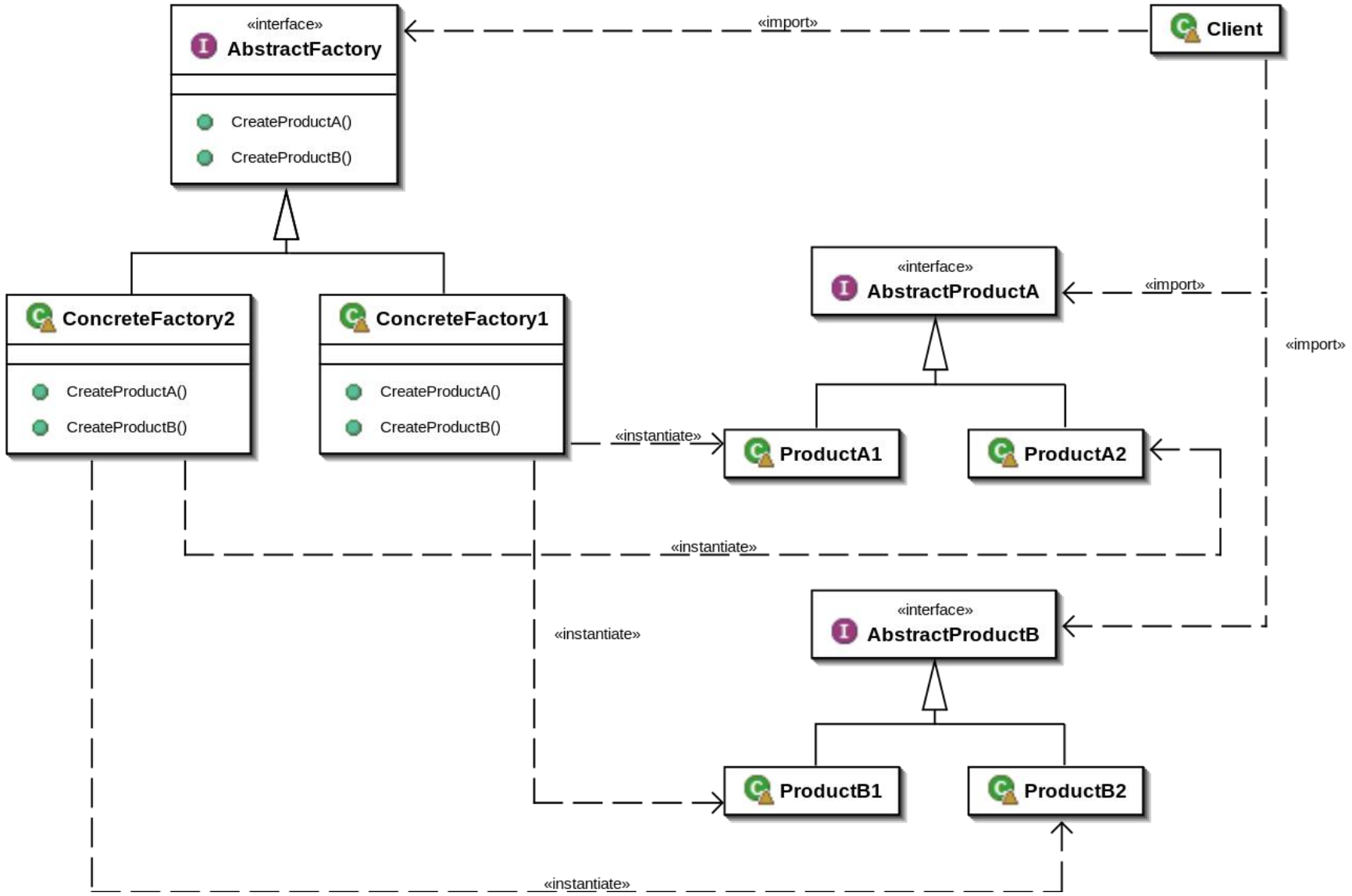
Singleton


Singleton
- singleton : Singleton
- Singleton()
+ getInstance() : Singleton

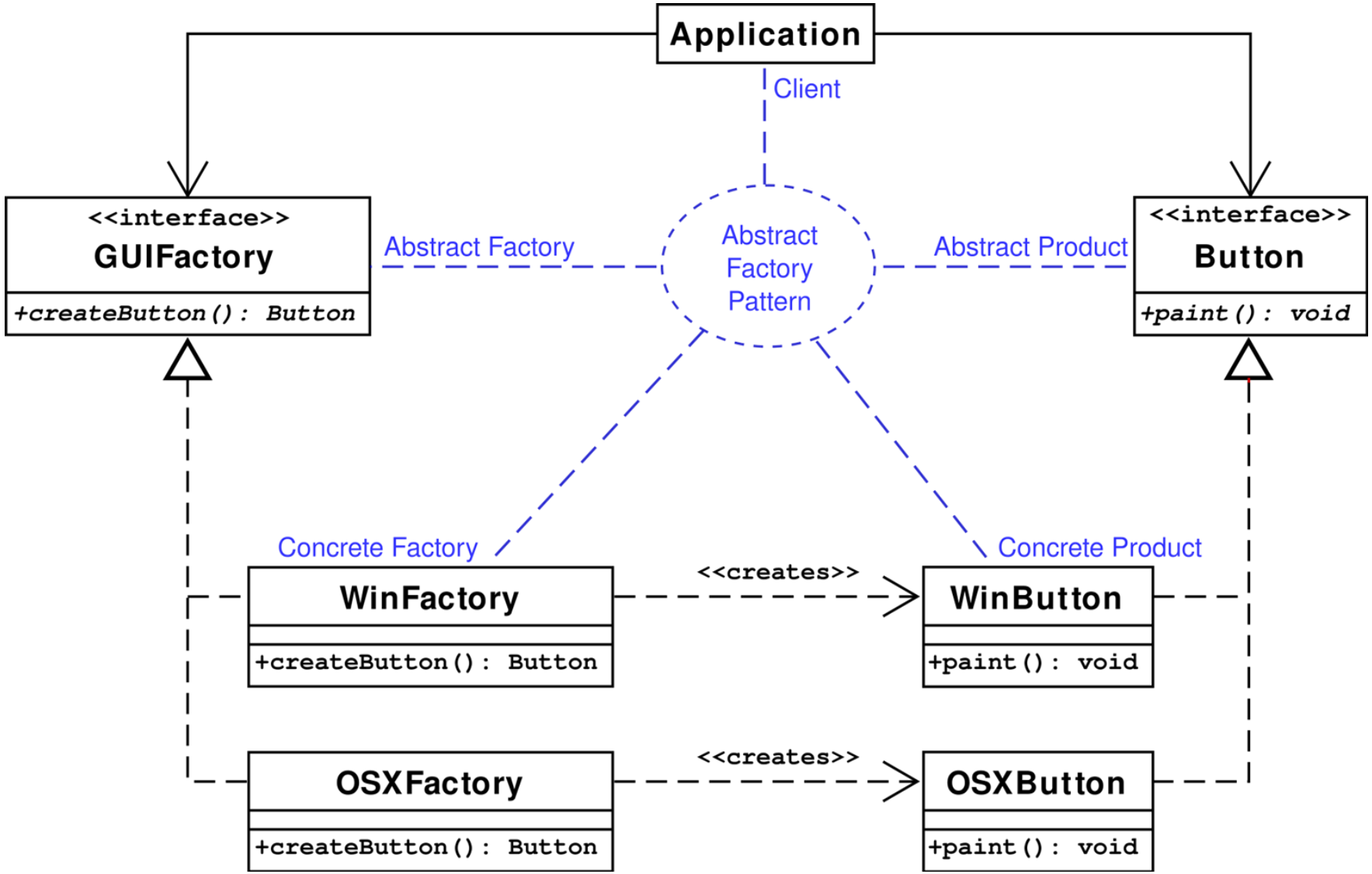
Abstract factory

- poskytnúť rozhranie pre vytvorenie skupiny závislých objektov bez špecifikácie konkrétnej triedy
- Ako môže byť aplikácia nezávislá od toho, ako sa vytvoria jej objekty, a objekty, ktoré potrebuje?
- vytvorenie objektov je skryté v osobitnom objekte
- úloha vytvorenia objektov je delegovaná factory objektu

