**ITERATOR PATTERN**

• **Intent**: Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.

• **Amaç**: Alt taraftaki yapısını ortaya çıkarmadan, torba nesnesinin elemanlarına sıralı bir erişim yöntemi sağla.

**PROBLEMS**

• Sıklıkla birden fazla nesneyi bünyesinde barındıran, torbalarla (collection, container) uğraşmamız gerekir.

• Torbaların yapısından bağımsız olarak torbadaki nesnelere tek tek ulaşmak isteriz.

• Torbalar dizi (array), list, set, map, vb. farklı yapılarda olabilir.

• Her torba için farklı bir erişim sistemi yerine hepsine tek bir şekilde ulaşmak istemci açısından tercih edilir.

• Bu durumlarda **Iterator** kalıbını kullanırız.

**SONUCLAR**

• **Iterator**, karmaşık ve uyumsuz torba arayüzlerini basit ve yeknesak hale indirger.

• Aynı anda birden fazla **Iterator** aynı torba üzerinde çalışabilir.

• Burada torbayı değiştirme söz konusu ise eş zamanlama problemleri çıkabilir.

**DİĞER KALIPLARLA İLİSKİ**

• Pek çok kalıp **Iterator’ü** kullanır.

• Parça-bütün ilişkisinin olduğu yerlerde **Composite** kalıbından **iteratör** alınıp parçalarına erişilebilir.

• **Observer** kalıbında konu nesne kendisine abone olanlara ulaşmak için **Iterator’ü** kullanır.

• **Mediator** kalıbında arabulucu nesne iletişimde olduğu nesneler için **Iterator’ü** kullanabilir

**KULLANIMLAR**

**• Dillerdeki ve kütüphanelerdeki torba (collection, container) yapılarından genelde iteratör nesneleri de olur.**

**• Java’da java.util paketinde, .NET’te System.Collections namespaceinde bir collection yapısı vardır.**

• Bu yapıda iteratörlar da vardır.

• Bundan dolayı bu kalıbı sıfırdan kurma ihityacı çok azdır.



**UYGULAMA**

Composite kalıbı için kurgulanan Proje üzerindeki Item nesnelerini almak için Iterator kalıbını

uygulayın.

• Bu amaçla ItemIterator nesnesini oluşturun.

