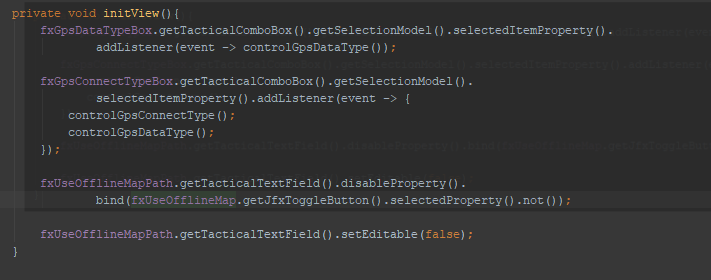
Projede bir Configuration ekranı yapmak için çalışmalara başlanıldı. Bu ekran üzerinde;

* Urn hane sayısı, gönderici veya alıcı geri bildirim portunun açık veya kapalı olması, MulticastPortunun kaç olarak ayarlanacağı gibi özel alanların ayarlanması sağlanacaktır. Ayrıca bu alanlar gruplandırılarak gösterilecektir. Bu gruplandırma Gps, Sync, Mobile, Scale ve Map şeklinde olacaktır.

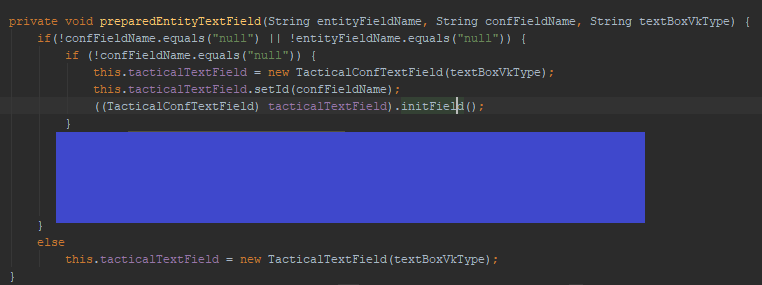
Gereksinimlere uygun olarak fxml’de bir JFXTabPane kullanıldı. Bu TabPane’e 5 adet tab eklendi. Tab’ların text özellikleri sırasıyla Gps, Sync, Mobile, Scale ve Map şeklinde ayarlandı. Tab içlerinde Hbox kullanılarak alanların alt alta olması sağlandı.

Bazı alanlar, diğer alanlara bağımlı olduğu için bunun için çalışmalar yapıldı. Örneğin; Gps açık ise, Gps Connect Type ve Data Type alanlarının disable özelliği true olacak şekilde ayarlanmalı, Gps kapalı ise de bu alnların disable özelliği false olarak ayarlanmalı. Böylece tüm ayarların kontrollü bir şekilde düzenlemesi sağlanacaktır.

Aşağıda verilen kod örneğindeki gibi listener eklenerek alanların kontrollü bir şekilde düzenlenmesi sağlanmıştır. Ayrıca bind kullanılarakda kontroller sürdürülmüştür.

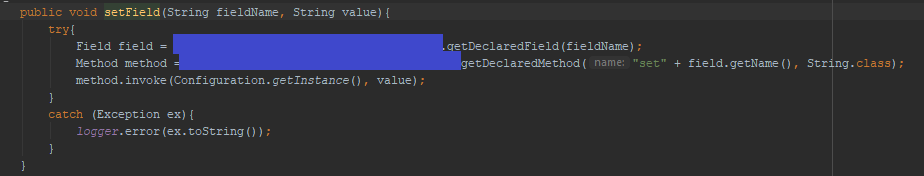


Configuration ekranı için kullanılan componentler (TextField, ToogleButton) özelleştirilmiştir. Bu özelleştirme işleminin sebebi, update gibi işlevler kurtulmak içindir. Özelleştirilmiş component’lerin her biri için fieldname property’si tanımlanmıştır. Bu property’nin değeri componenet’e id olarak tanımlanmıştır. Bu sayede component’e hangi alan üzerinden işlem yapması gerektiği bildirilmektedir.



Yukarıda ki kod parçacığında görüldü üzere initField metodu çağırılmıştır. InitField metodunda 3 adet işlev çağırılmıştır. Birinci işlevde compnenet’e id olarak bildirilen değer, local bir değişkene atanmıştır. Üçüncü işlevde componentin türüne göre değer atanması yapılmıştır. Bu değer ataması örneğin text field componenet’i için setText işlevi kullanılması ile sağlamıştır. İkinci işlevde ise yine componenetin türüne göre bir listener eklenmiştir. Örneğin text field component’i için focused property’e listener eklenmiştir, combobox component’i için ise selectedItem property’e listener eklenmiştir.

