Projekti obrasci

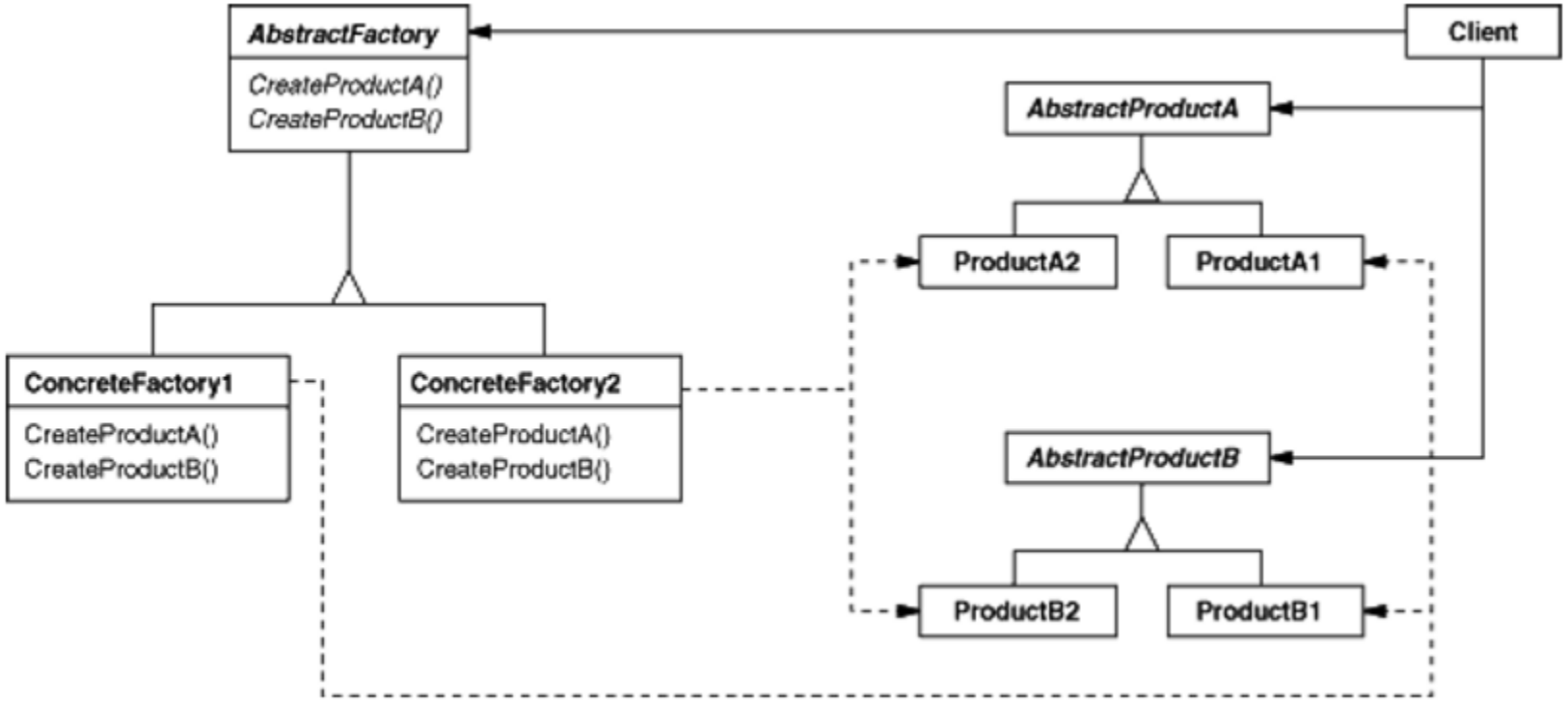
Kreiranje objekata

Uvod

- Enkapsuliraju znanje o tome koje konkretno klase koristi sistem
- Sakrivaju instanciranje objekata i načine na koji se oni povezuju i reprezentuju
- Ostatak sistema treba da zna samo interfejse definisane u apstraktnim klasama
  - Konkretno sistem može da sadrži objekte značajno različitih struktura i funkcionalnosti

