Yazılım Geliştirme Ortam ve Aracları

3. Hafta

UML Diyagramları

Kaynaklar:

1-) http://web.firat.edu.tr/iaydin/bmu112/week_6_uml.pdf
2-) http://www.kasapbasi.org/Dersler/Nesneye%20dayal%C4%B1%20prog/UML.pdf
3-) http://idiotechie.com/uml2-class-diagram-in-java/
4-) https://www.cs.umd.edu/class/spring2007/cmsc132/Lectures/20-UML-2p.pdf

UML

- UML nedir?
- UML Kullanımının Faydaları
- Sınıf (Class) Diyagramları
- Kullanım (Use case) Diyagramları
- Ardışık (Sequence) Diyagramları
- Durum (State) Diyagramları
- Aktivite/Etkinlik Diyagramları

UML Nedir?

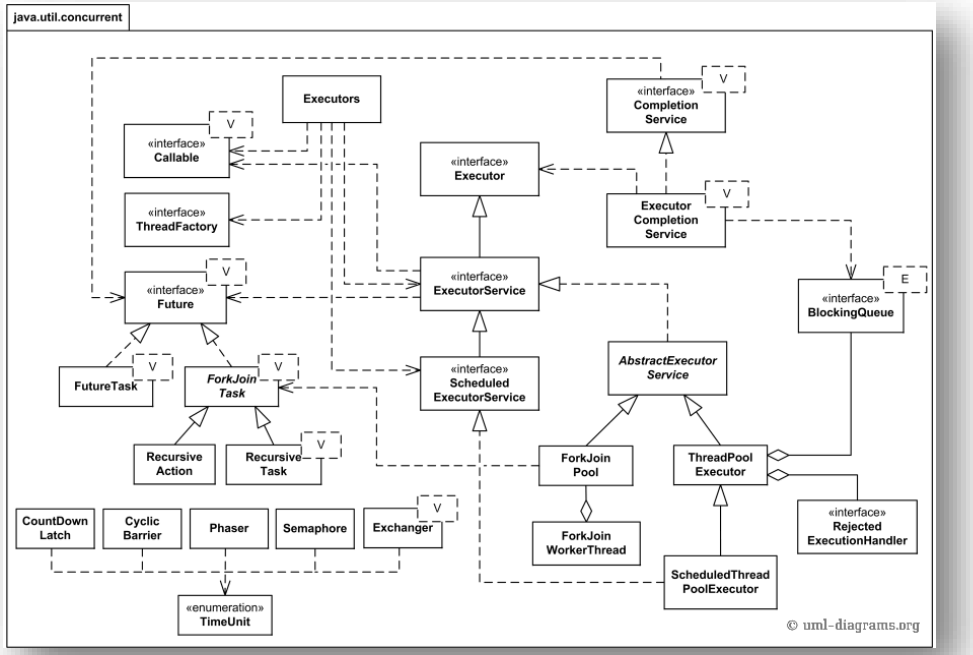
* Yazılım ve donanımların bir arada düşünülmesi gereken,
* Zor ve karmaşık programların,
* Özellikle birden fazla yazılımcı tarafından kodlanacağı durumlarda,
* Standart sağlamak amacıyla endüstriyel olarak geliştirilmiş grafiksel bir dildir.
* ~~Programlama dili~~ Diyagram çizme ve ilişkisel modelleme dili

UML Nedir?

* UML yazılım sisteminin önemli bileşenlerini tanımlamayı, tasarlamayı ve dokümantasyonunu sağlar
* Yazılım geliştirme sürecindeki tüm katılımcıların (kullanıcı, iş çözümleyici, sistem çözümleyici, tasarımcı, programcı,...) gözüyle modellenmesine olanak sağlar,
* UML gösterimi nesneye dayalı yazılım mühendisliğine dayanır.

UML Kullanımının Faydaları

* Yazılımın geniş bir analizi ve tasarımı yapılmış olacağından kodlama işlemi daha kolay ve doğru olur
* Hataların en aza inmesine yardımcı olur
* Geliştirme ekibi arasındaki iletişimi kolaylaştırır
* Tekrar kullanılabilir kod sayısını artırır
* Tüm tasarım kararları kod yazmadan verilir
* Yazılım geliştirme sürecinin tamamını kapsar
* “resmin tamamını” görmeyi sağlar



