JAVA Eğitimleri

Visitor Örüntüsü

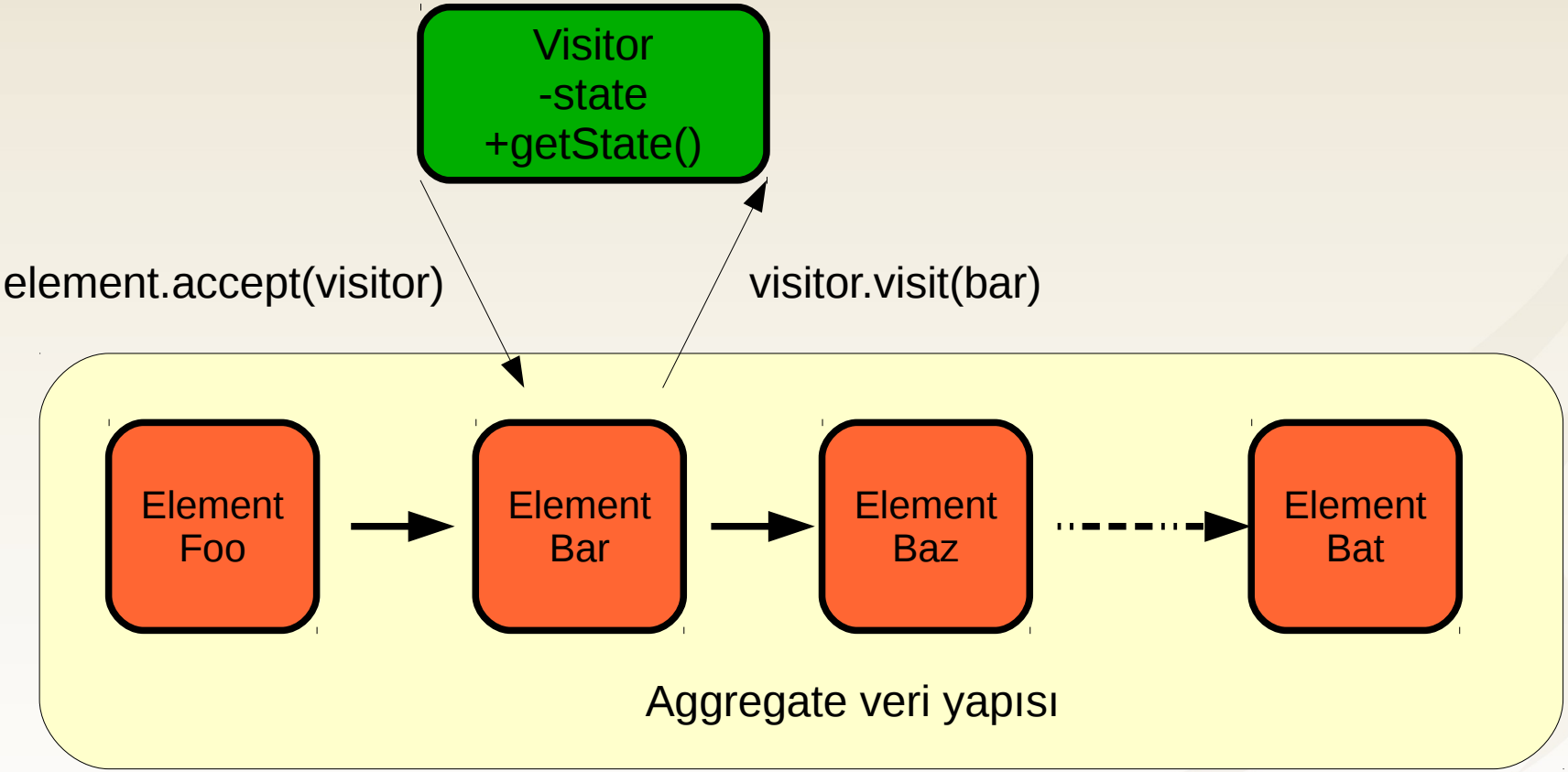
Visitor

- Bir nesne hiyerarşisi içerisinde **değişik türde sınıflar** olabilir
- Farklı senaryolar için bu nesne hiyerarşisi üzerinde **farklı operasyonların** tanımlanması gerekebilir
- Bu operasyonlar **çok çeşitli** ve birbirlerinden **alakasız** olabilir
- Hatta operasyonların işletilmesi belirli bir takım **kurallara veya duruma göre** olabilir
- Bu operasyonların hiyerarşik yapıdaki sınıflara eklenmesi bu sınıfları **kirletecektir**

