SHRI RAMDEOBABA COLLEGE OF ENGINEERING AND MANAGEMENT, NAGPUR - 440013

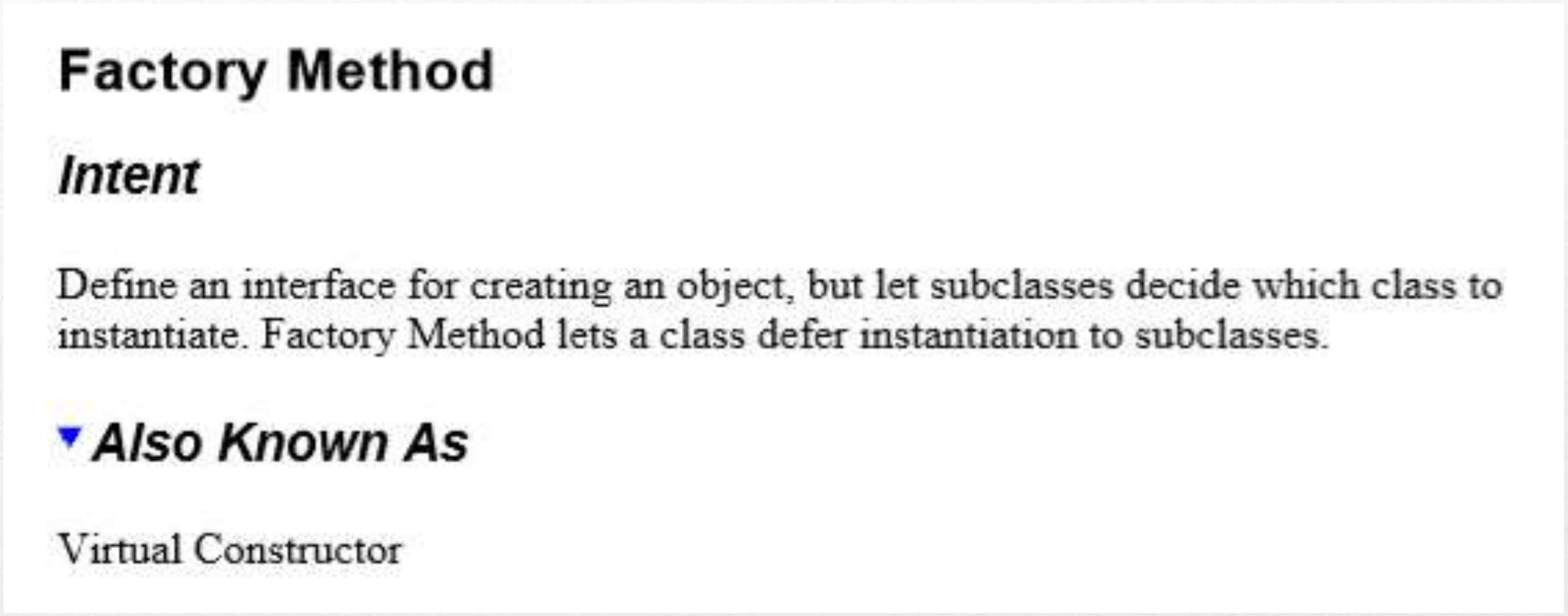
DESIGN PATTERNS (CST355-4)

V SEMESTER SECTION A, B

COURSE COORDINATOR: RINA DAMDOO

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACTORY METHOD DESIGN PATTERNS



Factory Method

Intent

Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.