Sınıf Diyagramları

- Sınıf ve Nesne Gösterimi
- İlişkiler
- Kalıtım
- İçerme
- Ara yüz

Sınıf Diyagramları

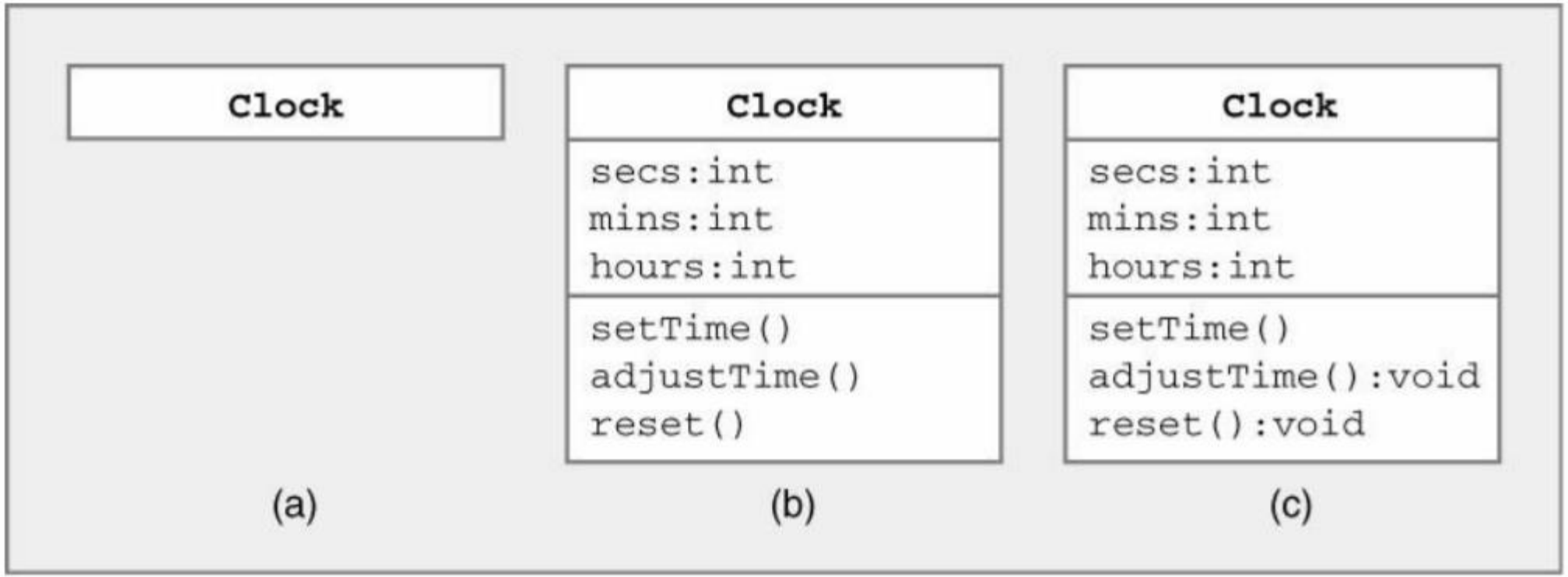
* Sınıf Diyagramları UML 'in en sık kullanılan diyagram türüdür.
* Sınıflar nesne tabanlı programlama mantığından yola çıkarak tasarlanmıştır.
* Sınıf diyagramları bir sistem içerisindeki nesne tiplerini ve birbirleri ile olan ilişkileri tanımlamak için kullanılırlar.

Sınıf Diyagramları

* Sınıfların
  * bir adı
  * nitelikleri ve
  * İşlevleri vardır
  * Bunlara ek olarak
    * “notes”
    * “Constraints”

Sınıf Diyagramları

Sadece sınıf adı zorunlu



Sınıf Diyagramları

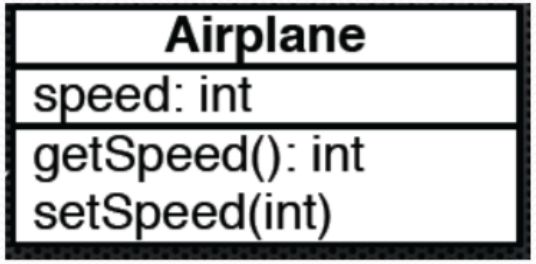
```
class Clock { // name
   // state
   int seconds;
   int minutes;
   int hours;
   // behavior
   void start();
   void adjustTime();
   void reset();
}
```

Java Code

Class Diagram

Sınıf Diyagramları

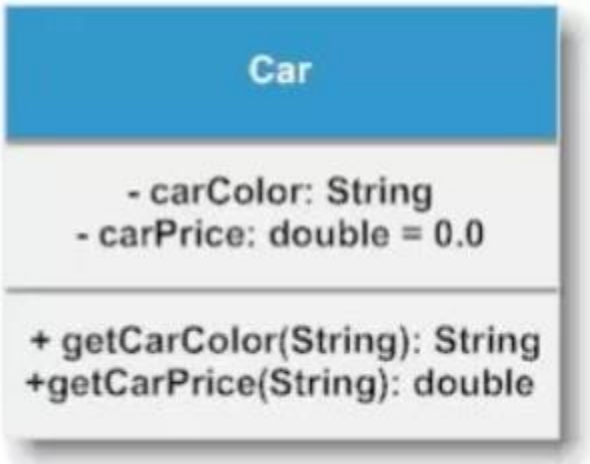
UML Diyagramı




```java
public class Airplane {

    private int speed;

    public Airplane(int speed) {
        this.speed = speed;
    }

    public int getSpeed() {
        return speed;
    }

    public void setSpeed(int speed) {
        this.speed = speed;
    }

}
```