Abstract factory



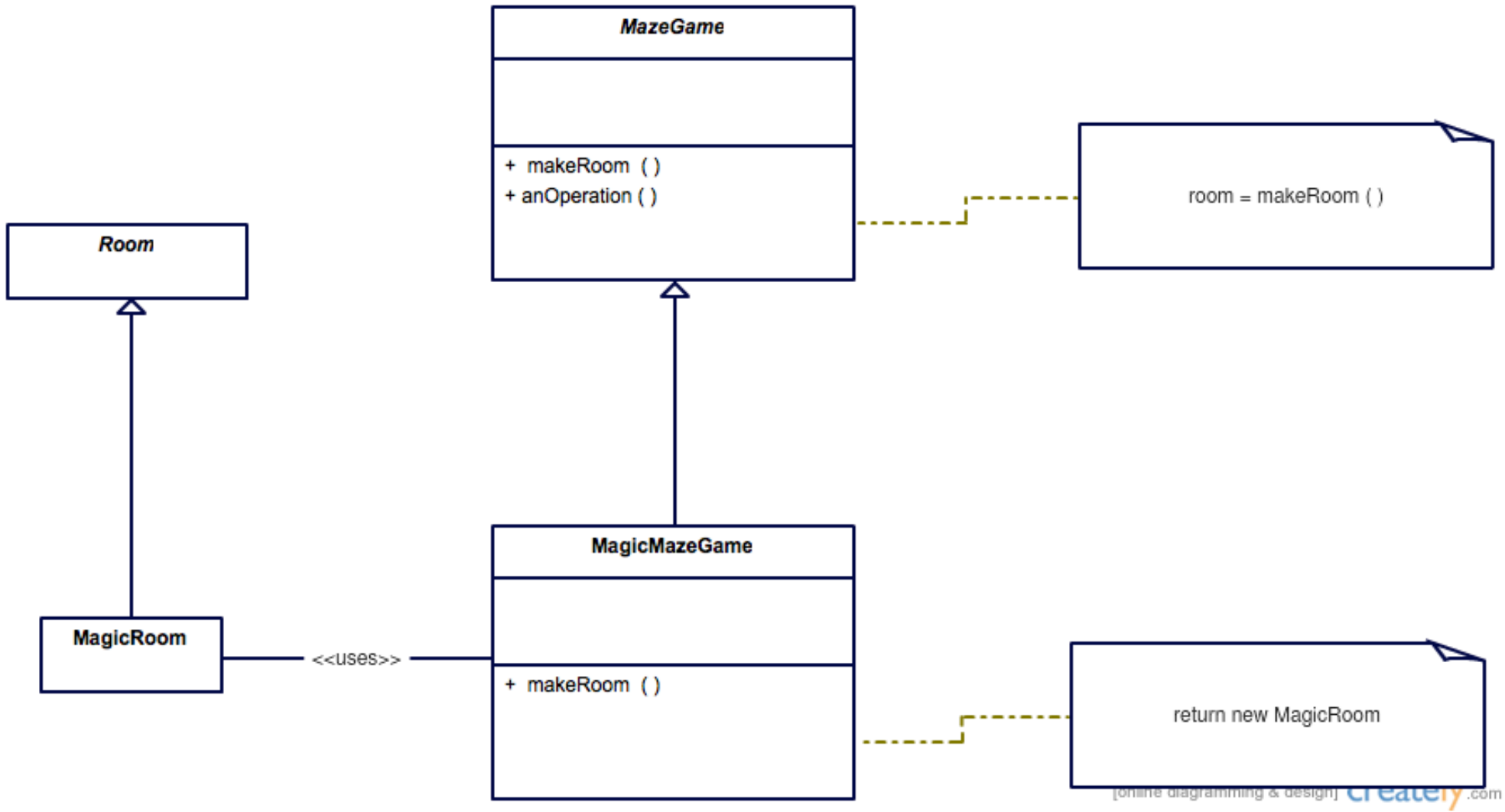
Abstract factory



Factory method

- rozhranie pre vytvorenie jediného objektu, ale podtriedy rozhodujú, inštanciu ktorej triedy majú vytvoriť
- vytvorenie inštancií je úlohou podtried
- osobitná operácia (factory method) je zodpovedná za vytvorenie