

REFACTORING

Devam etmekte olan kurumsal yazılım projelerinin büyük çoğunluğu, spaghetti kodlar ve güncellenmemiş mimarilerin sistem için yetersiz kalmasından dolayı;

- Zamanlama (Time to Market),
- Değişime cevap verme,
- Üretkenlik,
- Kalite,

noktalarında problemler yaşamaktadır. Bu problemleri aşmada izlenebilecek en iyi yollardan biri mevcut fonksiyonlitenin korunarak dizaynın geliştirilmesidir. Bu işlemin yapılabilmesi için kodlama seviyesinde çalışan personelin mevcut kodun açıklarını yakalayabilmesi ve bunları geliştirebilecek şablonlara hakim olması kritik önem taşımaktadır. Refactoring eğitimi sırasında;

- Mevcut fonksiyonliteni bozmadan dizaynı geliştirme,
- Anlaşılması kolay kodlar üretme,
- Değiştirmesi kolay kodlar üretme,
- Kod tekrarını azaltmak,
- Tekrar kullanılabilen kodlar üretme,
- Hata oranı düşük kodlar üretme,
- Yazılım projelerinde üretkenlik ve kalitenin artması

gibi konularda örnek uygulamalar üzerinde çalışılacaktır.

EĞİTİMİN İÇERİĞİ:

1. Gün

- Giriş
- Refactoring ve önemi
- Kodda “bad smell”in tespit edilmesi
- Çeşitli temel teknikler
 - Extract method
 - Move method
 - Move Field
 - Replace conditional with polymorphism

- Örnek uygulama

2. Gün

- Mevcut fonksiyonlitenin korunması

- Unit test ve mock kullanımı
- Örnek uygulama

- Basit dizayn prensipleri

- Refactoring teknik katalođu ve uygulama noktaları

- Örnek uygulama

- Geniş kapsamlı refactoringler

KİMLER KATILMALI:

Eđitim, mevcut projelerinde kalite ve üretkenlik sorunu yaşıyan, en az 6 ay programlama deneyimi olan, Design Pattern'ları anlayabilecek ve uygulayabilecek seviyede Object Oriented programlama deneyimi olan tüm yazılım uzmanlarına, yazılım mimarlarına, bilgi işlem yöneticilerine ve akademisyenlere yöneliktir.

Eđitim öncesi katılımcıların aşığıdaki kaynak ve toolları incelemeleri önerilmektedir;

www.junit.org

www.jmock.org

Design Patterns: Elements of Reusable Object-Oriented Software

Eclipse

Java 1.5