Sınıf Diyagramları

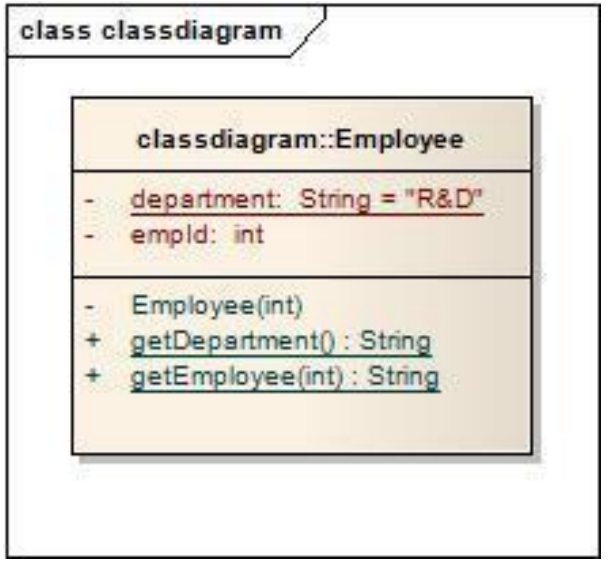
```java
public class Car {
    private String carColor;
    private double carPrice = 0.0;
    public String getCarColor(String model) {
        return carColor;
    }

    public double getCarPrice(String model) {
        return carPrice;
    }
}
```



Java visibility	UML Notation
public	+
private	–
Protected	#
package	~

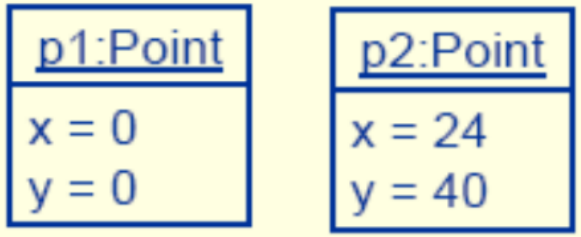
Sınıf Diyagramları

```java
public class Employee {
    private static String department = "R&D";
    private int empId;
    private Employee(int employeeId) {
        this.empId = employeeId;
    }
    public static String getEmployee(int emplId) {
        if (emplId == 1) {
            return "idiotechie";
        } else {
            return "Employee not found";
        }
    }
    public static String getDepartment() {
        return department;
    }
}
```



Sınıf Diyagramları – Nesne Gösterimi

UML

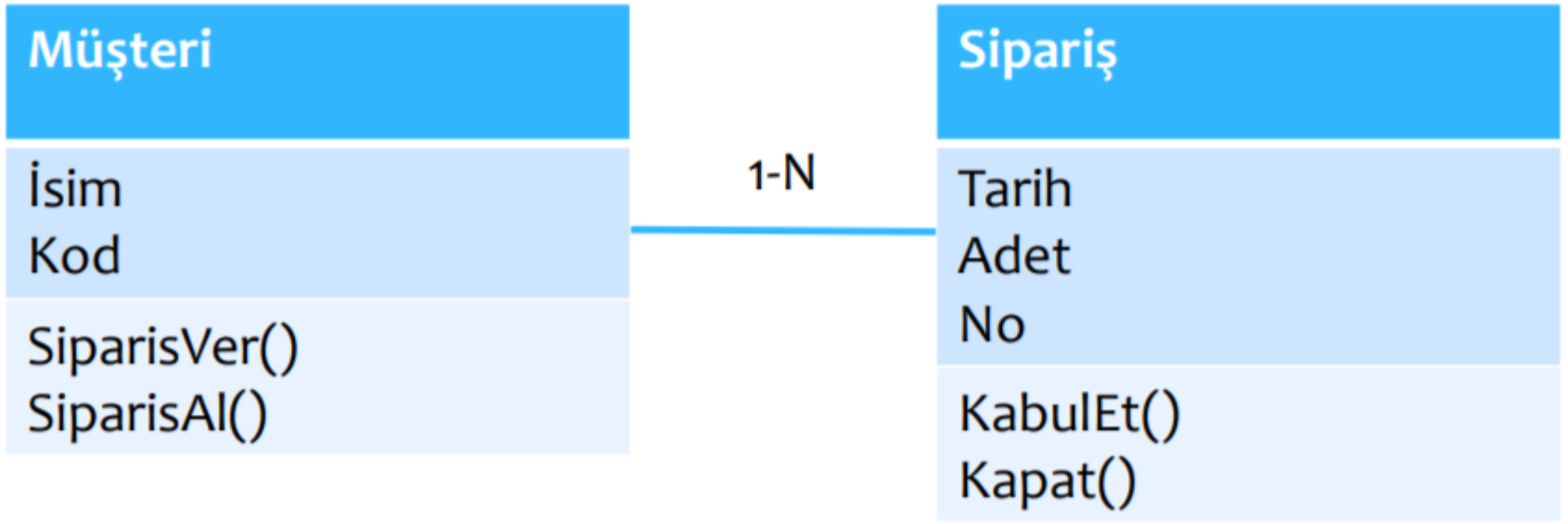


Java

```
Point p1 = new Point();
p1.x = 0;
p1.y = 0;

Point p2 = new Point();
p2.x = 24;
p2.y = 40;
```