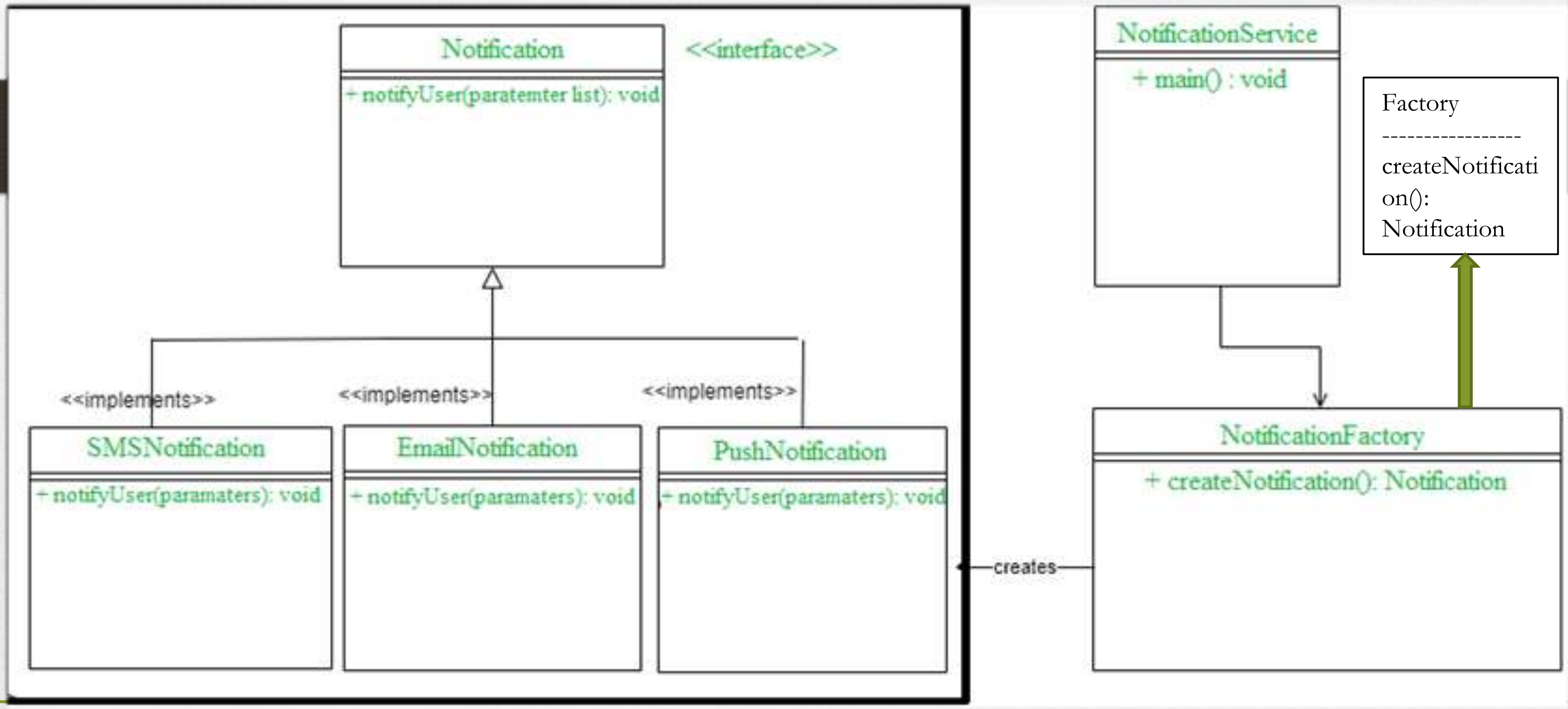
▼ *Also Known As*

Virtual Constructor

FACTORY METHOD EXAMPLE

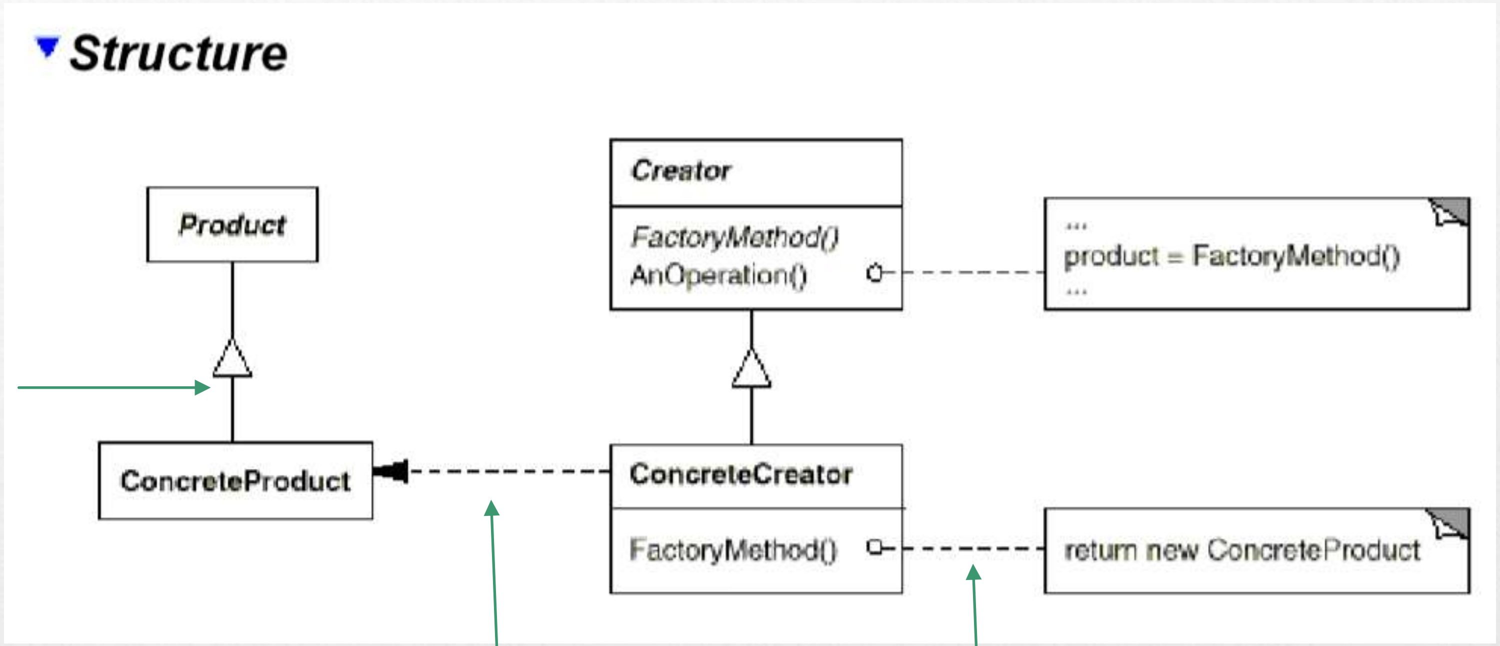
Problem Statement :

Implement a notification service through email, SMS, and push notification. Implement this with the help of factory method design pattern.


Notification
<<interface>>
+ notifyUser(paratemter list): void
<<implements>>
<<implements>>
<<implements>>
SMSNotification
+ notifyUser(paramaters): void
EmailNotification
+ notifyUser(paramaters): void
PushNotification
+ notifyUser(paramaters): void
creates
NotificationService
+ main() : void
Factory
createNotification(): Notification
NotificationFactory
+ createNotification(): Notification

