Spring"in Kullandığı Tasarım Desenleri

Creational Patterns (Yaratımsal Kalıplar): Bu tasarım deseni nesneleri doğrudan new operatörü kullanarak oluşturmak yerine nesne oluşturma mantığını gizleyerek sınıflardan nesne oluşturmaya alternatif çözümler sunar. Bu program akışında hangi nesneye ihtiyaç varsa onu oluşturmada esneklik ve kolaylık sağlar.

Oluşturucu Tasarım Şablonları : Factory , Factory Method , Abstract Factory , Singleton, Builder, Prototype Object Pool.

Structural Patterns (Yapısal Kalıplar): Bu tasarım deseni nesneler arasındaki ilişkinin yapısını düzenlemek için çözümler sunar.

YapısalTasarım Şablonları : Class Adapter ,Object Adapter ,Bridge ,Facade ,Composite ,Decorator, Virtual Proxy ,Dinamic Proxy, Protection Proxy, Remote Proxy, Flyweight

Behavioral Patterns (Davranışsal Kalıplar): Bu tasarım deseni çalışma zamanında nesneler arasındaki davranışlar için çözümler sunar.

Davranışsal Tasarım Şablonları : Command ,Iterator ,Memento ,State ,Observer ,Strategy ,Chain Of Responsibility ,Mediator ,Visitor ,Template Method ,Interpreter

Java EE Tasarım Şablonları : Model View Controller ,Front Controller ,Data Access Object ,Business Delegate ,Service Locator ,Intercepting Filter ,Business Objecta

Diğer Tasarım Şablonları : DataMApper ,RequestMapper ,ResponseMapper ,Active Record ,Message Channel ,Message Router , Registry , Null Object , Dependeny Injection

Yazılım esnasında tekrar eden sorunları çözmek için kullanılan ve tekrar kullanılabilir yapıda kod yazılımını destekleyen, bir ya da birden fazla sınıftan oluşmuş modül ve program parçalarına tasarım şablonu (design pa□ern) ismi verilir.Tasarım kalıbı adı üstünde bir tasarımı tasarlarken tasarımın kullanılabilirliği,test edilebilirliği,verimliliği,kalitesi vb birçok şeye dikkat edilmesi gerekilir.Yazılım dünyasında geçmişten bugüne oluşan bu kalıplar ,yazılımcıların çözdükleri problerin benzerlikleriyle ve sık tekrar etmesiyle ihtiyaç duydukları ve bu çözümleri belli kalıplar haline getirerek bulduğu kalıplara tasarım deseni diyoruz.Bunlar çözdükleri problemlere yaptıkları işlemlere göre geçmişten günümüze genel olarak isimlendirilerek bugünlü hallerini almışlar.Yaşanılan sorunları çözen kalıplar olarak ihtiyaca ve tasarıma göre kullanılırlar.

Creational Patterns

1.1.Tek Nesne (Singleton) Tasarım Deseni Bu tasarım deseninde, bir sınıfın sistem içinde yalnızca bir tane nesnesi oluşturulabilir. Tek bir arayüz sunularak, bu nesneye yalnızca buradan erişim sağlanabilir. Bu desen kullanılarak, sistem içinde değeri değişmeyen, genel değişkenler bu oluşturulan tek nesneye konulabilir. Sistemde tek nesne yaratılabilme, **statik** değişken ve yordamlar sayesinde olur.