Abstract factory

- Obezbjeđuje interfejs za kreiranje skupa povezanih i zavisnih objekata bez specifikovanja klasa kojima pripadaju



Učesnici

- AbstractFactory
  - Deklariše interfejs kojim se kreiraju *AbstractProduct* objekti
- ConcreteFactory
  - Implementira operacije kreiranja konkretnih *Product* objekata
- AbstractProduct
  - Deklariše interfejs za klasu proizvoda
- ConcreteProduct
  - Konkretan objekat iz klase proizvoda kojeg kreira odgovarajući factory objekat
  - implementira AbstractProduct interfejs
- Client
  - Koristi samo interfejse iz AbstractFactory i AbstractProduct klasa

Kolaboracije

- Uobičajeno se kreira jedna instanca *ConcreteFactory* klase. Ovaj objekat kreira produkte konkretne klase. Ako je potrebno kreirati produkte iz druge klase, klijent treba da koristi drugu instancu *ConcreteFactory*

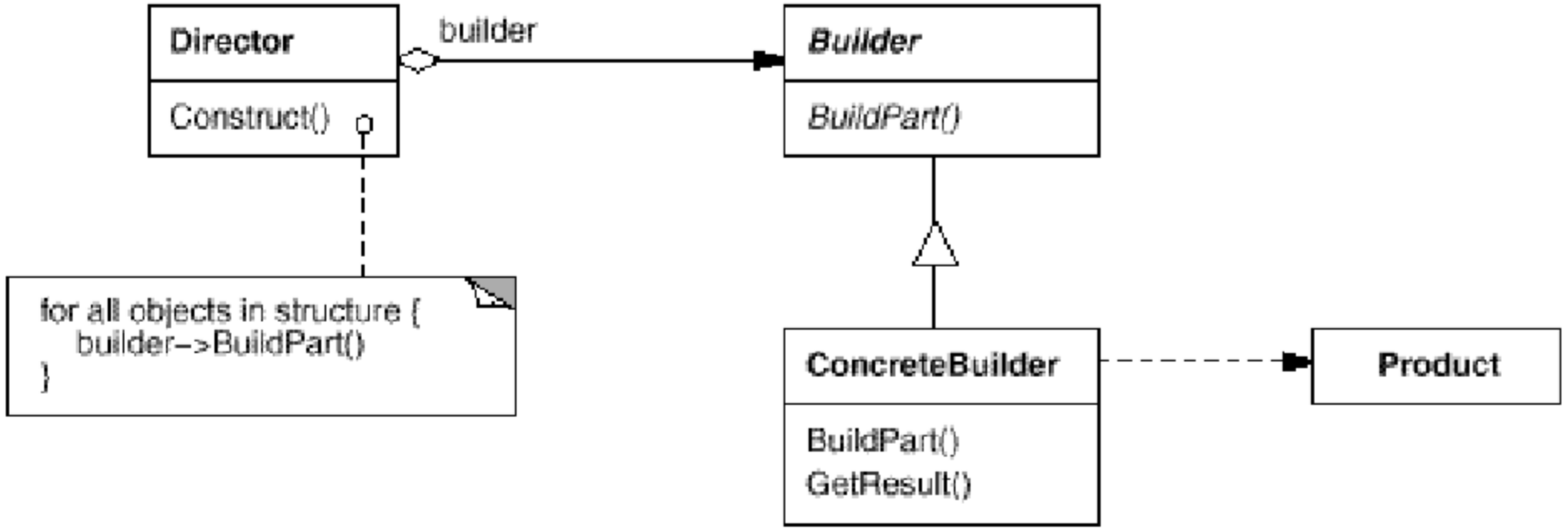
- *AbstractFactory* ne kreira objekte, već odlaže instanciranje na *ConcreteFactory* podklase

Kada se koristi?

