Design Pattern: Factory Method

De Long Iu

DEUTSCHES KREBSFORSCHUNGSZENTRUM
IN DER HELMHOLTZ-GEMEINSCHAFT

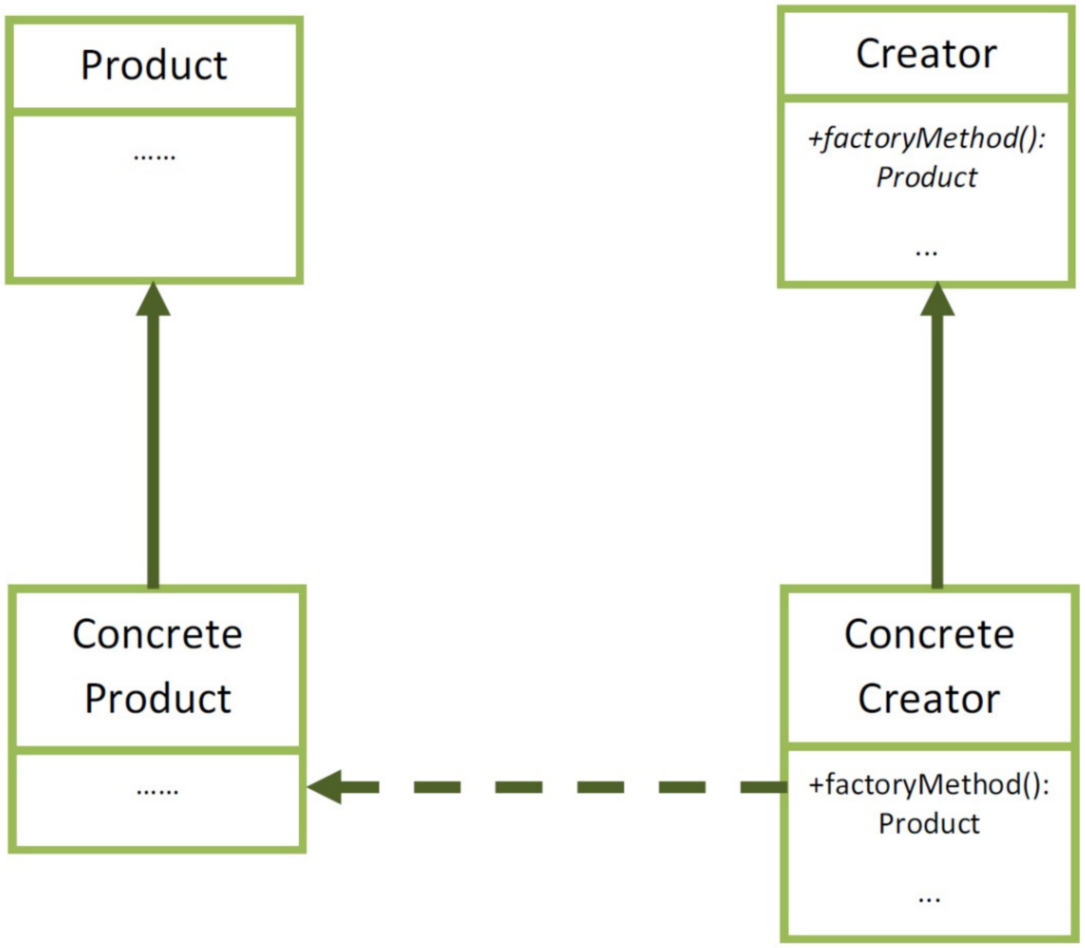
“ The elements of this language are entities called patterns. Each pattern describes a problem that occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.”

by Christopher Alexander

Design Pattern

- Guidelines for implementing software
- Approved designs to solve architectural problems
- Different types:
    - Creational
    - Structural
    - Behavioral
    - Concurrency

Factory Method



Factory Method

```
// Product
class Food {
};

// Concrete product
class Pizza : public Food {
public:
  Pizza() {
    std::cout << "Pizza ready." << std::endl;
  };
};

// Another concrete product
class Sausage : public Food {
public:
  Sausage(const char* side) {
    std::cout << "Sausage grilled." << std::endl;
    if(side) {
      std::cout << "Served with " << side << std::endl;
    }
  };
};
```

Factory Method

```
// Creator
class Restaurant {
protected:
  Food* _Food;

  // The factory method. The concrete product will be created here.
  virtual void prepareFood() = 0;

  virtual void takeOrder() {
    std::cout << "Your order please!" << std::endl;
  };

  virtual void serveFood() {
    std::cout << "Enjoy your meal!" << std::endl;
  };

public:
  // This method is using the factory method
  void deliverFood() {
    takeOrder();
    prepareFood(); // Call the factory method
    serveFood();
  }
};
```

Factory Method

```
// Concrete creator for concrete product "Pizza"
class Pizzeria : public Restaurant {
public:
  virtual void prepareFood() {
    _Food = new Pizza();
  }
};

// Concrete creator for concrete product "Sausage"
class HotDogStand : public Restaurant {
public:
  virtual void prepareFood() {
    _Food = new Sausage("Fries and Ketchup");
  }
};

int main() {
  Pizzeria daToni;
  daToni.deliverFood();

  HotDogStand theHotGrill;
  theHotGrill.deliverFood();
}
```

Factory Method

- Dynamic Factory Pattern
  - Basically same structure
  - Meta data is stored externally
  - Classes can be instantiated at runtime using reflection

Factory Method

- MITK classes:
  - AddContourToolFactory
  - BinaryThresholdToolFactory
  - NrrdTensorImageWriterFactory
  - ObjFileIOFactory
  - OpeningToolFactory
  - RegionGrowingToolFactory
  - …..

Pros and Cons

- Pros:
  - Loose coupling
  - Information hiding
  - Descriptive names

- Cons:
  - A lot of subclasses

THANKS