Visitor

- Yeni operasyon türleri ortaya çıktığında **hiyerarşide de değişikliklere gidilmesi** de gerekebilecektir
- Visitor örüntüsü ile bu operasyonlar **nesne hiyerarşisi dışında ayrı bir sınıf**ta encapsule edilir
- Nesne hiyerarşisi **Visitor tarafından dolaşılarak** her bir nesne için spesifik bir operasyon işletilir
- Dolaşma ve bu operasyonlar sonucu Visitor'de **biriken bilgi** istemci tarafından alınıp kullanılır

Visitor

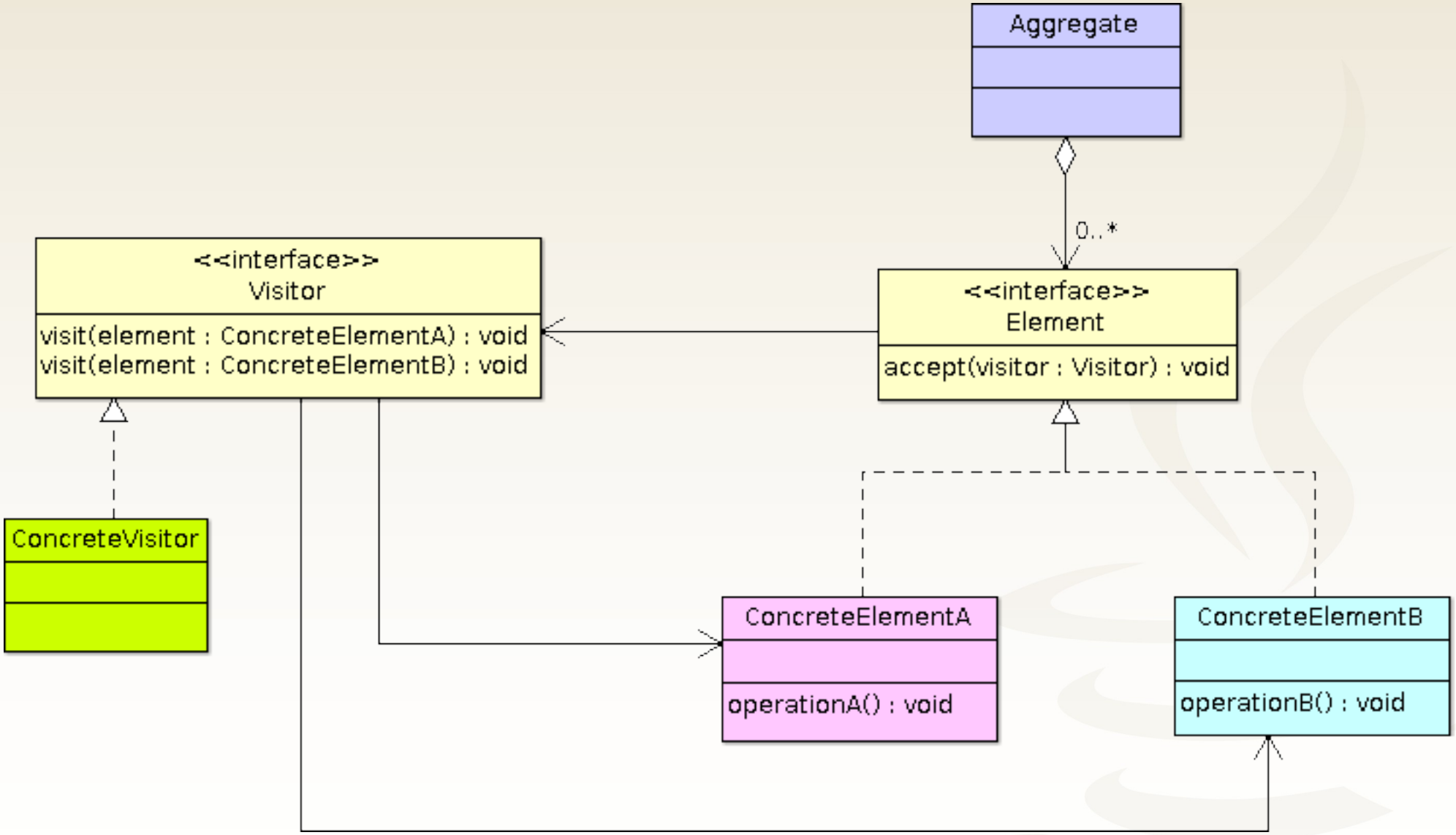


Visitor, aggregate veri yapısındaki her bir elemanı ziyaret eder

Bu sırada Visitor üzerinde state bilgisi birikebilir

Dolaşma sonucunda Visitor üzerinden bu bilgiye erişilebilir

Visitor Sınıf Diagramı



Java ve Visitor

- Java **polymorphic** (çok formlu) bir programlama dilidir
- Çok formluluk bir nesnenin metodu çağrıldığı vakit bu **metot çağrısı**nın nesnenin **sınıf hiyerarşisindeki hangi metot tanımı ile gerçekleştirileceği**nin tespitinin **çalışma zamanına kadar ötelenmesi** ile elde edilir
- Buna **late binding** adı da verilir

Java ve Visitor: Single Dispatch

```
public class A {

}

public class B extends A {

}
```

```
public class Visitor {

    public void visit(A a) {
        System.out.println("visit A");
    }

    public void visit(B b) {
        System.out.println("visit B");
    }
}

public class SubVisitor extends Visitor {

    @Override
    public void visit(A a) {
        System.out.println("sub visit A");
    }

    @Override
    public void visit(B b) {
        System.out.println("sub visit B");
    }
}
```

Java ve Visitor: Single Dispatch

```
A a1 = new A();
A a2 = new B();

Visitor v = new SubVisitor();

v.visit(a1);
v.visit(a2);
