# Abstract Factory Pattern

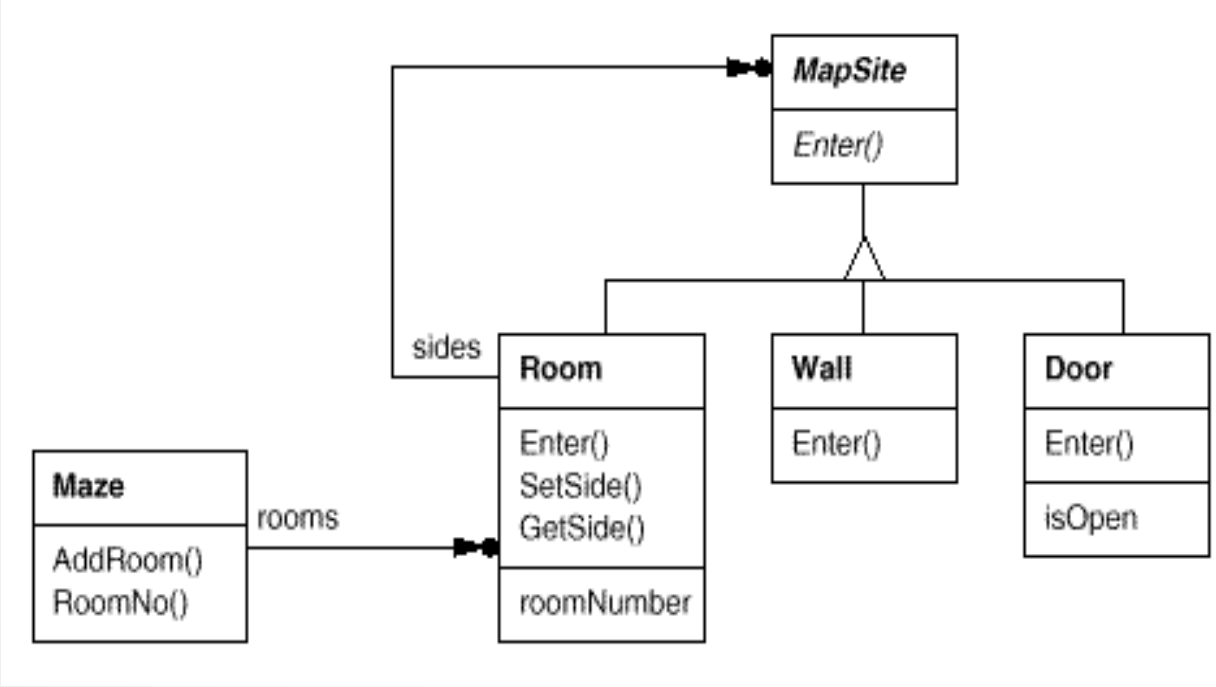
# 생성 패턴(Creational Pattern)

- 인스턴스를 만드는 절차를 추상화하는 패턴
- Why Creational Pattern is Important?
  - 시스템이 상속보다 복합 방법을 사용하는 쪽으로 진화(Why?)
  - 따라서 복잡한 행동을 만드는 데 필요한 구성요소가 될 수 있는 기본적인 행동 집합을 정의하는 쪽에 더 많은 관심과 노력

# 생성 패턴(Creational Pattern)

- 생성 패턴이란?
  - 시스템이 어떤 구체 클래스를 사용하는지에 대한 정보를 **캡슐화**
  - 클래스의 인스턴스들이 **어떻게 만들고 어떻게 서로 맞붙는지**에 대한 부분을 가려준다.
  - 결론: 무엇이 생성되고 누가 생성하며 어떻게 생성되는지 언제 생성할 것인지에 대한 유연성 확보

