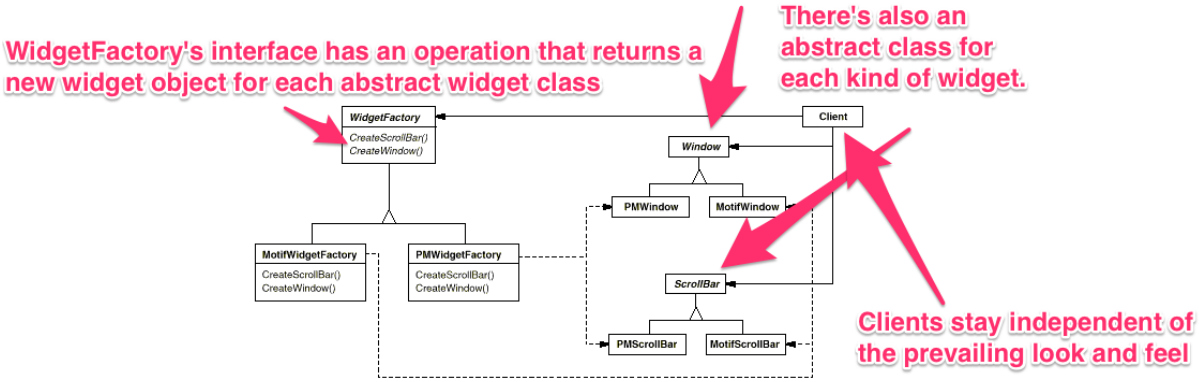
Abstract Factory - Object Creational

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

Also Known As (别名)
Kit

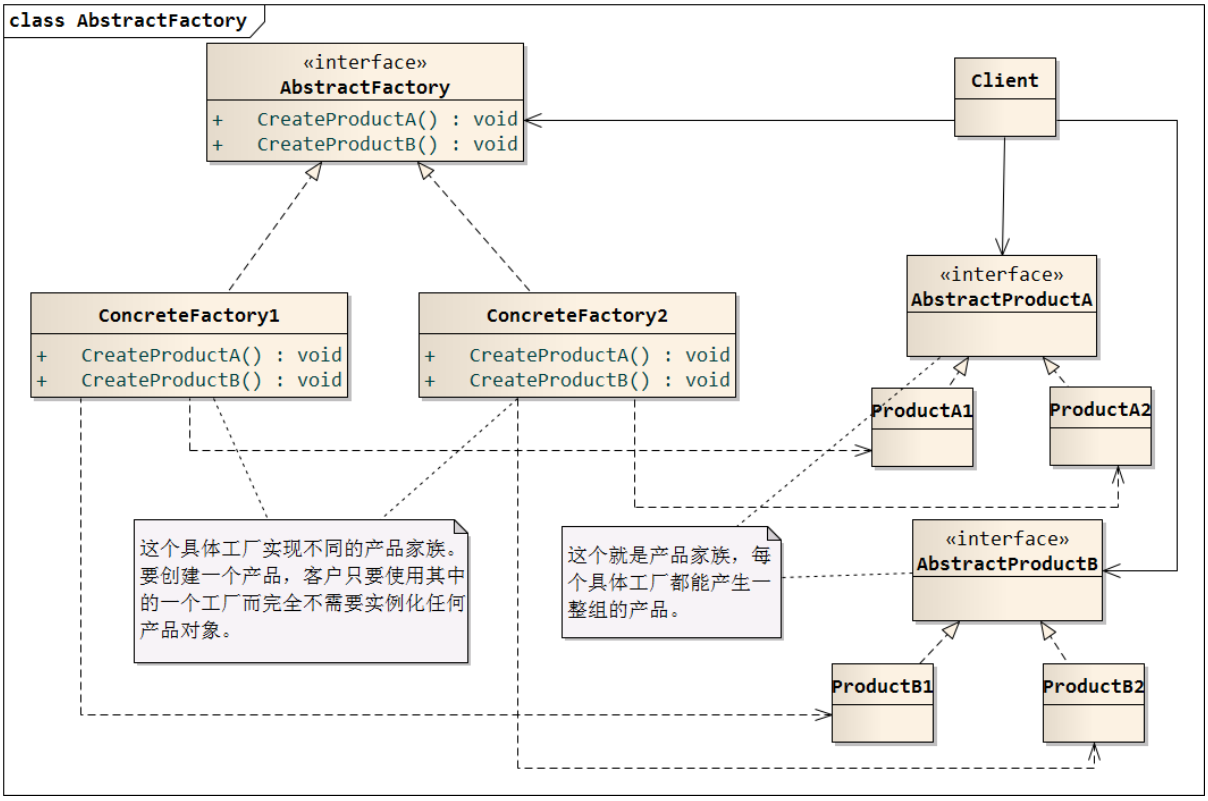
Motivation (动机)
Consider a user interface toolkit that supports multiple look-and-feel standards. Different look-and-feels define different appearances and behaviors for user interface.
- Define an abstract WidgetFactory class that declares an interface for creating each basic kind of widget.