- Sistem je nezavisan od načina na koji se kreiraju *Product* objekti, načina na koji se povezuju i reprezentuju
- Sistemu je potrebno više familija proizvoda koji će se koristiti istovremeno
- Potrebno je objaviti samo interfejse prema tipovima proizvoda, a sakriti njihovu implementaciju

Builder

- Razdvaja se procedura konstruisanja kompozitnih objekata od njihove reprezentacije, tako je moguće jednom procedurom kreirati više različitih reprezentacija



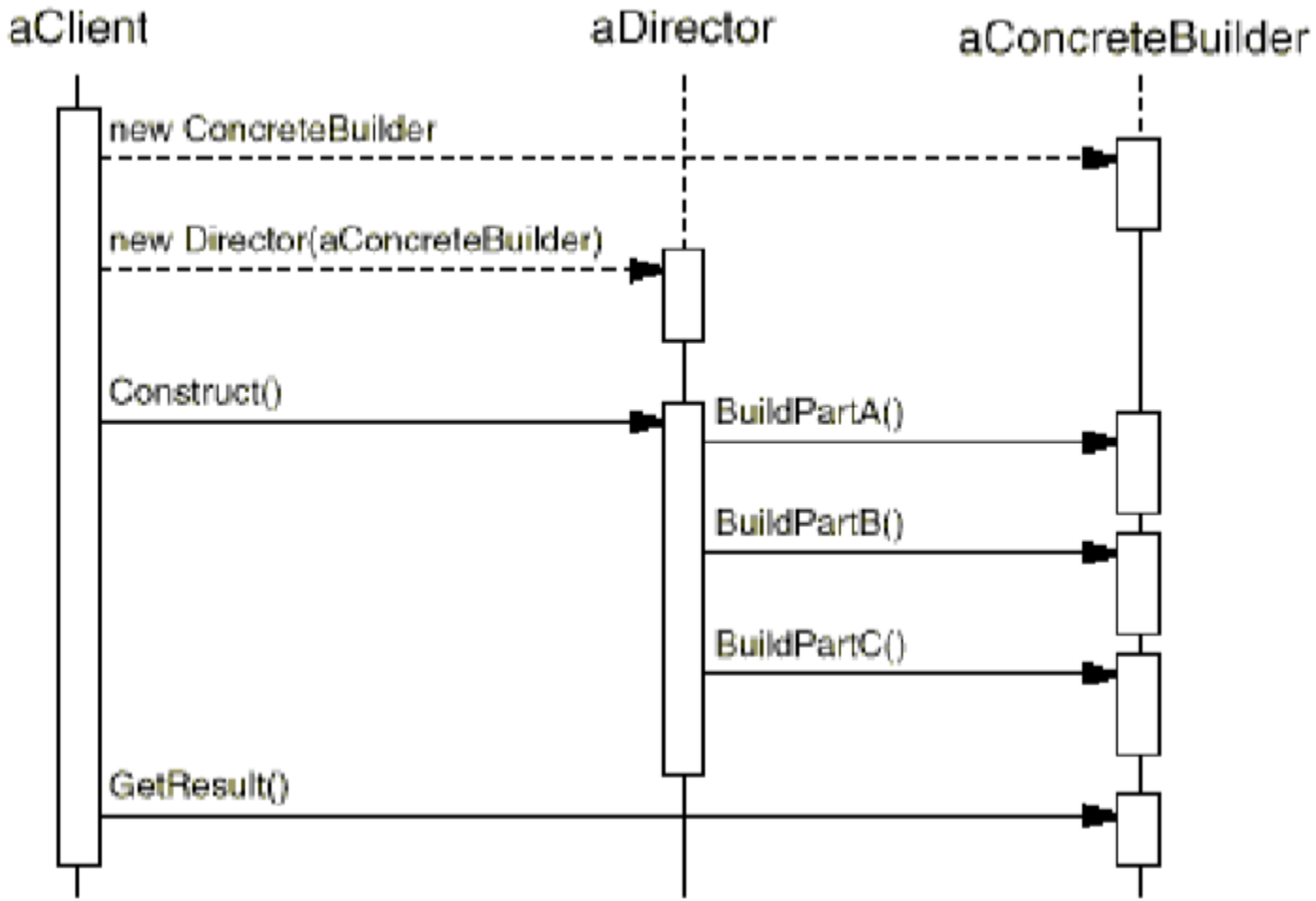
Učesnici

- Builder
  - Definiše interfejs za kreiranje dijelova kompozitnog *Product* objekta
- ConcreteBuilder
  - Kreira i povezuje dijelove, implementira *Builder* interfejs
  - Održava logičku strukturu *Product* objekta, odnosno njegovih dijelova