UML class diagram

FACTORY METHOD DESIGN PATTERNS

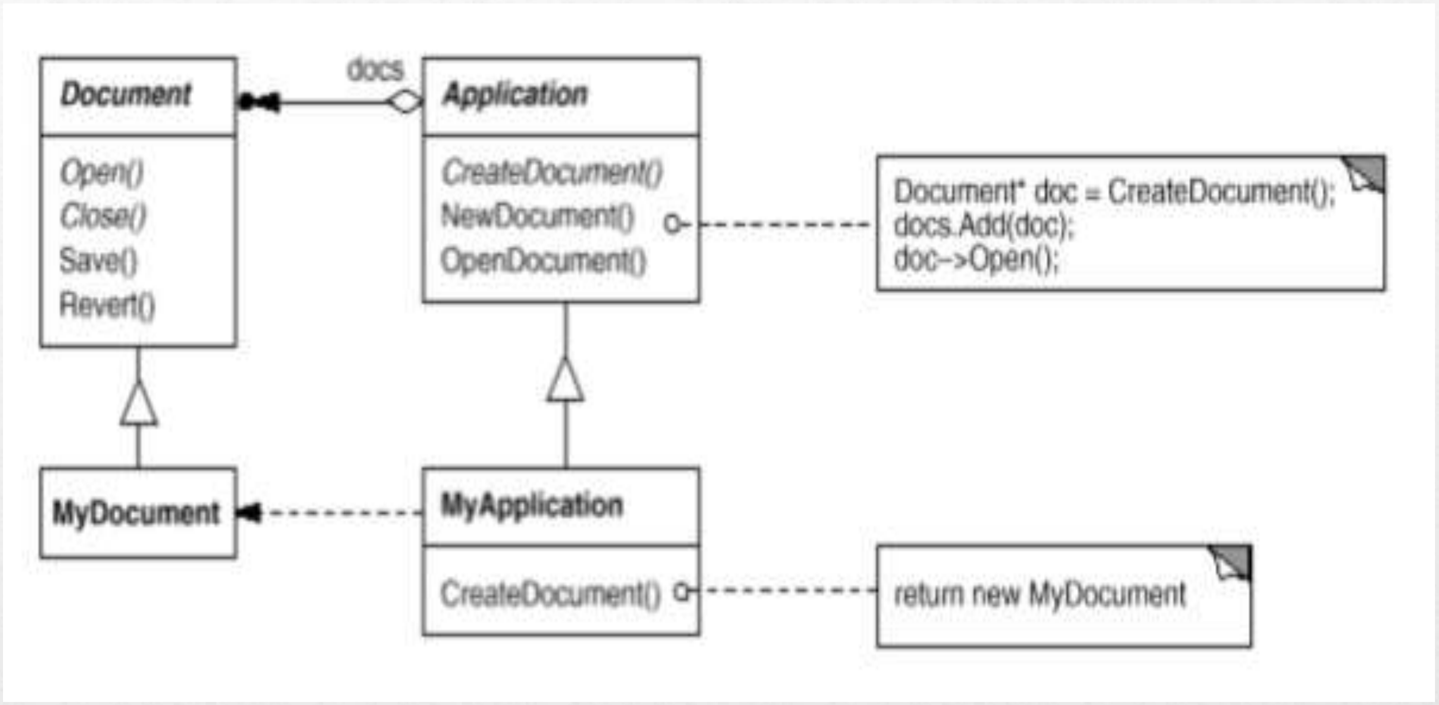


ISA

creates

Pseudocode annotation

MOTIVATION


docs
Document
Open()
Close()
Save()
Revert()
Application
CreateDocument()
NewDocument()
OpenDocument()
Document* doc = CreateDocument();
docs.Add(doc);
doc->Open();
MyDocument
MyApplication
CreateDocument()
return new MyDocument

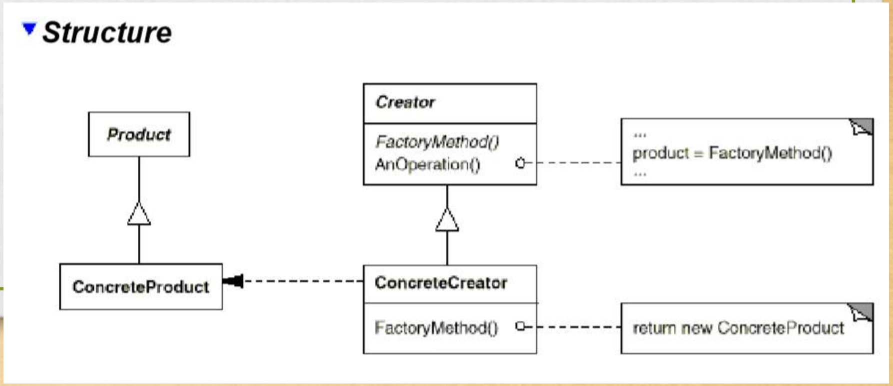
APPLICABILITY

Use the Factory Method pattern when:

- a class can't anticipate the class of objects it must create.
- a class wants its subclasses to specify the objects it creates.
- classes delegate responsibility to one of several helper subclasses, and you want to localize the knowledge of which helper subclass is the delegate.