Sınıf Diyagramları – İlişkiler

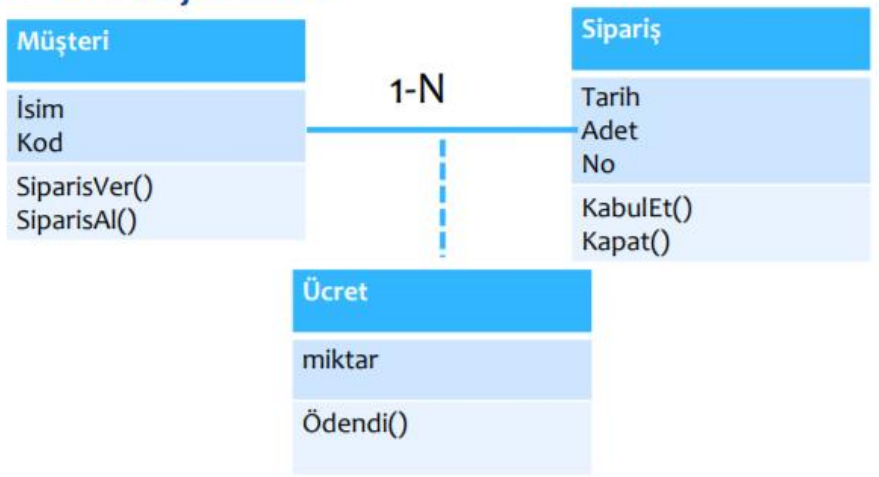
* Sınıflar arasındaki ilişkiyi göstermek için iki sınıf arasına düz bir çizgi çekilir.
* İlişkiyi gösteren çizginin üzerine ilişkinin türü yazılır.



Sınıf Diyagramları – İlişkiler

* Bazı durumlarda sınıflar arasındaki ilişki, bir çizgiyle belirtebilecekşekilde basit olmayabilir.
* Bu durumda ilişki sınıfları kullanılır.
* İlişki sınıfları bildigimiz sınıflarla aynıdır.
* Sınıflar arasındaki ilişki eğer bir sınıf türüyle belirleniyorsa UML ile gösterilmesi gerekir.

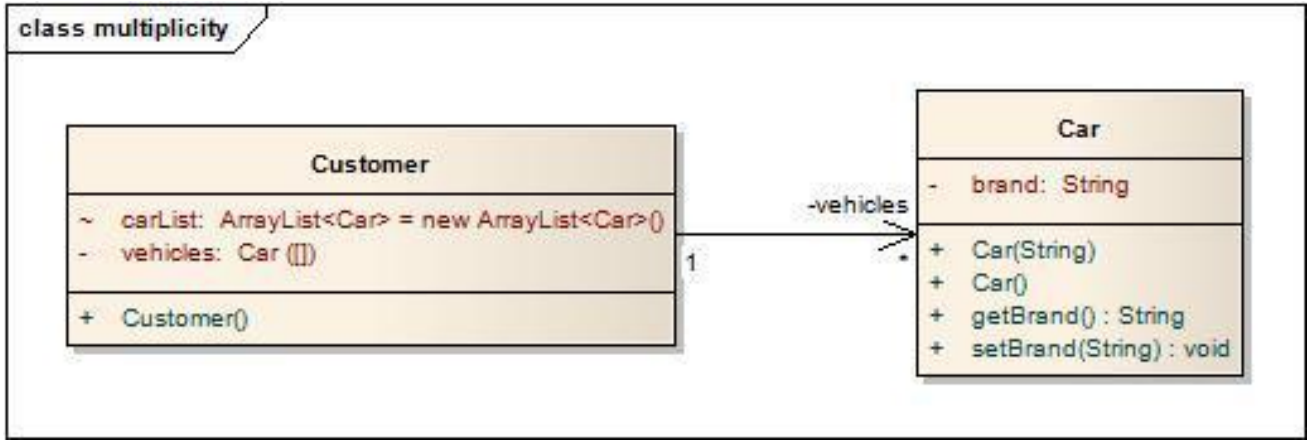
* Müşteri ile Sipariş sınıfı arasında ilişki vardır.Fakat müşteri satın alırken Ücret ödemek zorundadır
* Bu ilişkiyi göstermek için Ücret sınıfı ilişki ile kesikli çizgi ile birleştirilir.



Sınıf Diyagramları – İlişkiler


```java
public class Customer {
private String name;
private String address;
private String contactNumber;
private Car car;
}

public class Car {
    private String modelNumber;
    private Customer owner;
}
```

Sınıf Diyagramları – İlişkiler



Car.java

Code:

```java
public class Car {
    private String brand;

    public Car(String brands){
        this.brand = brands;
    }
    public Car() {
    }
    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

}
```