# 생성 패턴(Creational Pattern)




- cpp.sh/6jess

# 생성 패턴(Creational Pattern)

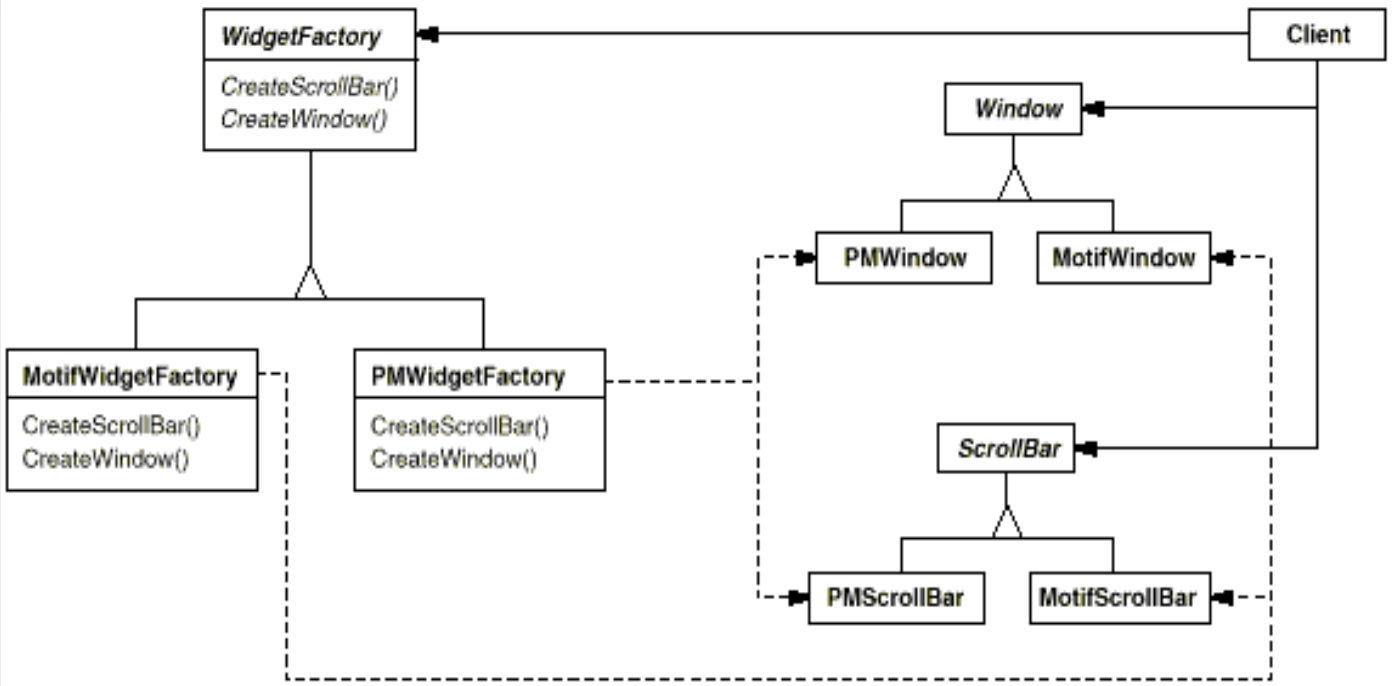
- How about other maze?
- How to create other layout maze?
- How I make Enchanted Maze?

- Creational Patterns show how to make this design more ***flexible***. In particular, they will make it **easy to change** the classes that define the components of a maze.

# 추상 팩토리(Abstract Factory)

- Intent
  - 상세화된 서브클래스 정의 없이 서로 관련성 있거나 독립적인 여러 객체의 군을 생성하기 위한 인터페이스 제공
- Motivation
  - 서로 다른 룩앤필 표준에 상관없이 이식성을 가지려면 응용프로그램이 각 사용자 인터페이스 툴킷에서 제공하는 위젯을 직접 사용하지 못하도록 해야함

# 추상 팩토리(Abstract Factory)




- WidgetFactory
  - 위젯의 기본 사용자 인터페이스 요소를 생성할 수 있는 인터페이스
- Window
  - 생성할 룩앤필에 대응하여 사용자 인터페이스 툴킷이 제공하는 위젯 및 구현 방식에 따라 PMWindow, MotifWindow 등을 만듬