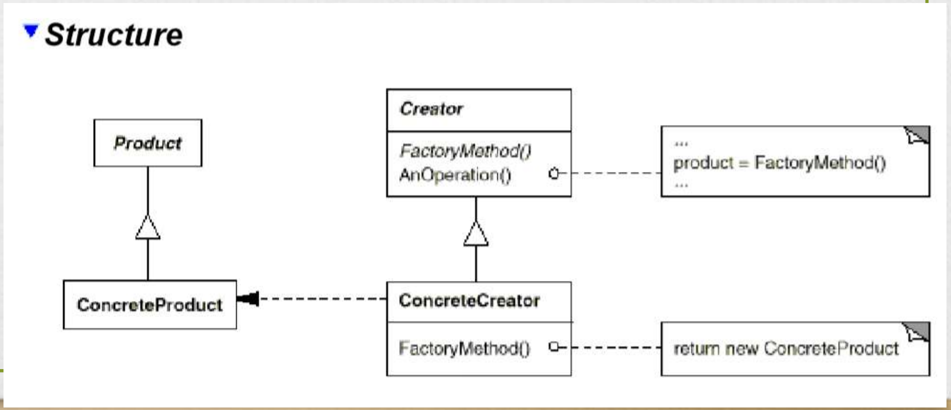
PARTICIPANTS

Participant	Responsibility
Product	Defines the interface of objects the factory method creates.
ConcreteProduct	Implements the Product interface.
Creator	Declares the factory method, which returns an object of type Product. Creator may also define a default implementation of the factory method that returns a default ConcreteProduct object.
ConcreteCreator	Overrides the factory method to return an instance of a ConcreteProduct.



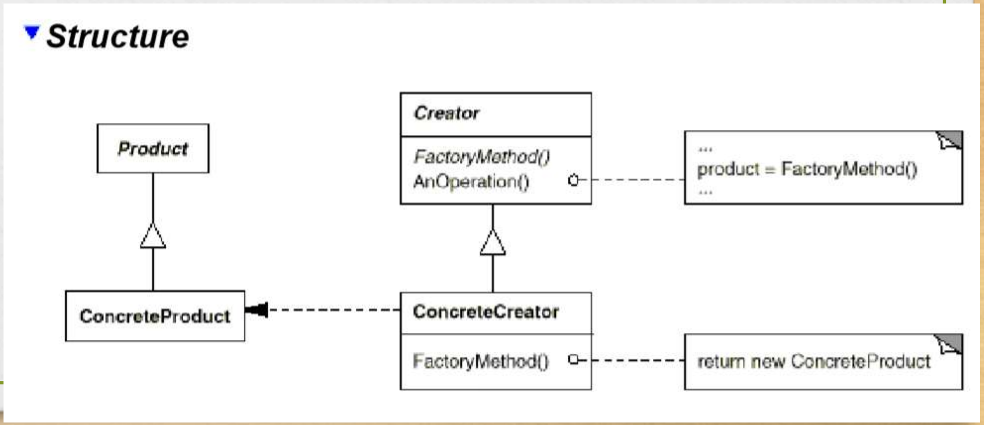
COLLABORATIONS

•**Creator** relies on its subclasses to define the factory method so that it returns an instance of the appropriate **ConcreteProduct**.



CONSEQUENCES

- Factory methods eliminate the need to bind application-specific classes into your code. The code only deals with the **Product** interface; therefore it can work with any user defined **ConcreteProduct** classes.
- A potential disadvantage of factory methods is that clients might have to subclass the Creator class just to create a particular ConcreteProduct object.



CONSEQUENCES

Here are two additional consequences of the Factory Method pattern:

1. **Provides hooks for subclasses.**

Creating objects inside a class with a factory method is always more flexible than creating an object directly. Factory Method gives subclasses a hook for providing an extended version of an object.

2. Connects parallel class hierarchies.

In the examples we've considered so far, the factory method is only called by Creators. But this doesn't have to be the case; clients can find factory methods useful, especially in the case of parallel class hierarchies (in Abstract Factory Pattern).

IMPLEMENTATION

Implementation Consider the following issues when applying the Factory Method pattern:

1. **Two major varieties.**

   (1) the case when the Creator class is an abstract class and does not provide an implementation for the factory method it declares

   (2) the case when the Creator is a concrete class and provides a default implementation for the factory method. It's also possible to have an abstract class that defines a default implementation, but this is less common.