- There’s also an abstract class for each kind of widget, and concrete subclasses implement widgets for specific look-and-feel standards.
- Clients only have to commit to an interface defined by an abstract class, not a particular concrete class.
- A WidgetFactory also **enforces** dependencies between the concrete widget classes.

Applicability (适用性)
- a system should be independent of how its products are created, composed, and represented.
- a system should be configured with one of multiple families of products.
- a family of related product objects is 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.

Structure (结构)



Participants (参与者)

- **AbstractFactory**

  - declares an interface for operations that create abstract product objects.

- **ConcreteFactory**

  - implements the operations to create concrete product objects.

- **AbstractProduct**

  - declares an interface for a type of product object.

- **ConcreteProduct**

  - defines a product object to be created by the corresponding concrete factory.
  - implements the AbstractProduct interface.

- **Client**

  - uses only interfaces declared by AbstractFactory and AbstractProduct classes.

Collaborations (协作)

- Normally a single instance of a ConcreteFactory class is created at run-time.
- AbstractFactory defers creation of product objects to its ConcreteFactory subclass.

Consequences (结果)

Benefits:

- It isolates concrete classes.
- It makes exchanging product families easy.
- It promotes **consistency** among products.

Liabilities:

- Supporting new kinds of products is difficult.

Implementation (实现)

- Factories as singletons.
- Creating the products.
    - AbstractFactory only declares an interface for creating products.
    - the concrete factory can be implemented using the Prototype pattern.
- Defining extensible factories.
    - A more flexible but less safe design is to add a parameter to operations that create objects.

Sample Code (代码示例)

Sample 1

```
class MazeFactory
{
public:
MazeFactory();

virtual Maze *MakeMaze() const;
virtual Wall *MakeWall() const;
virtual Room *MakeRoom(int n) const;
virtual Door *MakeDoor(Room *r1, Room *r2) const;
};

class EnchantedMazeFactory : MazeFactory
{
public:
EnchantedMazeFactory();

virtual Room *MakeRoom(int n) const;
virtual Door *MakeDoor(Room *r1, Room *r2) const;
};

class BombedMazeFactory : MazeFactory
{
public:
BombedMazeFactory();

virtual Wall *MakeWall() const;
virtual Room *MakeRoom(int n) const;
};
```

```
Maze*MazeGame::CreateMaze (MazeFactory& factory)
{
Maze* aMaze = factory.MakeMaze();
Room* r1 = factory.MakeRoom(1);
Room* r2 = factory.MakeRoom(2);
Door* aDoor = factory.MakeDoor(r1, r2);

aMaze->AddRoom(r1);
aMaze->AddRoom(r2);

r1->SetSide(North, factory.MakeWall());
r1->SetSide(East, aDoor);
r1->SetSide(South, factory.MakeWall());
r1->SetSide(West, factory.MakeWall());

r2->SetSide(North, factory.MakeWall());
r2->SetSide(East, factory.MakeWall());
r2->SetSide(South, factory.MakeWall());
r2->SetSide(West, aDoor);

return aMaze;
}
```

- Note that MazeFactory is not an abstract class; thus it acts as both the AbstractFactory and the ConcreteFactory.

Sample 2
See the Abstract Factory in android SDK integration project.

Known Uses (已知应用)

...

Related Patterns (相关模式)

- **Abstract Factory** classes are often implemented with **Factory Methods**
- **Abstract Factory** can also be implemented using **Prototype**.
- A concrete factory is often a **Singleton**.