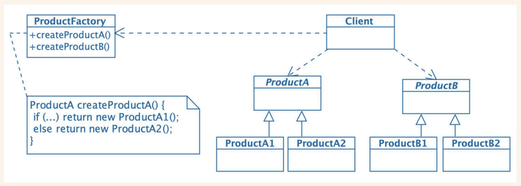
Design Pattern 2

sabato 23 novembre 2024 17:10

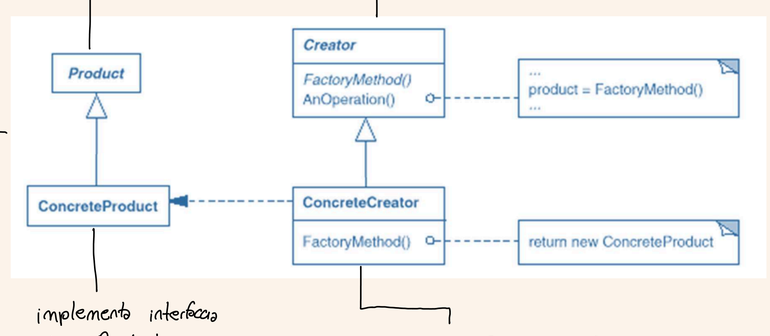
Design Pattern

- Pattern Creazionali — astraggono — processo — istanziazione — oggetti
  - nascondono — creazione — oggetti
  - rendono — sistema — indipendente — come — creati — oggetti
  - semplificano — costruzione — oggetti

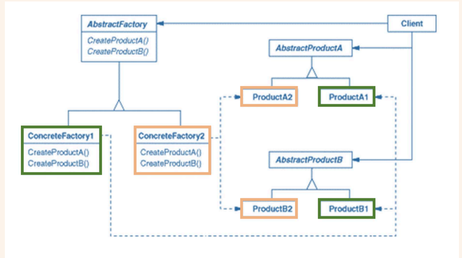
- Creazione oggetti
  - tramite classe
    - Factory — compito — creare / restituire — istanze — altre — classi
      - riduce — linee codice — complessità — tempo creazione oggetti
      - Tre tipi
        - **Simple Factory** (a.k.a. Concrete Factory) — non — Pattern — GoF
          - semplificazione — Abstract Factory
          - risolve — problema — creazione — oggetti — complessi
          - separa — logica creazione / funzionalità oggetto
          - crea — oggetto Factory — gestisce — creazione



        - **Factory Method**
          - definisce interfaccia tipo oggetti creati da Factory Method
          - dichiara il Factory Method

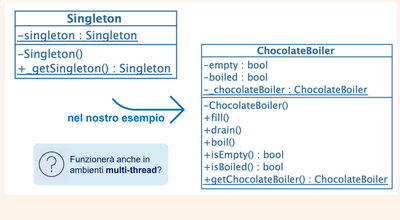


          - implementa interfaccia di Product
          - sovrascrive factory method per restituire istanza del Concrete Product
          - Utilizzo — consente — disaccoppiamento — classe — da — classi — create / utilizzate
            - delega — sottoclassi — creazione — oggetti
          - vantaggi — codice — più — flessibile — riutilizzabile
            - «code to an interface» — classe — basata — interfaccia di Product — funziona — con qualsiasi — ConcreteProduct — supporta — interfaccia
          - svantaggi — necessario — estendere — classe — Creator — istanziare — ConcreteProduct
          - implementazione — Creator — possono essere — astratti / concreti
            - se — usato — creare — diversi — product — serie — parametro — decidere quale — (if-then-else)
        - **Abstract Factory** — delega — altre classi — decisione — quali — classi — istanziare
          - Abstract vs Factory Method
            - Abstract Factory — delegata — ad altri oggetti — tramite — composizione
            - Factory Method — delegata — sottoclassi — tramite — ereditarietà

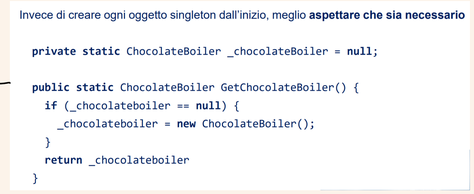


- Pure Fabrication — pattern GRASP — assegna — responsabilità — fortemente correlate — classe artificiale
  - introdotta — convenienza — designer
  - non — rappresenta — niente — dominio — problema
  - implementa — behavioural decomposition — obiettivo — riuso / alta coesione / basso accoppiamento
- Factory — è un — Pure Fabrication — obiettivo
  - confinare — responsabilità — in — oggetti — coesi
  - incapsulare — complessità — logica — creazione

- Singleton Pattern — garantisce — classe — abbia — una sola istanza — fornendo — punto — accesso — globale — istanza



  - 1) Si rende il costruttore della classe privato
    `private ChocolateBoiler() = { ... }`
    2) Si aggiunge un oggetto statico privato (che conterrà l'unica istanza disponibile)
    `private static ChocolateBoiler _chocolateBoiler = new ChocolateBoiler()`
    3) Si rende l'unica istanza disponibile accessibile solo attraverso un metodo statico
    `public static ChocolateBoiler getChocolateBoiler() { return _chocolateBoiler; }`
    — inoltre — uso — inizializzazione Lazy —



    - perché? — informazioni — insufficienti — istanziare — Singleton
    - Singleton — potrebbe — essere — resource intensive — non necessario
  - se — usato — multi-thread — rischio — inizializzare — più istanze — insieme
    - soluzione
      - Double-checked locking — evita — sincronizzazioni — costose — invocazioni — eccetto — prima

  - se — esteso — classe — singleton — garantisce — unica istanza — della — sottoclasse — tramite — ogni — sottoclasse — fornisce — metodo — statico — getIstance — decide — sottoclasse — istanziare
  - Singleton vs Attributi Statici
    - Attributi Statici — oggetti — classe — devono — condividere — una sola copia — variabile
    - Singleton — pro
      - più — essere — passato — come — parametro
      - più — essere — esteso — sottoclassi
      - più — essere — istanziato — mediante — fabbrica