objektu, objekt vytvoríme zavolaním tejto metódy

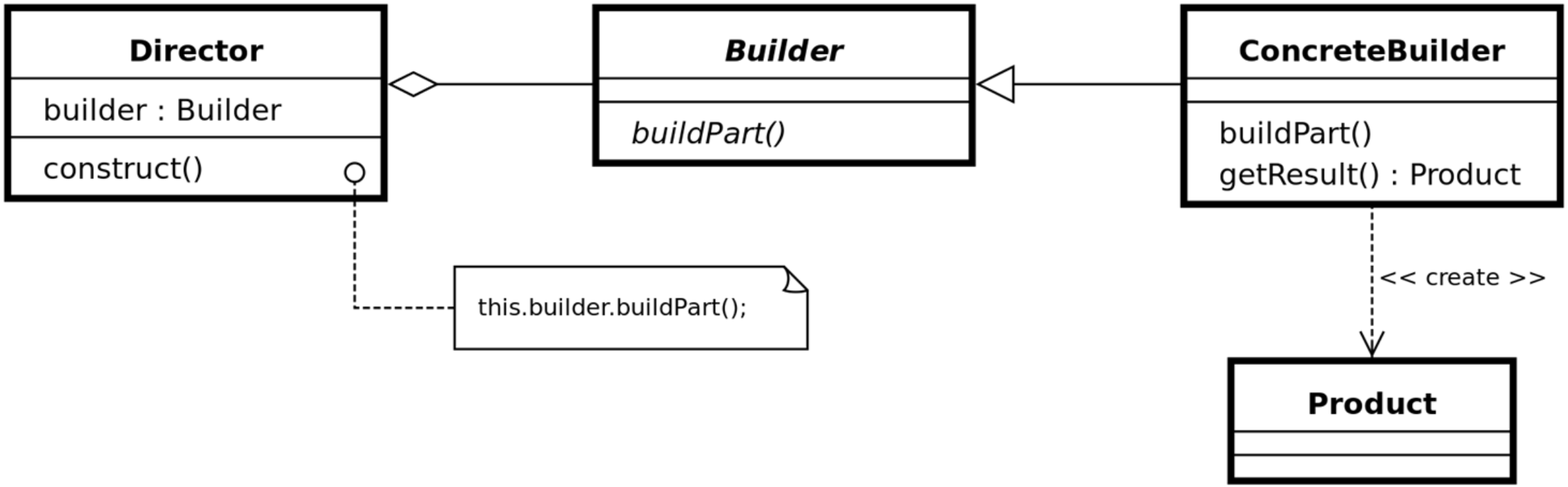
Factory method



Builder

- pre zložité objekty, oddelíme ich reprezentáciu od ich vytvorenia
- rovnaký proces môže vytvoriť rôzne reprezentácie
- Ako môžeme zjednodušiť triedu, ktorá obsahuje vytvorenie zložitého objektu?