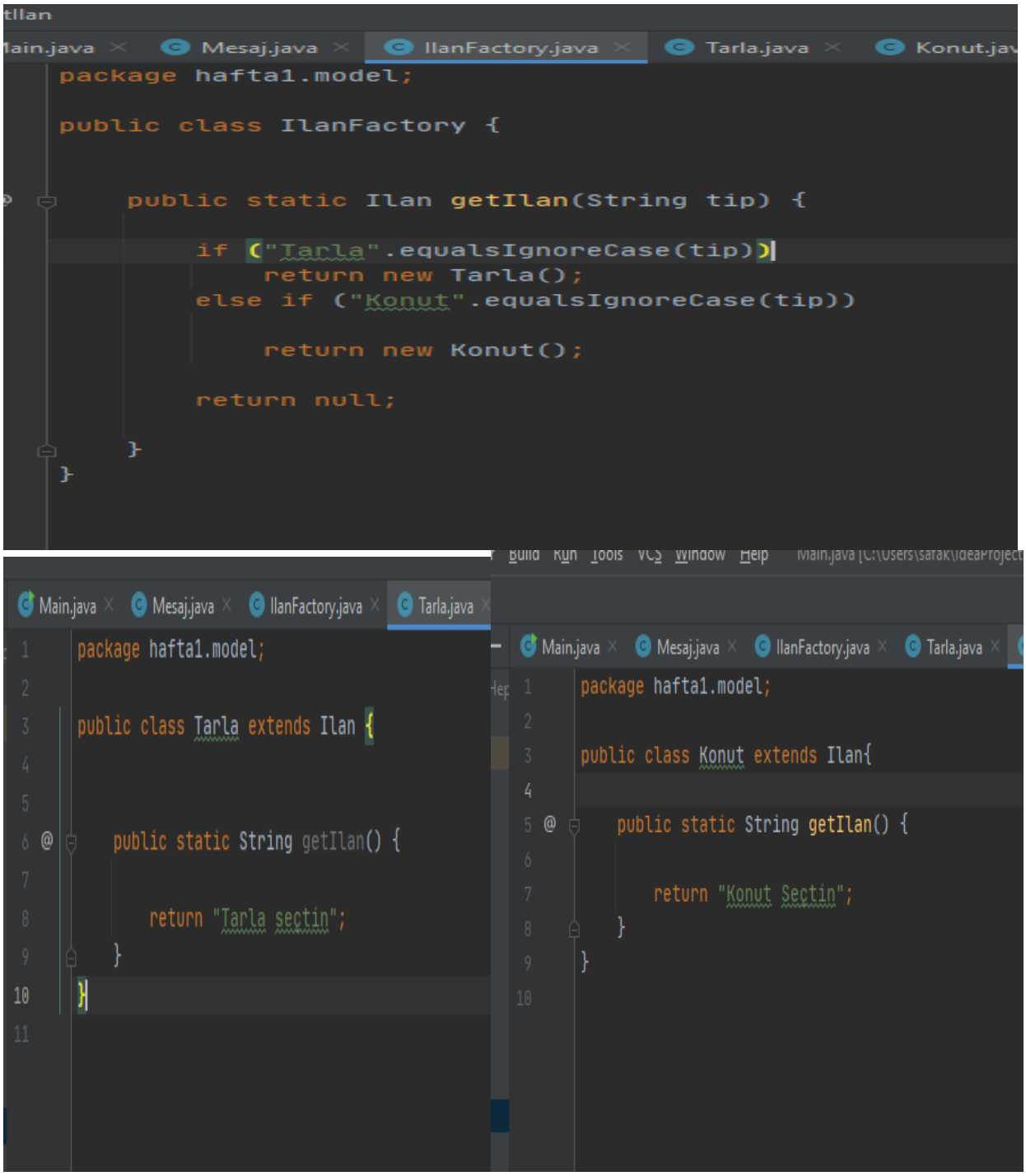
```
class Singleton
{
        private static Singleton instance;
        private Singleton()
        {
                ...
        }
        public static synchronized Singleton getInstance()
        {
                if (instance == null)
                        instance = new Singleton();
                return instance;
        }
        ...
        public void doSomething()
        {
...
    }
  }
```