  - Obezbjeđuje pristup kreiranom objektu
- Director
  - Kreira *Procudt* objekat pomoću *Builder* interfejsa
- Product
  - Reprezentuje kompozitni objekat. *ConcreteBuilder* implementira konkretnu reprezentaciju i proceduru za povezivanje dijelova u cjelinu
  - Uključuje klase koje predstavljaju gradivne elemente, kao i interfejse za njihovo povezivanje u konačni rezultat.

Kolaboracije

- The client creates the Director object and configures it with the desired Builder object.
- Director notifies the builder whenever a part of the product should be built.
- Builder handles requests from the director and adds parts to the product.
- The client retrieves the product from the builder.

Kolaboracije (2)



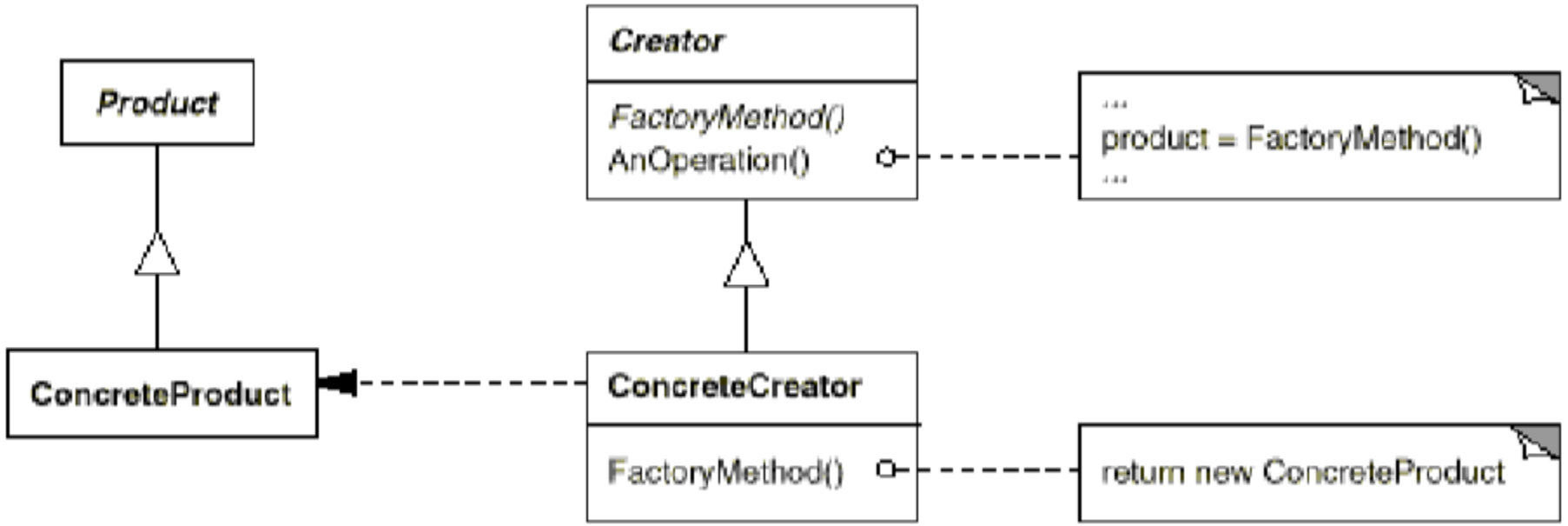
Kada se koristi?