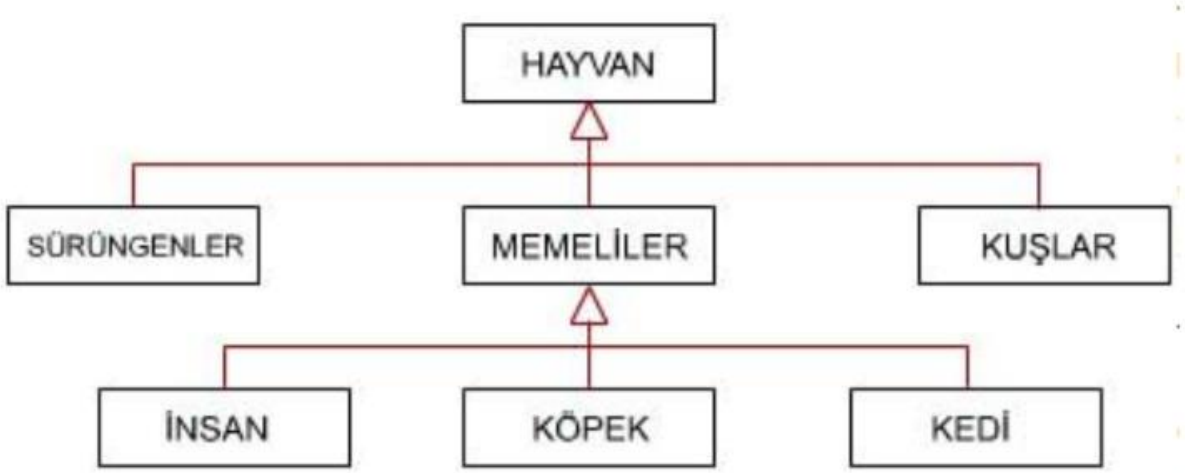
Customer.java

Code:

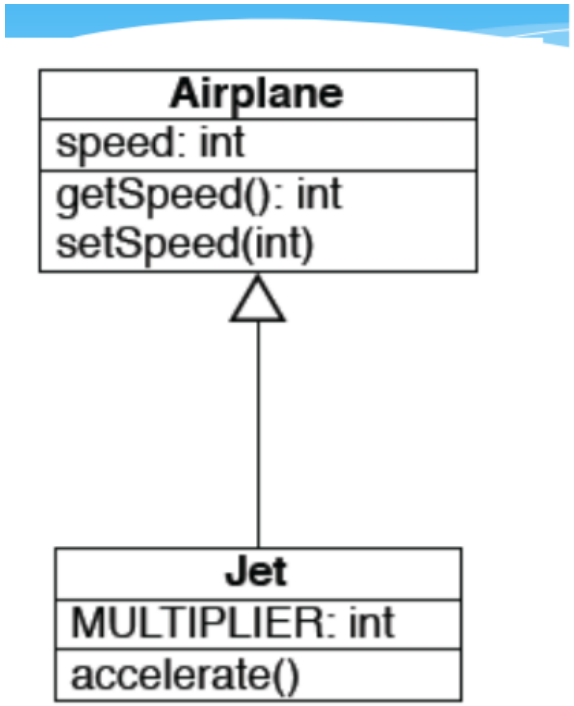
```java
public class Customer {
    private Car[] vehicles;
    ArrayList<Car> carList = new ArrayList<Car>();
    public Customer(){
        vehicles = new Car[2];
        vehicles[0] = new Car("Audi");
        vehicles[1] = new Car("Mercedes");

        carList.add(new Car("BMW"));
        carList.add(new Car("Chevy"));
    }
}
```

Sınıf Diyagramları –Kalıtım

* Eğer eşyalar arasında genellemeler yapabiliyorsak genellemeyi yaptığımız eşyalarda ortak özelliklerin olduğunu biliriz.
* Mesela, "Hayvan" diye bir sınıfımız olsun.
* Memeliler, Sürüngenler, Kuşlar da diğer sınıflarımız olsun.
* Memeliler, Sürüngenler ve Kuşlar sınıfının farklı özellikleri olduğu gibi hepsinin Hayvan olmasından dolayı birtakım ortak özellikleri vardır.

