**Nesne Yaratma Problemi**

* Nesne-merkezli programlamada en temel ve yaygın iş nesne yaratmadır.
* Kötü oluşturulmuş bir modelde nesneler zaten gelişigüzellikten kaynaklanan karmaşıklığa sahiptir.
* İyi oluşturulmuş bir modelde nesneler, aralarındaki düzenli ilişkiler, tanımlı durumları (state) ve tekrar kullanılabilirlik ( reusability) gibi kaygılardan dolayı anlaşılır bir karmaşığa sahiptirler.
* Bu anlamda genel olarak iş nesneleri ile kontrol nesneleri daha karmaşık olma eğilimindedirler.
* Dolayısıyla nesne yaratma, sadece bir kurucu(constructor) çağrısına indirgenemeyecek kadar karmaşıktır.
* **Çoğu zaman bir nesneyi oluşturmak, o nesneyi kullanmaktan daha zordur.**
* Çünkü nesnenin sağladığı basit ve anlaşılır bir interface’n arkasındaki mekanizmayı kurmak zordur.
* Nesne yaratmada temel iki problem şunlardır:
  + **Nesnenin yaratılacağı yer**
    - Nesneler belirli yerlerde yaratılmalıdır.
* **Nesnenin nasıl yaratılacağı**
  + Nesnelerin nasıl yaratılacağını belirleyen şey genelde nesnelerin durumları (state) ve diğer nesnelerle olan ilişkileridir.

**Nesne Nerede Yaratılmalı ?**

* Nesne yaratmak, bir iş süreci ya da bir iş kuralı gibi özel bir sorumluluktur ve sistemde bu sorumluluk için oluşturulmuş özel yerler olmalıdır.
* Burada nesne ile kastedilen genel olarak iş nesneleridir.(Business objects)
* Çünkü framework nesneleri daha basittirler yaratılması karmaşık olan genelde bizim modellediğimiz nesnelerdir, iş nesneleridir.

**Nesne Yaratma Ters Kalıpları**

* **Nesne yaratan kodlar sistemde her tarafa yayılır,**
  + Bu yayılma gittikçe daha çok copy-paste ile yapılır hale gelir.
* **İstemcilerin nesnelere olan bağımlılıkları artar.**
  + İstemciler, nesnelerin yaratılma şekillerindeki değişikliklerden etkilenirler.
* **Nesneler hatalı bir şekilde, yanlış ilişki ve durumlarda yaratılır,**
  + Zor bulunan buglar ortaya çıkar.

**Problem Nedir?**

* Kurucuların (constructor) çok parametre alması normaldir.
* Bir sınıfın bir çok karmaşık constructor olması da normaldir.
* Normal olmayan, bu karmaşık kodları çağıdan kod parçalarının her tarafa yayılmasıdır.
* Öte taraftan constructlarda tasarımsızlıktan kaynaklanan gereksiz bir karmaşıklık vardır.

**Çözüm**

* Nesnelerin yaratılması basitleştirilmeli,
* Nesnelerin yaratılması tekrar kullanılabilir olmalı,
* İstemcilerin, nesnelerin yaratılmasına olan bağımlılıkları asgari seviyeye indirilmeli.

Bu üç durumu sağlamak amacıyla aşağıdaki yaklaşımlara önem verilmelidir:

* Nesnelerin yaratılması kodun diğer kısımlarından soyutlanmalıdır; nesneler, sadece ve sadece belirli yerlerde yaratılmalı, bunun dışında “new” kullanımı engellenmelidir.
* Nesneler, sadece onların durumlarını ve diğer nesneler ile ilişkilerini çok iyi bilenler tarafından yaratılmalıdır.

**YARATIMSAL KALIPLAR ( Creational Pattern)**

* Yaratımsal kalıplar nesnelerin yaratılmalı ile ilgidir.
* Yaratımsal kalıplar, nesne yaratmayı soyutlar ve nesne yaratma kodlarının etrafa dağılmasını önlerler.
* En basit ve anlaşılır kalıplar
* İhtiyacı hissedilen kalıplardır.

**Yaratımsal kalıplar 5 tanedir:**

* **Singleton**: Bir sınıftan sadece bir tane nesne yaratmak.
* **Factory Method**: Nesne yaratmayı soyutlamak.
* **Abstract Factory**: Nesne ailesi yaratmayı soyutlamak.
* **Builder**: Karmaşık nesne yaratma sürecini kurgulamak.
* **Prototype**: Bir örnek nesneden kopyalamayla yeni nesneler türetmek.,

**Son ikisi Nasıl , 2.ve 3. Nerede sorularına cevap verebiliyor.**

**Singleton biraz anti-pattern olarak da değerlendirilebiliyor.**

