**Abstract Factory Kalıbı(Pattern)**

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

**Amaç**: Birbirleriyle ilgili ya da bağımlı nesne ailelerini, nesnelerin somut sınıflarını belirtmeden yaratmak için bir arayüz sağla.

**Factory method Abstract Factory pattern’in özel bir halidir, onun parçasıdır.**

**PROBLEM**

**\*\*\*Abstract Factory**, tek nesne yerine, nesne ailesi oluşturmak üzere kullanılan bir kalıptır.

• İsimlendirmesi, **Factory Method** kalıbı da düşünüldüğünde problemlidir.

• **Abstract Factory** bir sınıftır, **Factory Method** ise bir metottur.

• **Factory Method** tek bir nesnenin, **Abstract Factory** birden fazla nesnenin yaratılmasını soyutlar.

• Bu amaçla **Abstract Factory** birden fazla **Factory Method**’a sahiptir.

**Object Family ( Nesne Ailesi)**

Kalıbın açıklamasında geçen **birbirleriyle ilgili ya da birbirlerine bağımlı nesne ailesi** (**families of related or dependent objects**) **birbirleriyle ilgili ürün ailesi** (**a family of related product**) ile kastedilen **birlikte kullanılacak şekilde tasarlanmış ve bunun uygulandığı nesneler**dir (a family of related product objects is designed to be used together, and you need to enforce this constraint.)

• **Abstract Factory**, birlikte kullanılacak nesneleri yaratmaktan sorumlu bir sınıftır.

Örneğin bir **Proje** nesnesi yaratıldığında muhtemelen ondaki farklı işler olarak **Task** nesneleri, tekrarlanan **Meeting** , **Report** vs. nesneleri beraber kullanılırlar.

• Ya da bir GUI (windowing) sisteminde **Button**, **Label**, **List**, **Scrollbar**, vs. hep beraber yaratılan nesnelerdir

**Solution**

**Factory Method** tek bir nesneyi yaratmaktan sorumludur.

• Yaratılacak nesne bir değil birden çok ise yapılacak şey birden fazla **Factory Method** kullanmaktır.

• Bu durumda farklı **Factory Method**’lar farklı arayüzlerde bulunur.

• Çünkü nesneler arasında bir ilgi ya da bağımlılık yoktur.

• Benzer şekilde her bir nesne için arayüzü gerçekleştiren dolayısıyla da **Factory Method**’unu ezen sınıflar oluşturulur.

* **Elinizdeki nesneler, bir nesne ailesi oluşturuyorlarsa yani bu nesneler**

**birbirleriyle ilgili nesnelerse, her birisi için ayrı ayrı oluşturulan ve farklı**

**sınıflara konan Factory Method’larının bir sınıfta toplanmaları makuldür.**

* **Bu durum, Factory Method’larını bir araya toplayan sınıfın birlikteliğini düşürmez!**
* **İşte bu sınıf Abstract Factory’dir.**
* **Abstract Factory bir sınıftır, Factory Method ise bir metottur.**
* **Abstract Factory, birden fazla Factory Method kullanarak oluşturulur.**

**SONUÇLAR**

* **Abstract Factory** nesne aileleri ve onları üreten nesnelerle ilgili sistematik bir isimlendirme sağlar.
* Sisteme yeni bir nesnenin (**ScrollBar**) katılması durumunda hem **Abstract Factory**’e hem de alt sınıf olan gerçekleştirmelerine (**OSXFactory** ve **WinFactory**) gerekli factory metotlarının (**createScrollBar()**) eklenmesi gereklidir.
* Sisteme yeni bir aile (**XWindows**) eklenmesi durumunda ise her nesnenin o aileye ait alt sınıfları (**XWindowsButton**, **XWindowsList**, …) ile üretici alt sınıfının (**XWindowsFactory**) sisteme eklenmesi gereklidir

**KULLANIMLAR**

* Java API’lerinde kullanım örnekleri:
  + **javax.xml.parsers.DocumentBuilderFactory**
    - **abstract** olan **newDocumentBuilder()** metodu
  + **javax.xml.parsers.SAXParserFactory** 
    - **abstract** olan **newSAXParser()** metotları
  + **javax.xml.transform.TransformerFactory**
    - **abstract** olan **newTransformer()** metotları
  + **javax.xml.xpath.XPathFactory** 
    - **abstract** olan **newXPath()** metodu

**DİĞER KALIPLARLA İLİŞKİSİ.**

* **Abstract Factory***,* **Factory Method**’u kullanılır.
* **Factory Method** bir metot iken, **Abstract Factory** bir sınıftır, dolayısıyla **Abstract Factory** bir üst soyutlama yapısı sağlar.
* **Factory Method** bir nesne için bir yaratma yöntemi sağlarken, **Abstract Factory** birden fazla ilgili nesne, nesne ailesi için birden fazla yaratma yöntemi sağlar.
* **Abstract Factory’de Factory Method yanında Builder ve Prototype’da kullanabilir.**

**Önemli Not :**

**Nerede yaratacağımız ve nasıl yaratacağımız konusu yaratımsal(creational) kalıplarda iki ayrı sorunsal.**

**Nerede yaratacağımız sorusunun cevabı Abstract Factroy, Factory method. Bu şunu sağlıyor istemcileri yaratma yükünden uzaklaştırıyor.**

**Nesneyi Nasıl yaratacağımız sorusunun cevabı ise Builder ve Prototype kalıplarındadır.**

UYGULAMA.

Aşağıda verilen problemdeki nesnelerin yaratılması problemini **Abstract Factory** kalıbını kullanarak çözün.

• İsimleri HotelA, HotelB ve HotelC olan 3 hotel aynı şirket tarafından satın alınmış olup ve BT altyapılarının birleştirilmesine çalışılmaktadır. Her otelin ayrı müşteri, oda, rezervasyon ve ödeme nesnesi vardır.

• Bu nesnelerin oluşturulmasını bir yapı altında toplamak için **Abstract Factory** nasıl kullanırdınız? • Çözümünüzü modelleyip kodlayın.