**REUSE/RELEASE EQUİVALENCE**

Program modülleri paketler kullanılarak organize edilir. Paketler arasında sınıfların birbirlerini kullanmaları bağımlılıkları meydana getirir. Varsayalım ki elimizde iki adet paket var. Bunlar paket X ve paket Y olsun. Paket X’in içindeki bir sınıf, paket Y’nin içerisinde ki bir sınıf tarafından kullanılıyorsa, bu durumda paket Y paket X’e bağımlıdır deriz. Bu gibi bağımlılıklar olağandır. İstenilen durum, bu gibi bağımlılıkları kontrol edilebilir olmasıdır. Reuse/Release Equivalence prensibi bunu gerçekleştirmek için vardır.

Reuse/Release Equavelence prensibi paket bazında bir tasarım prensibidir.

Bir proje bünyesinde değişik modüllerden oluşan bir yazılım sistemi üzerinde çalıştığımızı düşünelim. Her modülden ayrı bir ekip sorumlu olsun. Üzerinde çalıştığımız modül için büyük bir ihtimalle diğer modülleri kullanmamız gerekecektir. Örneğin veri tabanı üzerinde işlem yapmamızı sağlayacak bir modülü başka bir ekip kullanmış olabilir. O ekip veri tabanı üzerindeki işlemler için gerekli tüm sınıfları başka bir paket içerisine yerleştirmiş olabilir. Bizim bu paket içindeki sınıfları tekrar kullanabilmemiz için belirli şartların yerine gelmesi gerekmektedir.

Bu şartlar:

* Veri tabanı ekibi, veri tabanı üzerinde işlem yapmak için kullanılan tüm sınıfları, bu sınıflar birbirleriyle ilişkili olduğu için aynı paket içine koymalıdır. Bu tekrar kullanımı kolaylaştırır.
* Tekrar kullanımı desteklemek için oluşturulan paketin bir versiyon numarasıyla yeni sürümünün oluşturulması gerekmektedir.
* Paket kullanıcıları paket üzerinde yapılan değişikliklerden haberdar edilmelidir. Onlar için kullanabilecekleri yeni bir paket sürümünün oluşturulması yanı sıra mevcut kodun kırılmasını önlemek için paketin eski versiyonlarının da paralel kullanıma açık tutulması gerekir. Sadece bu durumda paket kullanıcıları paket üzerinde yapılan değişiklerinden etkilenmeden eski versiyonlarla çalışmaya devam edebilirler. Zaman içinde yeni paket versiyonuna geçerek son değişiklikleri entegre ederler.

Tekrar kullanımı kolaylaştırmak için paket sürümlerinin oluşturulması şarttır. REP e göre tekrar kullanılabilirlik, sürüm ile doğru orantılıdır. Sürüm ne içeriyorsa, o tekrar kullanılabilir.

Yazdığımız kodlar yeniden kullanılabilir ve geliştirilebilir olmalıdır. Yeniden kullanım ve geliştirme işlemlerinde hatalarla karşılaşmamak için paketlerimiz birbirleri ile alakası olmayan sınıfların bir araya gelmesinden oluşmamalı. Sınıflar aynı amaca sahip ise bir arada olmalıdır. Yani farklı amaçlara sahip sınıfları tek bir modülede bir araya getirme ‘Reuse Release’ prensibine aykırı bir durumdur.

Özetle, ‘Reuse Release Equivalence’ prensibi paketlerin birden çok projede hatalarla karşılaşmadan kullanımını sağlar

**KAYNAKLAR:**

[1] Acar, Ö., Design Patterns Tasarım Şablonlarının Pratik Uygulanış Rehberi, Nisan 2016

[2] <https://medium.com/bilişim-hareketi/kalite-4-bileşenleri-kurgulamak-component-principles-32da67e46dd0>

[3] <https://www.ozgurozkok.com/emege-saygi-temiz-yaz-kodun-kokmasin/>

[4] <http://www.kurumsaljava.com/2009/12/09/reuse-release-equivalence-principle-rep-tekrar-kullanim-ve-surum-esitligi/>

[5] <https://dev.to/naomidennis/package-cohesion-reuse-release-equivalence-principle-3d28>

**Esra Fatma BİROL**

**Onur Karaman**