Creational Design Patterns

Part 3

Creational Design Patterns

- Singleton
- Factory Method
- Abstract Factory
- Builder
- Prototype

Creational Patterns: Abstract Factory

- Factory method allows us to create one product through inheritance

  i.e., the monsters: fire monsters and ice monsters

- Sometimes, we want to create families of related products

- Consider our `GameLevel` classes
  - In addition to specific monsters, we may want levels to have a specific floor, sky, walls, and so on

Creational Patterns: Abstract Factory

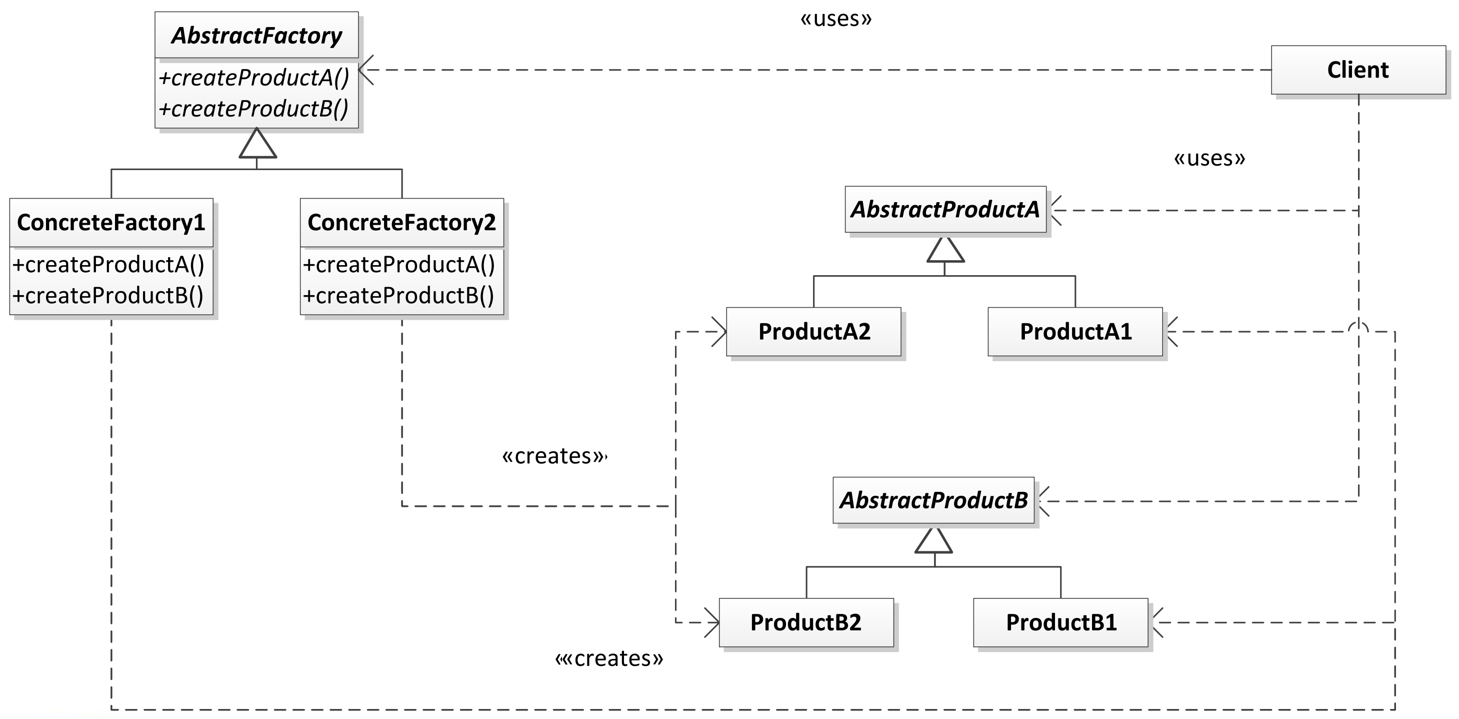
Design Pattern:
Abstract Factory

Provide an interface for creating families of related or dependent objects without specifying their concrete classes.