- Algoritam za kreiranje kompozitnog objekta treba da je nezavisan od njegovih djelova, odnosno načina na koji se oni povezuju

- Proces kreiranja kompozitng objekta mora da omogući različite reprezentacije

Factory method

- Definiše interfejs za kreiranje objekata, ali odlaže podklasama konkretno instanciranje - *Factory Method lets a class defer instantiation to subclasses*



Učesnici

- Product
  - Definiše interfejs objekata koji se kreiraju
- ConcreteProduct
  - Implementacija *Product* interfejsa
- Creator
  - Deklariše *FactoryMethod*, koji vraća objekat tip *Product.* Dozvoljeno je da *Creator* definiše difolt implementaciju koja kreira podrazumijevani *ConcreteProduct* objekat
  - U tom smislu, može pozivati FactoryMethod za kreiranje *Product* objekta
- ConcreteCreator
  - Redefiniše *FactoryMethod* koji vraća instancu klase *ConcreteProduct*

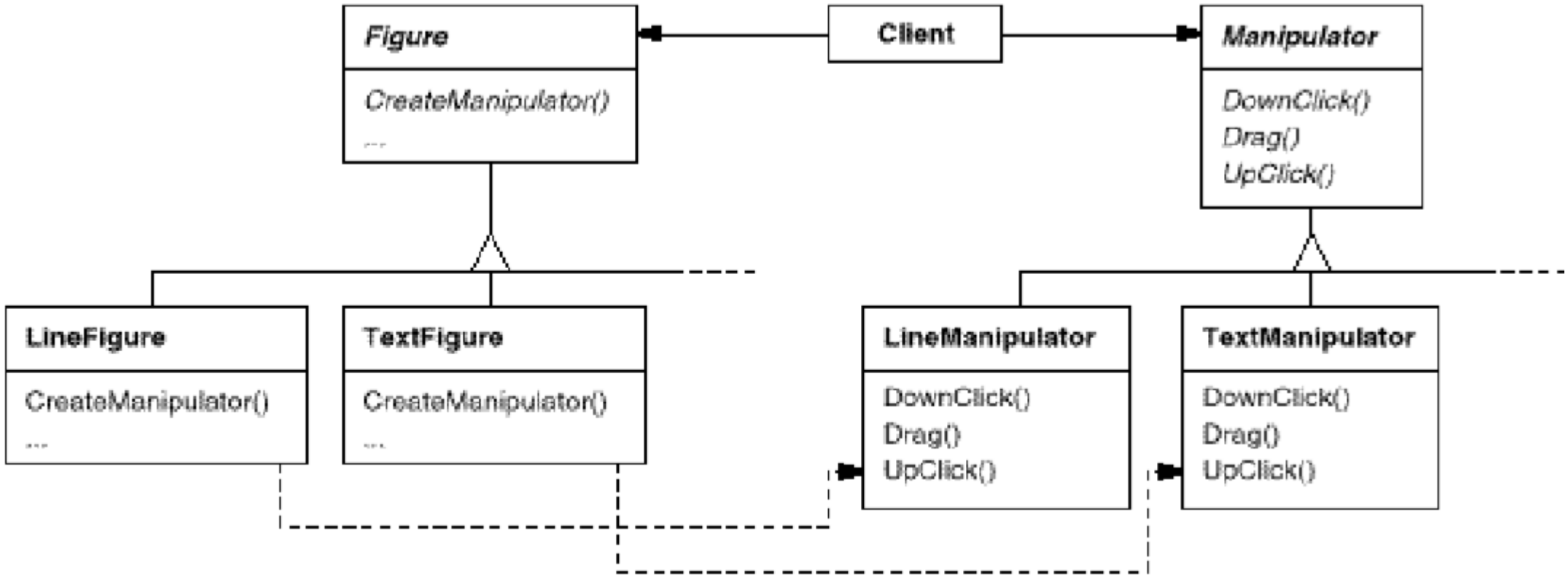
Kolaboracije

- *Creator* se oslanja na podklase, odnosno redefiniciju *FactoryMethod* koja će da vraća instancu odgovarajuće klase *ConcreteProduct*

Kada se koristi?