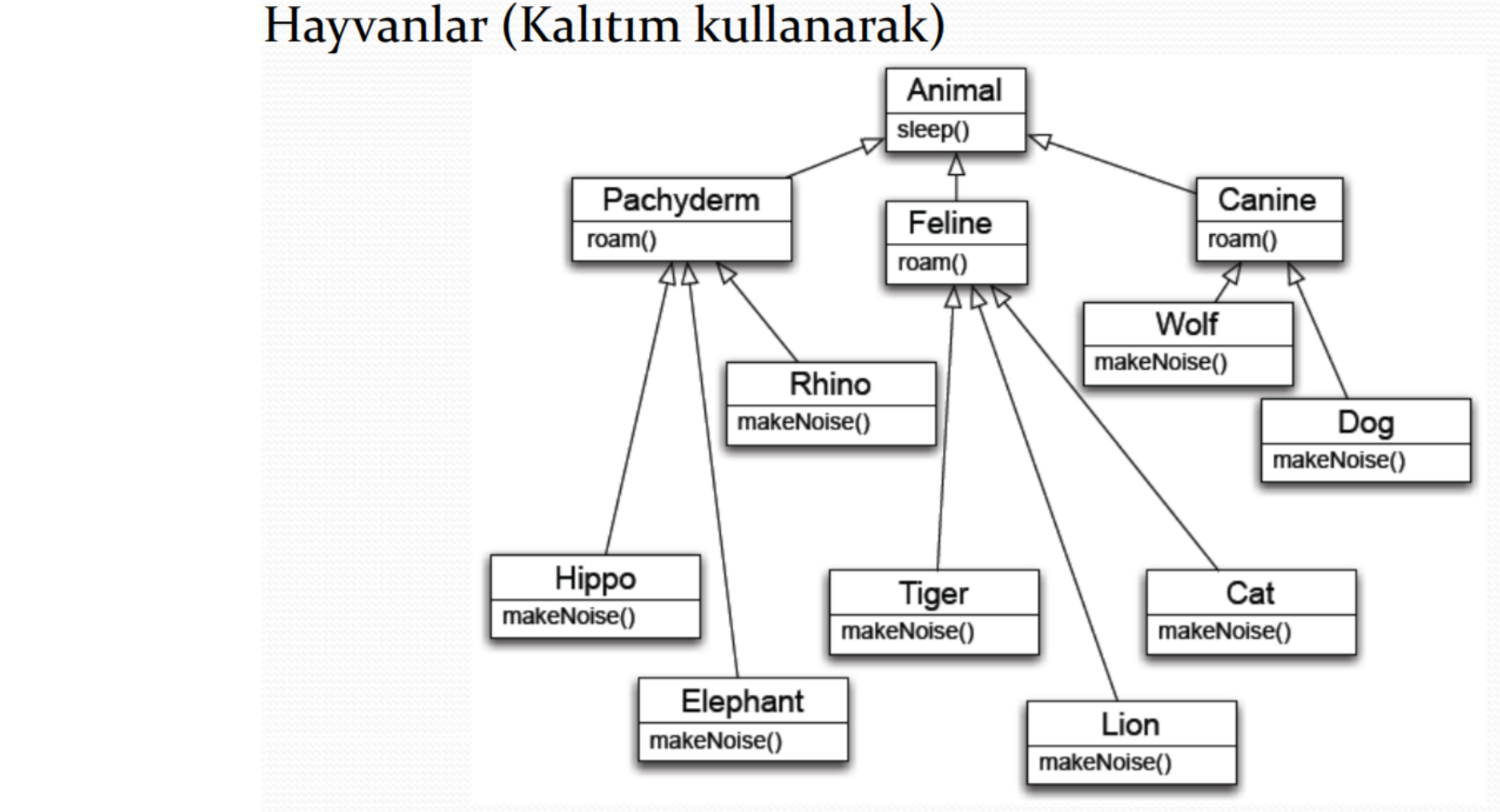


Sınıf Diyagramları – Kalıtım



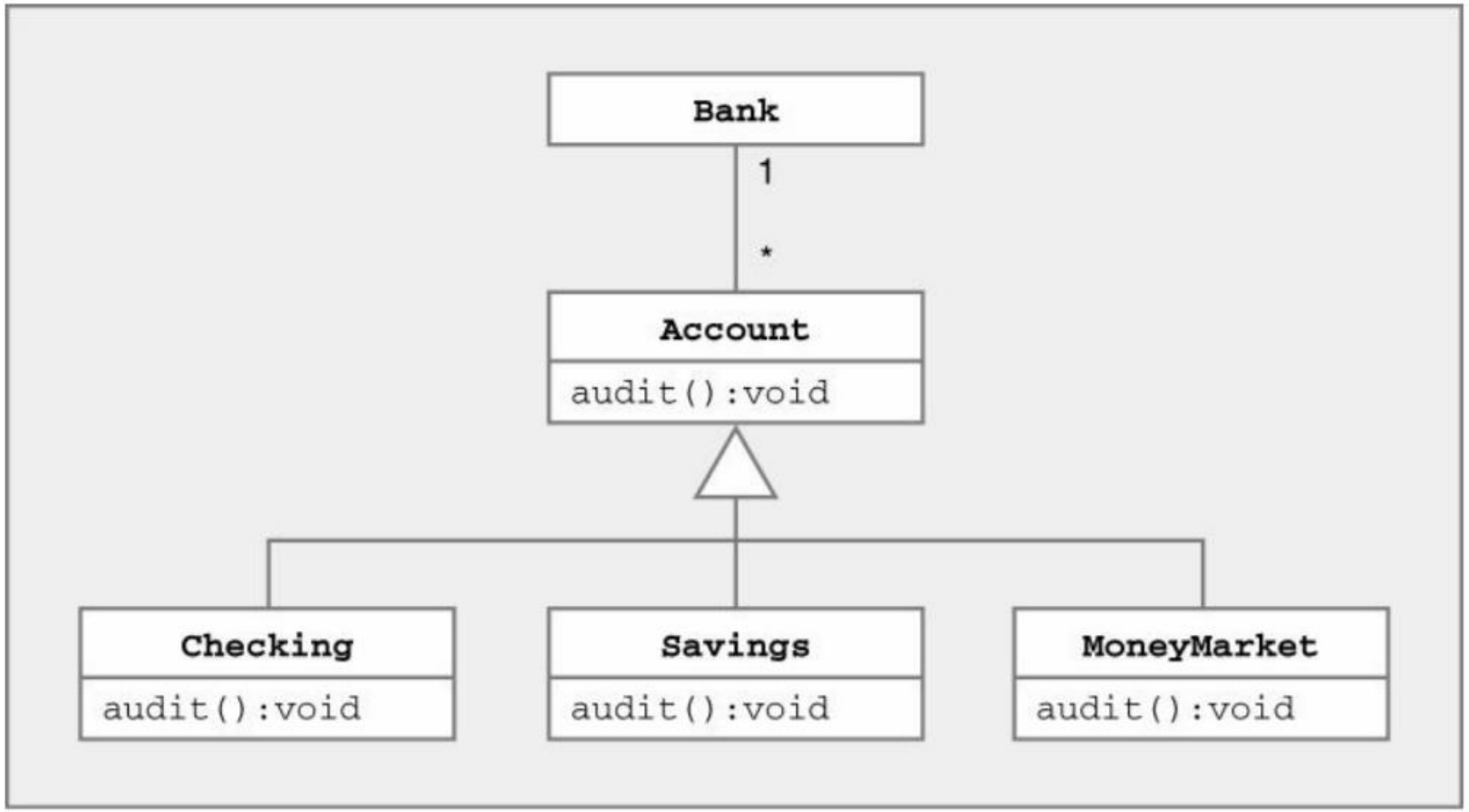

```
1  public class Jet extends Airplane {
2
3      private static final int MULTIPLIER = 2;
4
5      public Jet(int id, int speed) {
6          super(id, speed);
7      }
8
9      public void setSpeed(int speed) {
10         super.setSpeed(speed * MULTIPLIER);
11     }
12
13     public void accelerate() {
14         super.setSpeed(getSpeed() * 2);
15     }
16
17 }
18
```