```

```
sub visit A
sub visit A
```

Console çıktısı

- Hangi metot tanımının kullanılacağına karar verilirken metodun çağrıldığı nesnenin tipine bakılarak karar verilmesine **single dispatch** adı verilir
- Bazı diller hangi metodun çağrılacağına karar verirken **metot input argümanlarının tipi**ne de bakarlar
- Bu işleme **double dispatch** adı verilir

Java ve Visitor: Double Dispatch

```
public class A {
    public void accept(Visitor v) {
        v.visit(this);
    }
}

public class B extends A {
    public void accept(Visitor v) {
        v.visit(this);
    }
}
```

```
A a1 = new A();
A a2 = new B();

Visitor v = new SubVisitor();

a1.accept(v);
a2.accept(v);
```

```
sub visit A
sub visit B
```

Console çıktısı

Java ve Visitor: instanceOf ile Double Dispatch

JAVA Eğitimleri

```
public interface Element {
}

public class ConcreteElementA
             implements Element {
}
public class ConcreteElementB
             implements Element {
}
```

```
public class ConcreteVisitor extends Visitor {
     @Override
     protected void visit(ConcreteElementA elementA) {
          //...
     }
     @Override
     protected void visit(ConcreteElementB elementB) {
          //...
     }
}
```

```
public abstract class Visitor {
     public void visit(Element element) {
          if(element instanceof ConcreteElementA) {
               this.visit((ConcreteElementA)element);
          } else if(element instanceof ConcreteElementB) {
               this.visit((ConcreteElementB)element);
          }
     }

     protected abstract void visit(ConcreteElementA elementA);

     protected abstract void visit(ConcreteElementB elementB);
}
```