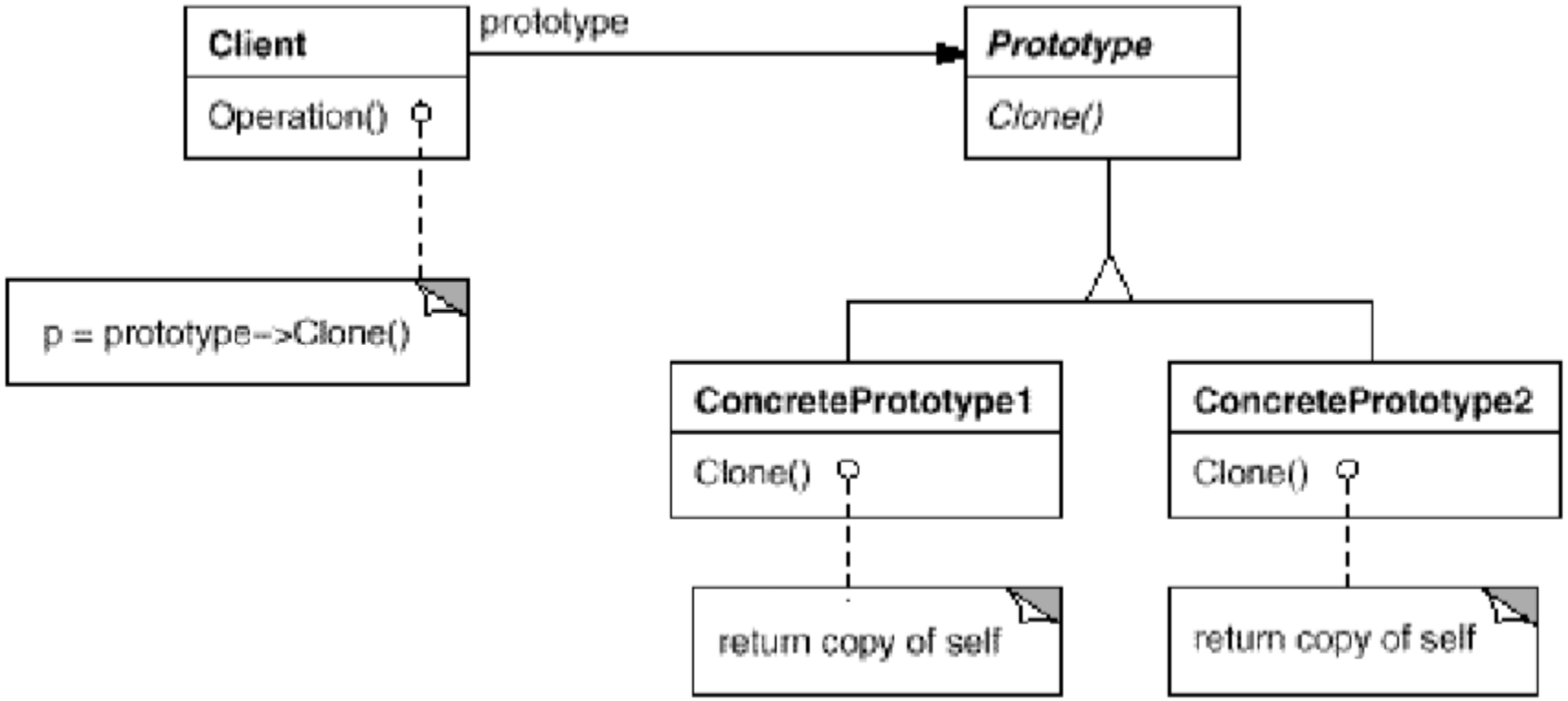
- *Creator* ne zna konkretnu klasu objekta koji kreira
- *Creator* ostavlja svojim podklasama da specificiraju objekat koji se kreira
- Creator delegira odgovornosti na više pomoćnih klasa, povezuje hijerarhije klasa

Kada se koristi?