# 추상 팩토리(Abstract Factory)

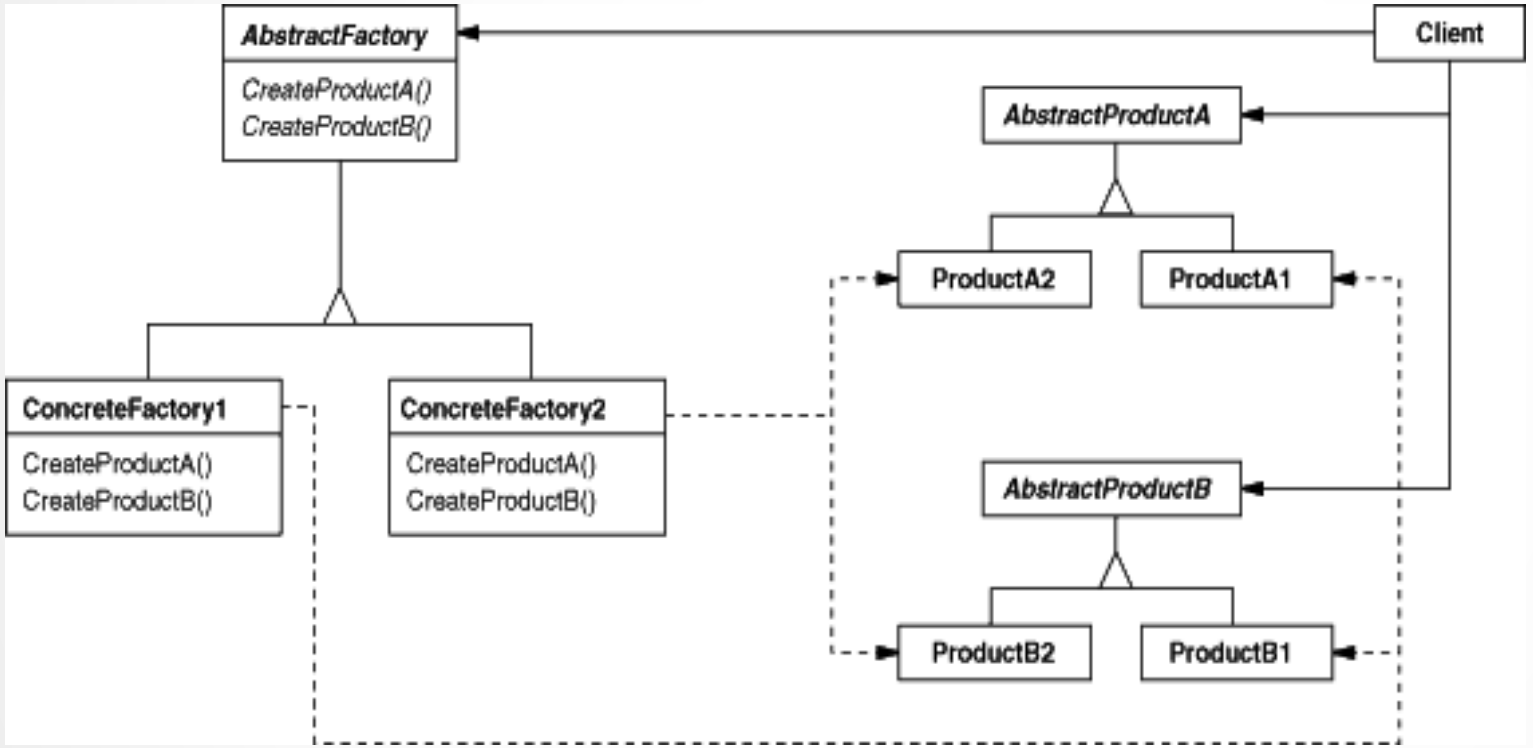
- Applicability
  - 객체가 생성되거나 구성, 표현되는 방식과 무관하게 시스템을 독립적으로 만들고자 할 때
  - 여러 제품군 중 하나를 선택하고 한번 구성한 제품을 다른 것으로 대체해야 할 때
  - 관련된 제품 객체들이 함께 사용되도록 설계되었고 이 부분에 대한 제약이 외부에도 지켜지도록 할 때
  - 제품에 대한 클래스 라이브러리를 제공하고, 그들의 구현이 아닌 인터페이스를 노출시키고 싶을 때