Builder


Director
builder : Builder
construct()
Builder
buildPart()
ConcreteBuilder
buildPart()
getResult() : Product
this.builder.buildPart();
<< create >>
Product

Použitie vzoru Builder

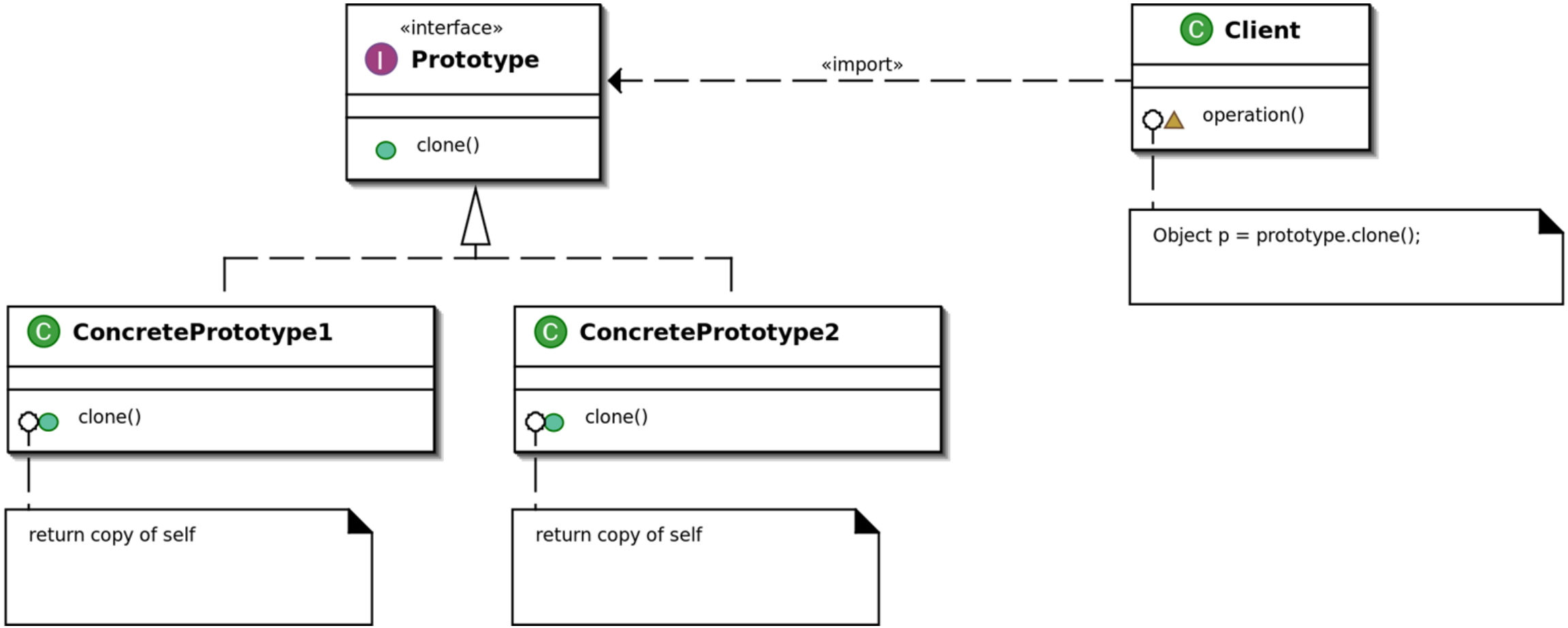
- môžeme meniť vnútornú reprezentáciu objektu
- enkapsulovaný kód pre vytvorenie a reprezentáciu
- môžeme kontrolovať proces vytvorenia

- musíme zadefinovať builder pre každý typ produktu
- triedy buildera môžeme meniť
- ťažší dependency injection

Prototype

- objekty vytvárame na základe prototypovej inštancie
- vytváranie inštancií použitím už existujúceho objektu
- lepší výkon, menšia záťaž na pamäť
- môžeme špecifikovať počas behu, ktorý objekt sa má vytvoriť - dynamicky načítané triedy
- definujeme abstraktnú triedu s metódou `clone()`, ktorú implementujú konkrétne podtriedy

Prototype


«interface»
Prototype
clone()
«import»
Client
operation()
Object p = prototype.clone();
ConcretePrototype1
clone()
return copy of self
ConcretePrototype2
clone()
return copy of self

Zhrnutie

- návrhový vzor
- typy návrhových vzorov
- singleton
- abstract factory
- factory method
- builder
- prototype