Prototype

- Specifikuje objekat koji je potrebno kreirati pomoću protopita, objekat se kreira kao kopija odgovarajućeg prototipa



Učesnici

- Prototype
  - Deklariše interfejs za kloniranje objekta
- ConcretePrototype
  - Implementira interfejs za kloniranje
- Client
  - Kreira novi objekta tako što poziva opearciju kloniranja za neki prototip objekat

Kolaboracije

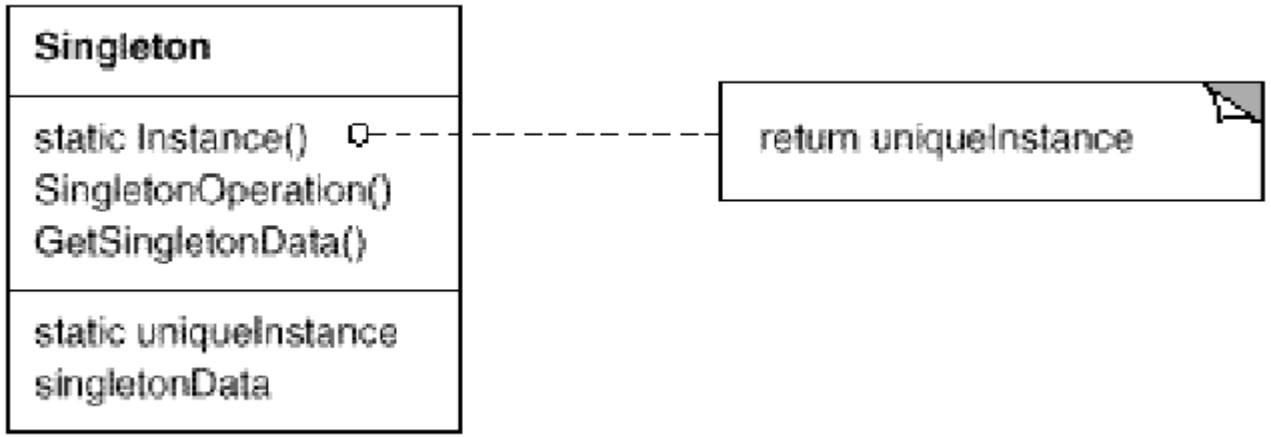
- Klijent poziva operaciju kloniranja za prototip objekat

Kada se koristi?

- Kada se tek u run-time zna koja se klasa instancira
- Izbjegava se kreiranja paralelnih hijerahija kao kod Factory Method obrasca
- Kada objekat može da bude u svega nekoliko različitih stanja – instalira se odgovarajući broj prototip objekata koji se kloniraju po potrebi, umjesto da se klasa instancira za svako stanje eksplicitno (skuplja operacija)

Singleton

- Obezbjeđuje da klasa ima samo jednu instancu. Obezbjeđuje globalnu referencu na tu instancu



Učesnici

- Singleton
  - Definiše operaciju *Instance* metodu na nivou klase i omogućava klijentu da pristupi jedinistvenoj instanci (static metoda C++)
  - Samo-instanciranje

Kolaboracije

- Klijenti pristupaju *Singleton* instanci isključivo preko *Instance* funkcije

Kada se koristi?

- Mora postojati samo jedna instanca klase kojoj se pristupa na jednistveni način

- Omogićiti izvođenje iz klase *Singlton*, a bez uticaja na implementaciju ostatka sistema