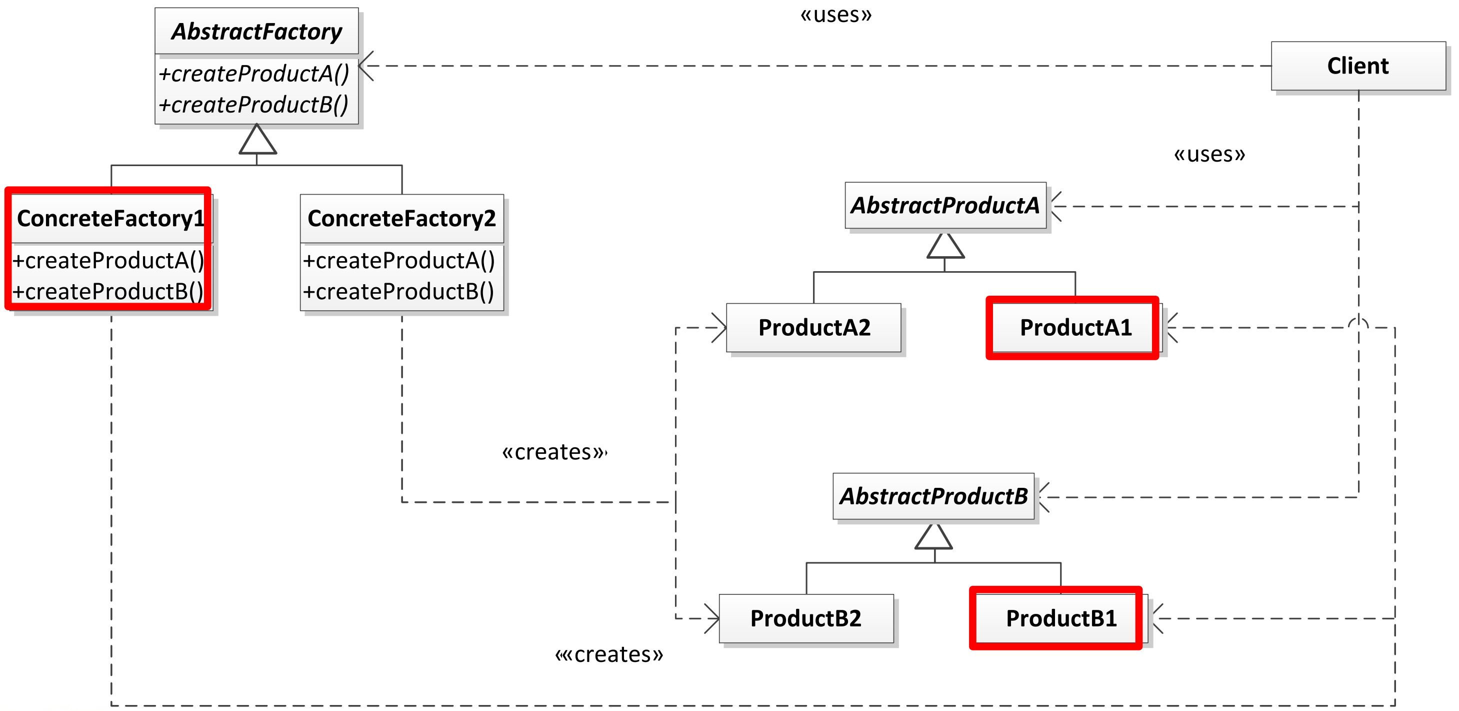
Creational Patterns: Abstract Factory

- Applicability:
  - A system should be independent of how its products are created
  - A system should be configured with one of multiple families of products
  - A family of related product objects are designed to be used together, and you need to enforce this constraint
  - You want to provide a class library of products, and you want to reveal just their interfaces, not their implementations