Eklenilen listener içeriğinde bir setField işlevi çağırılmıştır. Bu işlev için “Reflection” yapısı araştırılmış ve işlemler bu yapıyı kullanarak gerçekleştirilmiş. Bu işlemlere aşağıda yer verilmiştir.

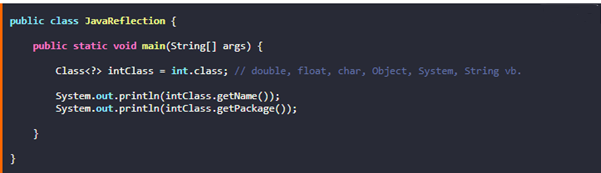


**Reflection Kullanımı**

Java.lang paketinde yer alan Class sınıfı, sınıf ile ilgili bilgi almak için **getName, getSimpleName, getModifiers, getPackage, getSuperclass, getInterfaces, getConstructors, getMethods, getFields, getAnnotations** gibi metotlara sahiptir.

Metotlarin dönüş değeri alınan bilgiye göre değişiklik göstermektedir.

Örnein sınıf içerisindeki metotları almak için kullanılan getMethods değerinin dönüş değeri Method türünden bir dizidir.



Method sınıfı kullanılarak metotlar hakkında bilgi alınabilir.

Reflection yapısı ile method çalıştırılmak istenildiğinde **Method** sınıfında yer alan **invoke** metodu kullanılabilir.

Reflection yapısı ile ayrıca sınıf içerisinde yer alan private, protected gibi erişim imkanı olmayan alanlara da erişim sağlanabilir.

Private, protected metotlara erişmek için getDeclaredMethods metodu kullanılır.

Ayrıca bu private, protected metotlar getDeclaredMethod ve setAccessible metodu kullanılarak çalıştırılabilmektedir.

**Configuration Sınıfı İçin Singleton Tasarım Deseni Kullanımı**

Configuration ekranında özelleştirilmiş componenetler sayesinde bir componenetin değeri değiştiği zaman (text field’ farklı bir değer girildiğinde, combobox’dan farklı bir değer seçildiğinde) direk olarak o componenete bağlı field değeride değişmekteydi. Bu durumda eğer kullanıcı bir değişiklik yapıp ardındanda cancel butonuna basarak çıkmak istediğinde (yapıldığı değişikliklerin kaydedilmesini istemediğinde) yinede değişiklikler özelleştirilmiş componentler vasıtasıyla kaydedilmeye neden olmaktaydı.

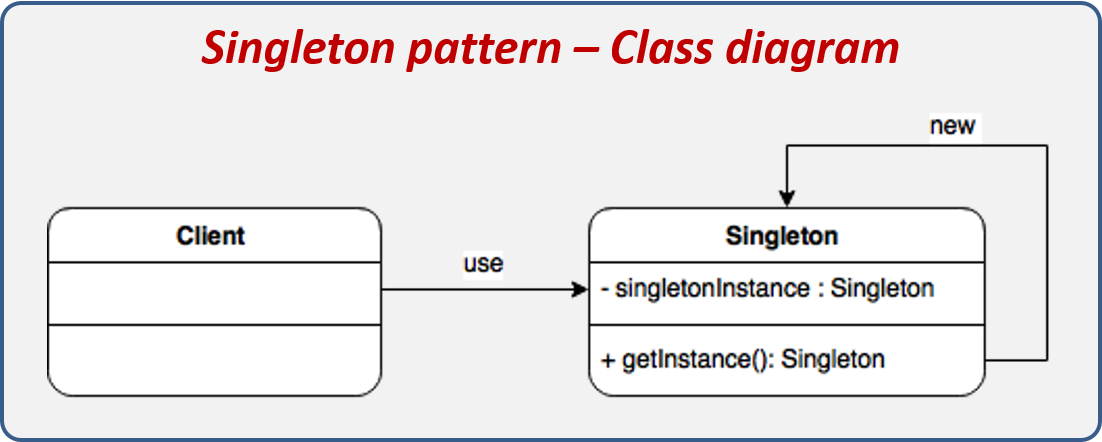
Yukarıda anlatılan durumun önüne geçmek için “Configuration” sınıfı için singleton tasarım deseninin kullanımı araştırılmış ve configuration sınıfında kullanılmıştır. Öncelikle Singleton tasarım deseni hakkında elde edilen bilgiler aşağıda yer verilmiştir. Daha sonra proje içerisinde nasıl kullanıldığı detaylı olarak anlatılmıştır.