# 추상 팩토리(Abstract Factory)

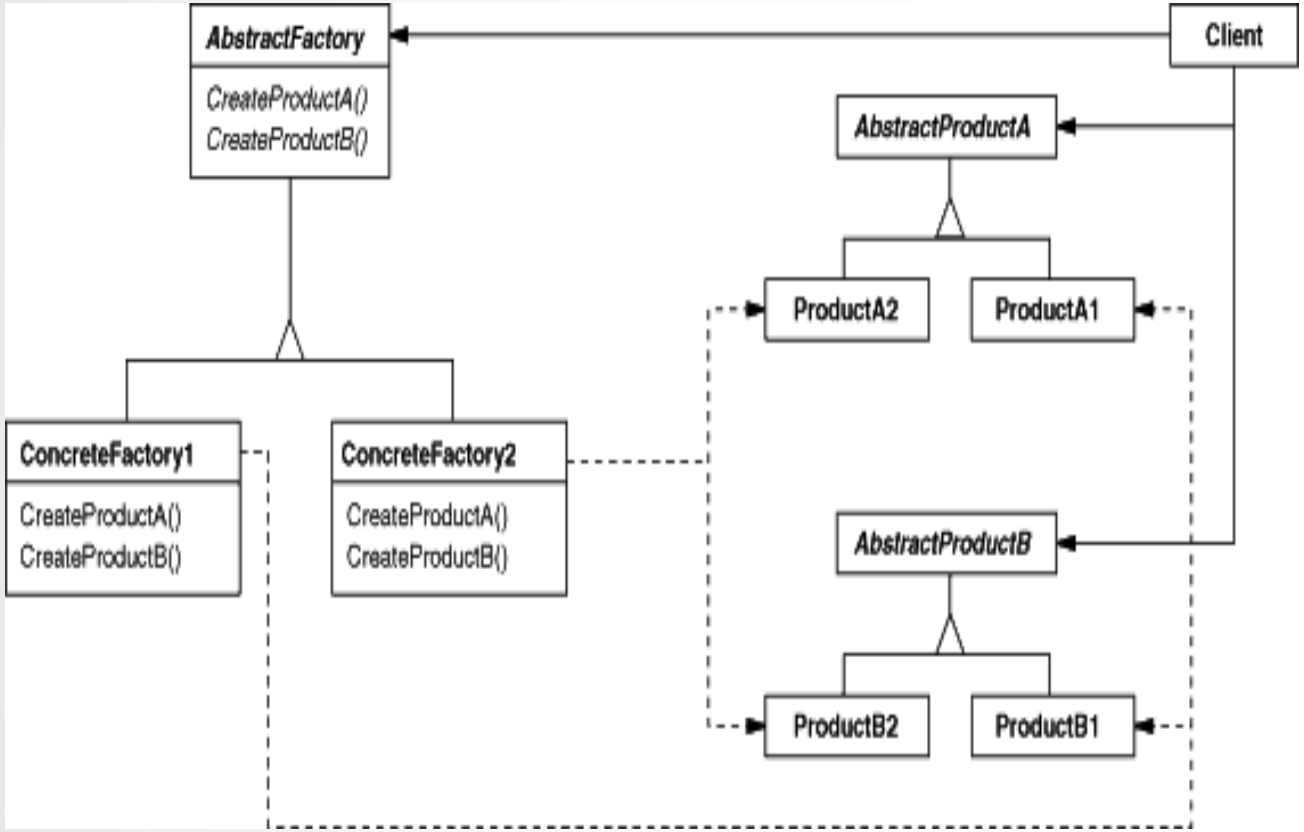
- Applicability
  - 객체가 생성되거나 구성, 표현되는 방식과 무관하게 시스템을 독립적으로 만들고자 할 때
  - 여러 제품군 중 하나를 선택하고 한번 구성한 제품을 다른 것으로 대체해야 할 때
  - 관련된 제품 객체들이 함께 사용되도록 설계되었고 이 부분에 대한 제약이 외부에도 지켜지도록 할 때
  - 제품에 대한 클래스 라이브러리를 제공하고, 그들의 구현이 아닌 인터페이스를 노출시키고 싶을 때

# 추상 팩토리(Abstract Factory)

- Structure



# 추상 팩토리(Abstract Factory)




Participants

- AbstractFactory
  - 개념적 제품에 대한 객체를 생성하는 인터페이스
- ConcreteFactory
  - 구체적인 제품에 대한 객체를 생성하는 연산 구현
- AbstractProduct
  - 개념적 제품 객체에 대한 인터페이스
- ConcreteProduct
  - 팩토리가 생성할 객체
- Client
  - AbstractFactory, AbstractProduct 사용

# 추상 팩토리(Abstract Factory)

- Collaborations
  - ConcreteFactory 인스턴스가 런타임에 만들어짐
  - ConcreteFactory를 통해 특정 구현을 갖는 제품 객체를 생성. 다른 제품군의 제품 객체를 생성하려면 다른 ConcreateFactory를 사용해야함
  - AbstractFactory는 필요한 제품 객체를 생성하는 책임을 ConcreteFactory 서브 클래스에 위임

# 추상 팩토리(Abstract Factory)

- Consequences
  - It isolates concrete classes.
  - It makes exchanging product families easy.
  - It promotes consistency among products.
  - Supporting new kinds of products is difficult(*limitation*).

# 추상 팩토리(Abstract Factory)

- Implementation
  - Useful Techniques
    - Factories as singletons.
    - Creating the products.
    - Defining extensible factories.