Sınıf Diyagramları – Kalıtım



Sınıf Diyagramları – Kalıtım

UML Example – Banking System

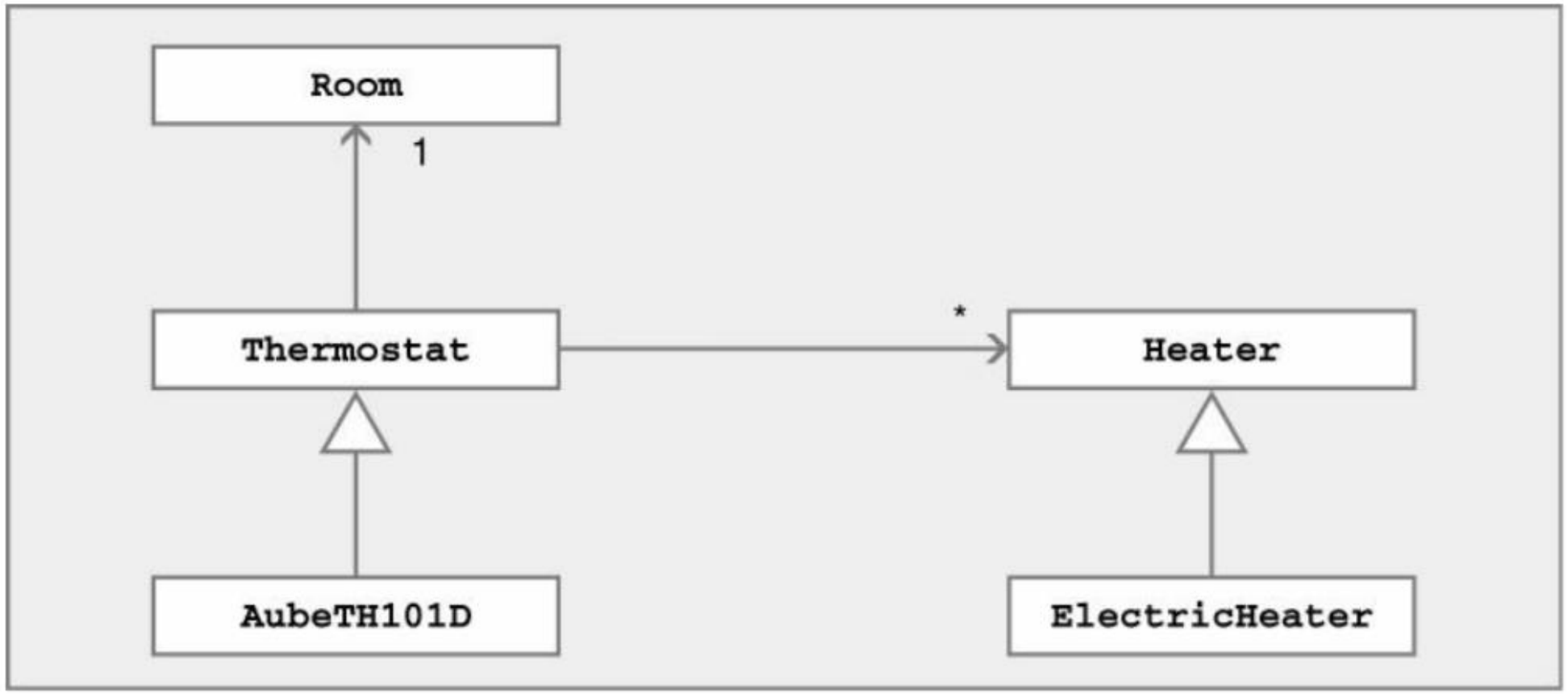


Banka

Hesap

Çek
Yatırım
Borsa

Sınıf Diyagramları – Kalıtım

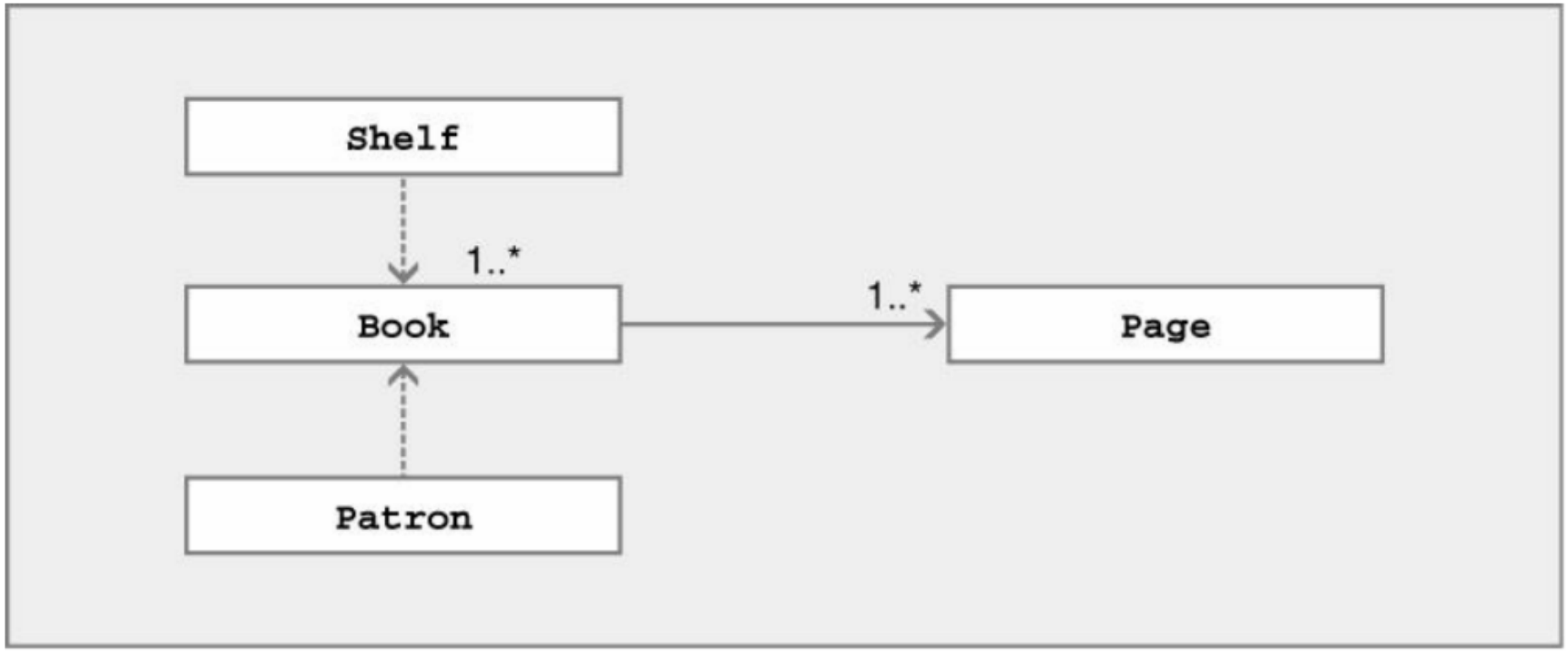


Oda