1. **Singleton Tasarım Deseni**

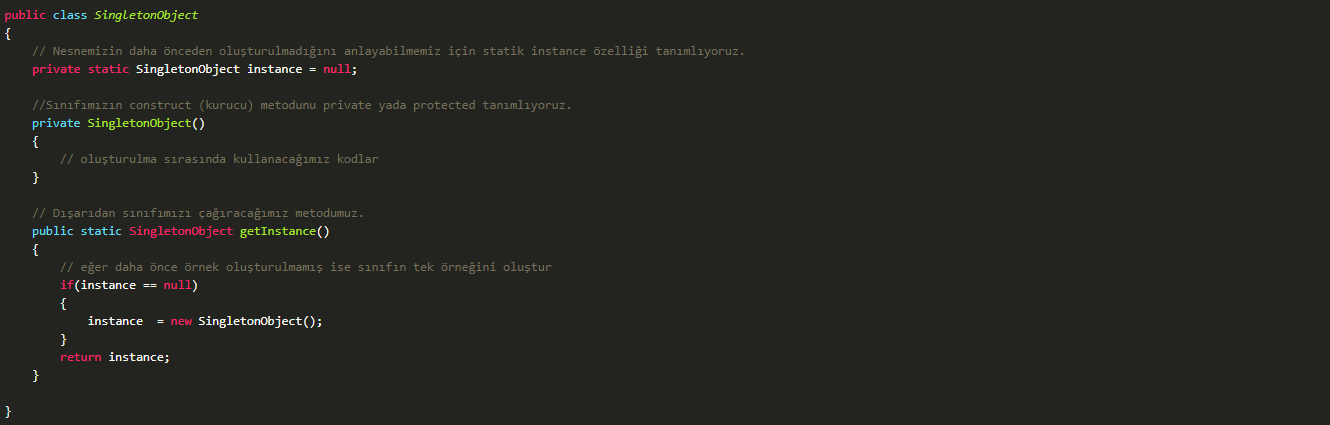
**Tanım:** Bu tasarım deseninin amacı uygulamanın yaşam süresince bir nesnenin bir kez oluşturulmasını sağlamaktır.

Creatinal patterns grubuna ait singleton tasarım deseni (Singleton design pattern) bir nesnenin application pool kapanana kadar bir kez üretilmesini kontrol altında tutar. Aynı zamanda bu nesne sınıf dışından da erişilebilinir olur. Bir sınıfın bir anda sadece bir örneğinin olması istenildiği zamanlarda kullanılır. Örneğin veritabanı uygulamalarında bir anda bir bağlantı nesnesinin olması sistem kaynaklarının verimli bir şekilde kullanılmasını sağlar.