2. **Parameterized factory methods.**

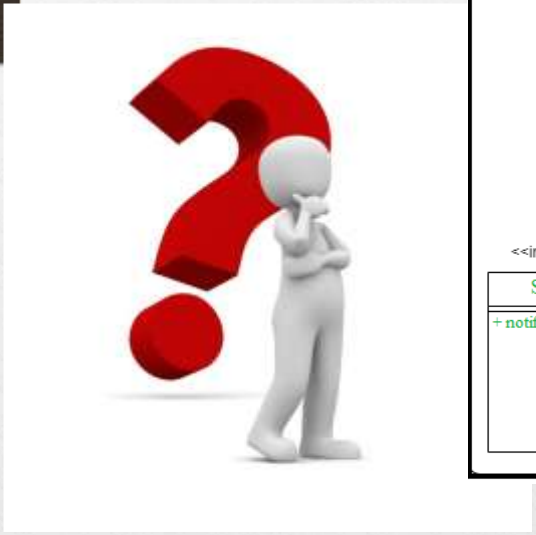
   Another variation on the pattern lets the factory method create multiple kinds of products. The factory method takes a parameter that identifies the kind of object to create. All objects the factory method creates will share the Product interface.

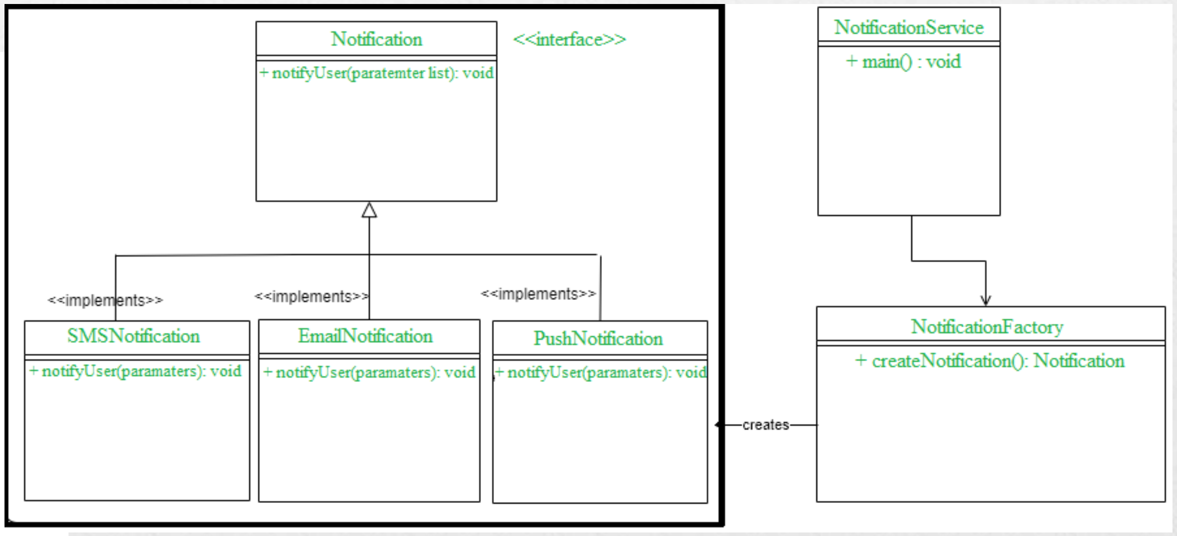
KNOWN USES & RELATED PATTERNS

- Factory methods encompass toolkits and frameworks.

- **Abstract Factory** is often implemented with factory methods.
- Factory methods are usually called within **Template Methods.**
- **Prototypes** don't require subclassing Creator. However, they often require an Initialize operation on the Product class. Creator uses Initialize to initialize the object. Factory Method doesn't require such an operation.

Does the Factory Method pattern violate the Open/Closed principle?





- ✓ Factory Method actually works well with the Open/Closed principle if done correctly.
- ✓ The Factory Method pattern creates a different type of object based on specified parameters.
- ✓ However, if you create a new class and then want the Factory Method to create a new object of that type you would have to change the Factory Method. Instead new class hierarchy can be created