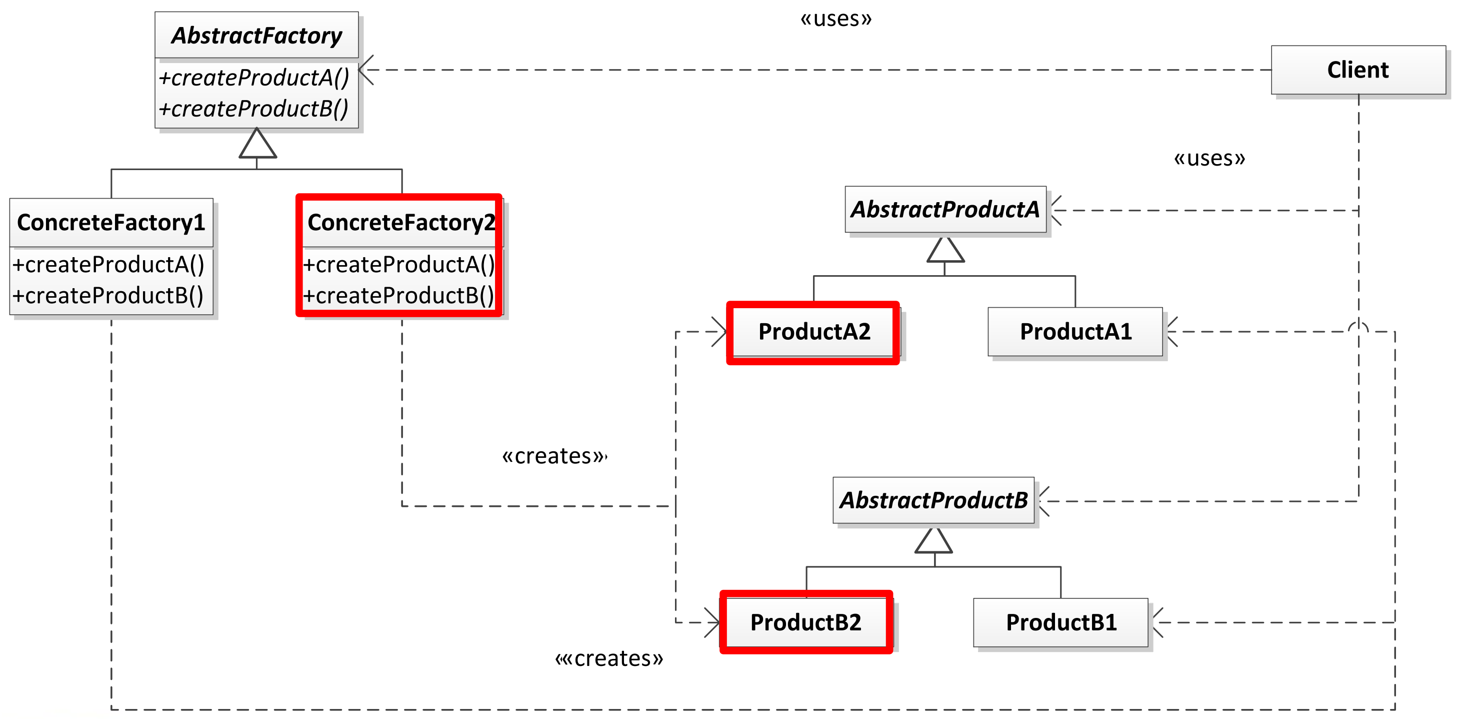
Creational Patterns: Abstract Factory



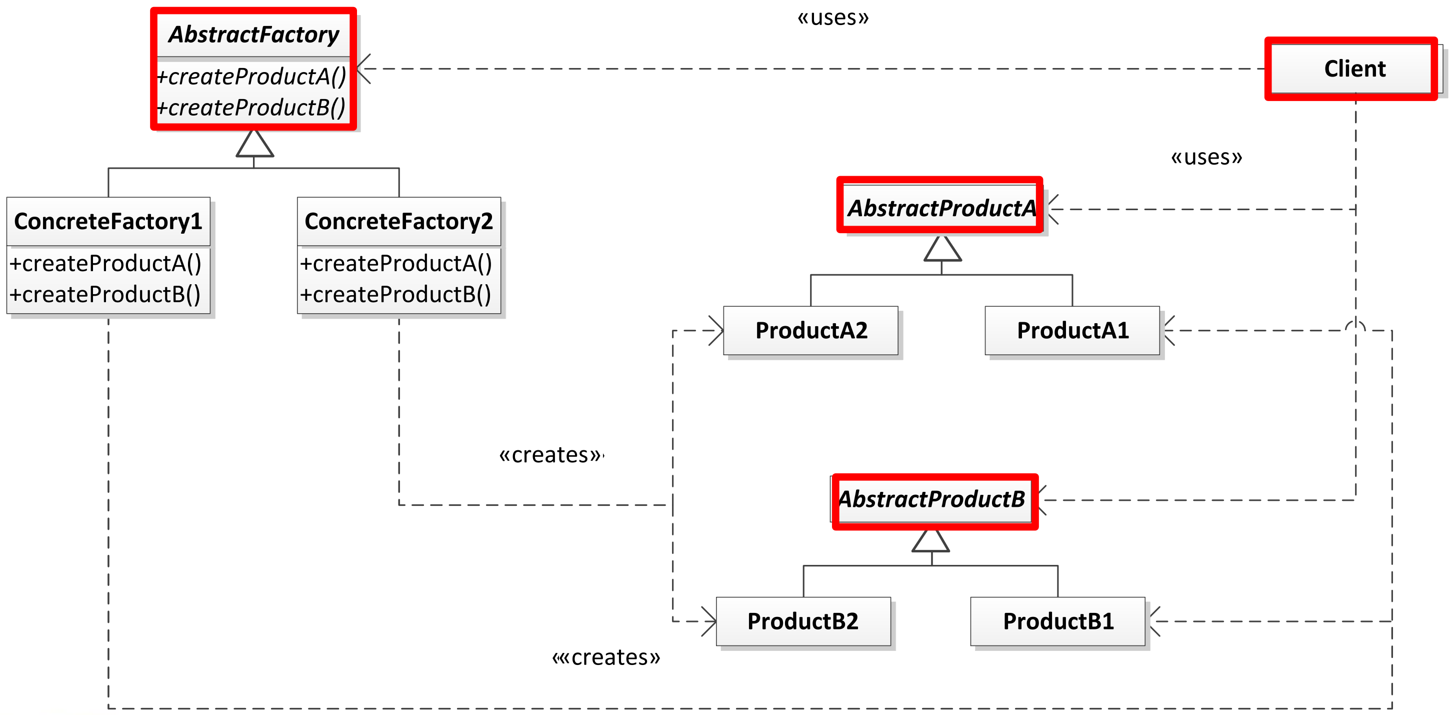
Creational Patterns: Abstract Factory



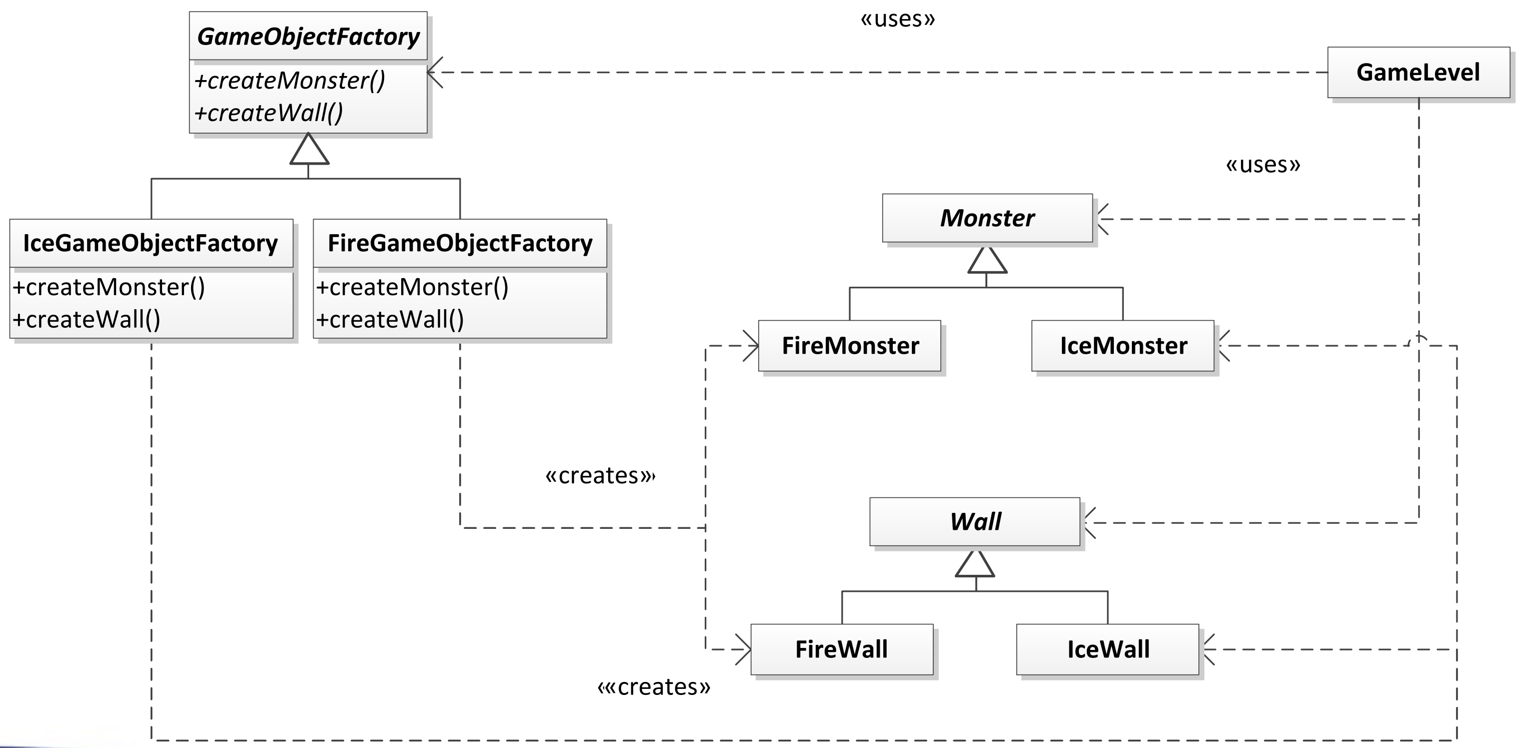
Creational Patterns: Abstract Factory



Creational Patterns: Abstract Factory



Creational Patterns: Abstract Factory



Creational Patterns: Abstract Factory

```
class GameLevel
{
   public:
      GameLevel(GameObjectFactory* factory)
      {
         this->_factory = factory;
         Monster* m1 = factory->createMonster();
         Monster* m2 = factory->createMonster();
         Wall* w1 = factory->createWall();
         // ...
      }
   private:
      GameObjectFactory* _factory;
};
```

Creational Patterns: Abstract Factory

Consequences:

- Isolates concrete classes
  - Client controls when objects are created
  - Factory controls which objects are created and how

a factory ensures the consistency of the objects

- Makes exchanging product families easy
- Promotes consistency among products
- Supporting new kinds of products is difficult

Creational Patterns: Abstract Factory

- Factory Method:
  - Creates a single product
  - Uses inheritance
  - Superclass methods remain generic and use the factory method as needed to create the product
- Abstract Factory:
  - Collects multiple factory methods into a class to create multiple related products
  - Uses aggregation / composition
  - Client remains generic and uses the factory as needed to create the products