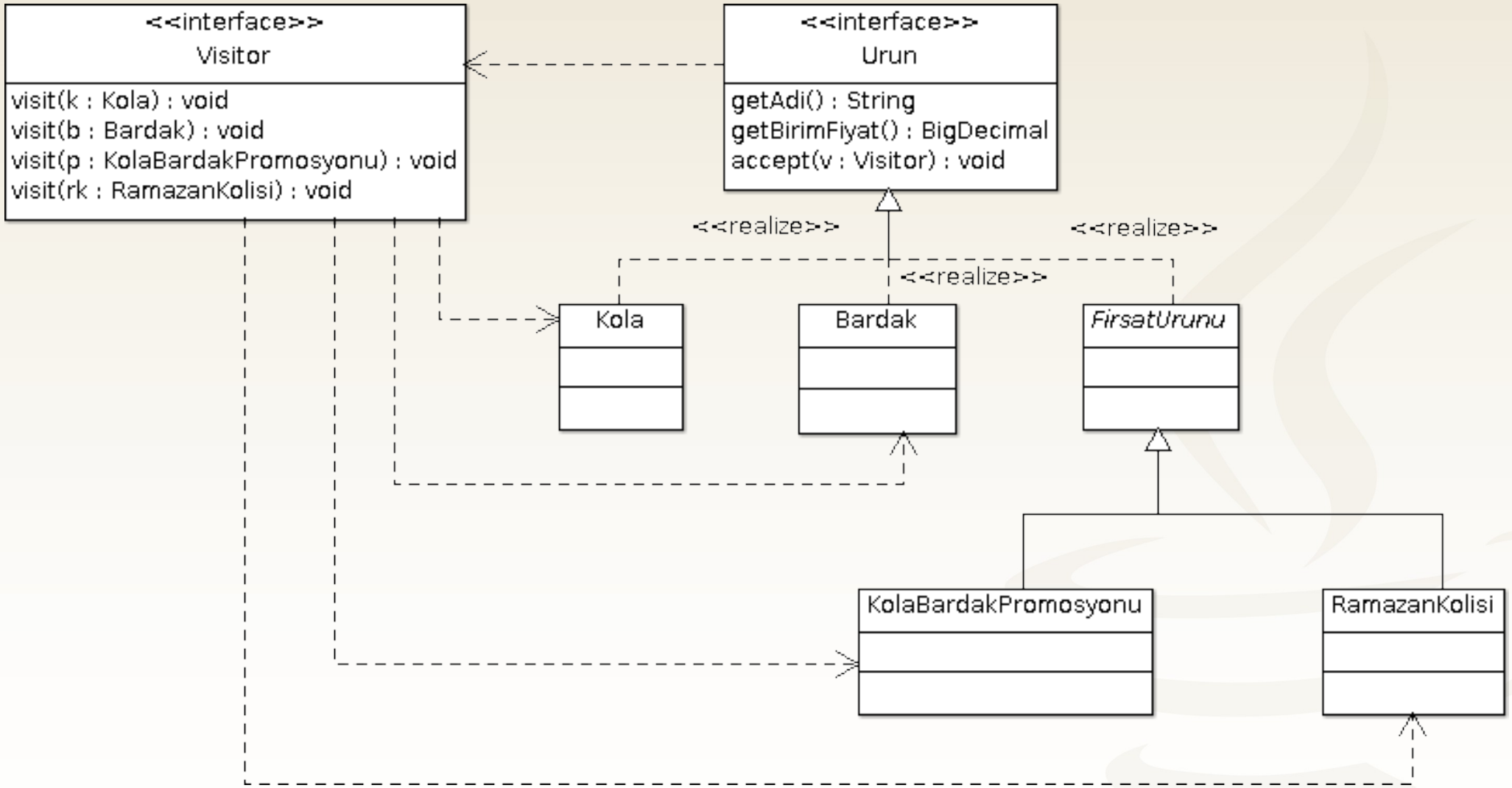
```
Visitor v = new ConcreteVisitor();
Element e1 = new ConcreteElementA();
Element e2 = new ConcreteElementB();

v.visit(e1);
v.visit(e2);
```

LAB ÇALIŞMASI: Visitor

- Müşterinin market alışverişi sırasında kasaya gitmeden önce sepetlerindeki ürünlerin toplam fiyatını hesaplayacak bir sistem istenmektedir
- Sistem ücret hesaplaması sırasında her bir fırsat ürününden sadece bir tanesinin indirimli fiyatı ile alınmasına izin vermelidir
- Eğer birden fazla fırsat ürünü mevcut ise diğerlerinin fiyatı içerdikleri temel ürünlerin toplam fiyatından hesaplanmalıdır
- Market içerisinde ayrıca 5 kola alana bir kola bedava gibi kampanyalarda düzenlenebilmektedir. Hesaplama bu kampanyaları da dikkate almalıdır

LAB ÇALIŞMASI: Visitor



Visitor Örüntüsünün Sonuçları

- Operasyonlar veri yapısı içerisine dağılmamış, **tek bir yerde** Visitor içerisinde **toplanmış** olur
- Veri yapısı üzerinde çalışacak **yeni bir operasyon** eklemek çok kolaydır
- Bu operasyon için **yeni bir Visitor** sınıfı implement edilmesi yeterlidir
- Ancak veri yapısına **yeni türde eleman** eklenmesi zor olabilir. Bu durumda bütün Visitor sınıflarının değişmesi gerekecektir
- Visitor'ün bazı durumlarda veri yapısının **dahili state bilgisi**ne erişmesi gerekebilir

İletişim

www.harezmi.com.tr

www.java-egitimleri.com

info@harezmi.com.tr

info@java-egitimleri.com

@HarezmiBilisim

@JavaEgitimleri