**Uml Diyagramı:**

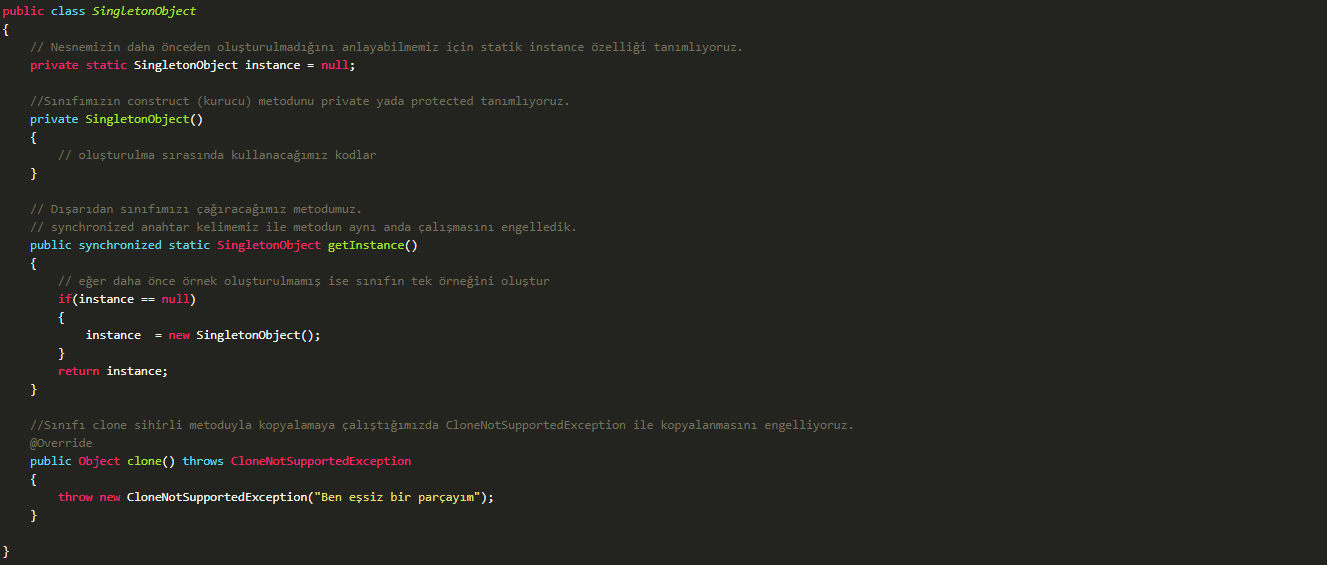
****

**Kullanımı:** Bu desenin kullanımı oldukça basittir. Singleton deseni uygulanacak sınıfın constructor(yapıcı) metodu private veya protected yapılarak nesne oluşturmasını yalnızca sınıfın içinde veya miras alan sınıf içinde tanımlanabilir.



Sınıftan yeni bir örnek oluşturamasak da var olan örneği kopyalayabiliriz, bu yüzden bunuda engellemek için **clone** metoduna düzenleme yapılmalıdır.

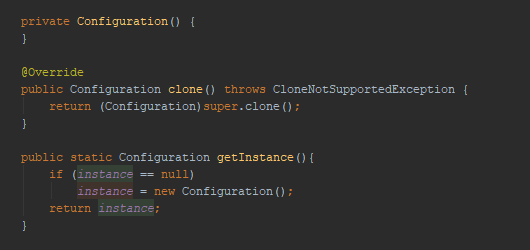
Son olarak JAVA'ya özgü thead (kanal) yapısı sayesinde birden fazla kanaldan aynı anda getInstance metoduna istek yapılır ve daha önce oluşturulmadıysa birden fazla örnek oluşturmuş olunur, bunu engellemek içinde metodun başına synchronized anahtar kelimesini eklenerek bu işlemleri sıraya konmuş olunur ve birden fazla oluşturulması engellenmiş olunur.



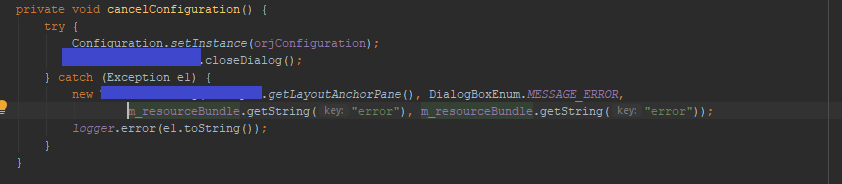
1. **Projede Singleton Tasarım Deseni Kullanımı**

Configuration sınıfına öncelikle Clonable sınıfı implements edilmiştir. Ardından clone metodu override edilerek eklenmiştir.

getInstance işlevide static olarak configuration sınıfına eklenmiştir. Bu işlevde öncelikle static olan ve Configuration türündeki instance adlı değişkenin null olup olmadığı kontrol edilmiştir. Eğer null ise yeni bir configuration nesnesi üretilmiştir.



Daha önce anlatılan cancel problem için orjinalConfiguration adlı bir değişken tanımlanmıştır. Bu değişkene Configuration sınıfından bir instance clone edilerek atanmıştır. Böylece configuration sınıfının saf hali bir değişkende tutulmuştur. Cancel butonuna tıklanıldığında configuration sınıfındaki inctance değeri orjinalConfiguration değeri kullanılarak setlenmiştir.



Save butonuna tıklanıldığındaki yapılacak olan işlemler de güncellenmiştir. Öncelikle orjinalConfiguration değişkenindeki değer güncellenmiştir. ve bir updateControl metodu eklenerek, eski değerler ile yeni değerler karşılaştırılmıştır. Eğer değişiklik var ise setMetodu çağırılmıştır. Hiçbir değişiklik yok ise işlem yapılmamıştır. UpdateControl metoduna aşağıda verilmiştir.