Fabrika Yordam (Factory Method) Tasarım Deseni

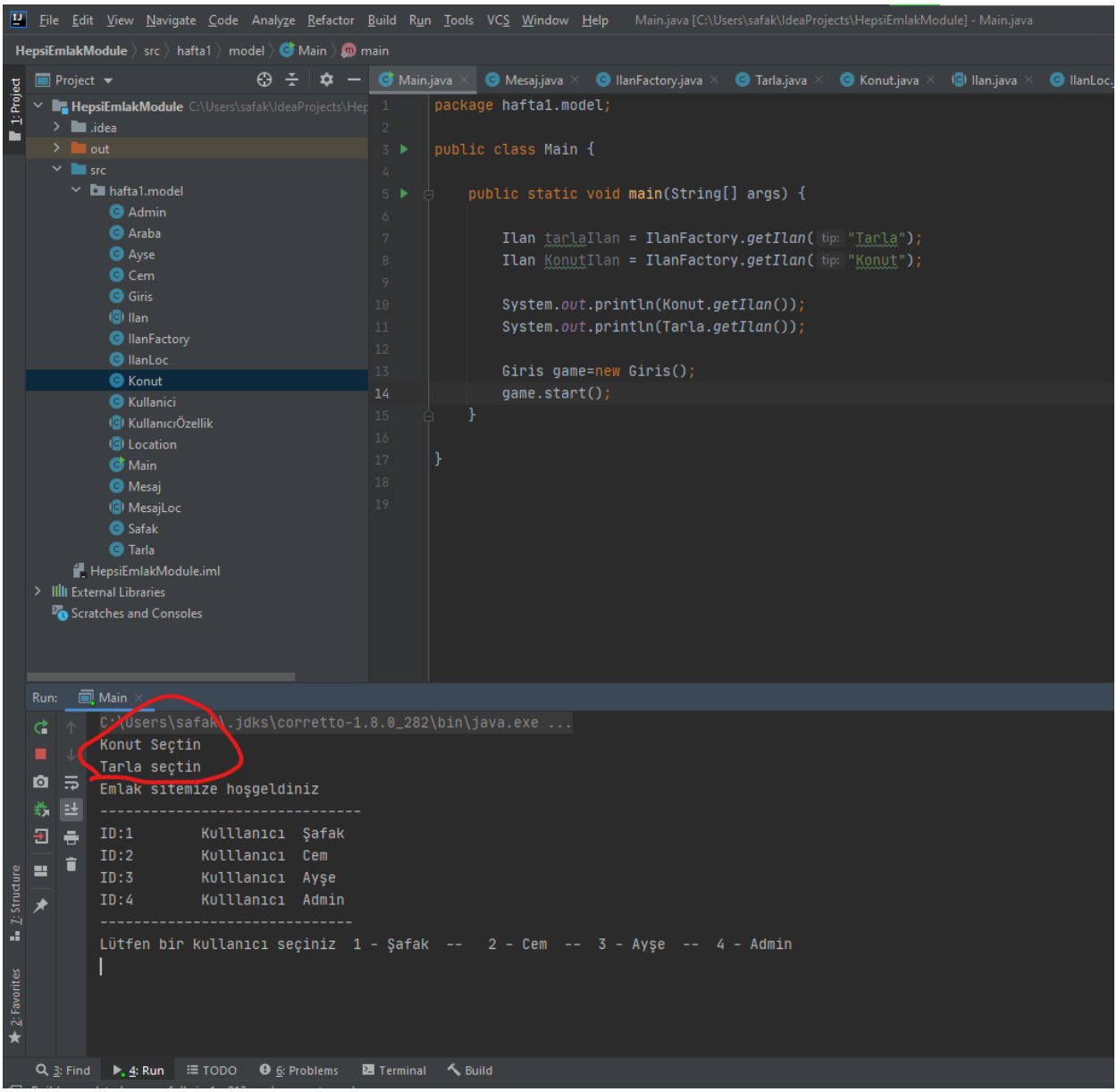
Fabrika yordam tasarım deseni, nesne yaratma sorumluluğunun bir yordama verilmesidir. Yaratılan nesne, bir sınıf hiyerarşisindeki alt sınıflardan biridir. Hangi alt sınıfın yaratılacağı kararı fabrika yordam içinde verilir. Bu yordam ile belirli bir sınıf hiyerarşisindeki alt nesnelerden birinin yaratma sorumluluğu belirli bir arayüze verilerek sistemden soyutlanmış olur. Böylece nesneleri yaratma kodlarında, kod tekrarları önlenmiş olur. Sistem içinde sınıfların yaratılacağı yer tek olduğu için, ilgili mantıklar tek bir yerde toplanabilir.

Bu kalıbı tasarlamak için bir fabrika sınıfına ihtiyaç duymaktayız.

İlanımızın Tarla Tipinde ve Konut Tipinde oluşturabildiğimizi varsayarak böyle şartlandırarak nesne oluşumunda bağımlılığı azaltmış oluyoruz ve new ile yeni bir nesne üretmemiş oluyoruz.



Tarla ve konut classlarımız şekildeki özelleştirerek ayırıyoruz.



Ve sonunda Ilan sınıfımızdan neşen oluşturup Fabrikamızdan yardım alarak new kullanmadan istediğimiz nesneyi istediğimiz şekilde oluşturuyoruz.