Termostat ---→ Isıtıcı

Demirdöküm Elek. Isıtıcı

Sınıf Diyagramları – Kalıtım



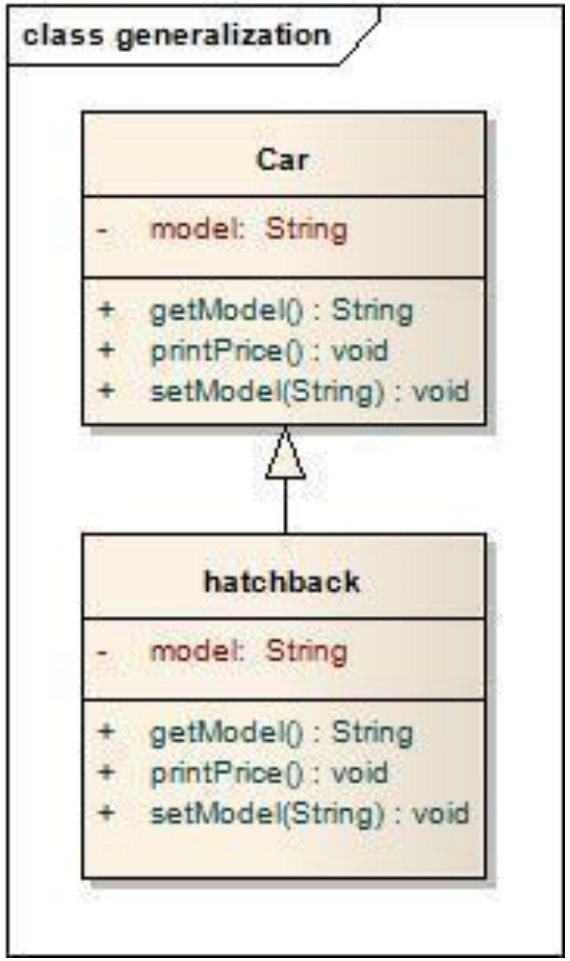
Raf

Kitap --->Sayfa

Müşteri

Sınıf Diyagramları – Kalıtım

```java
public class Car {
    private String model;
    public void printPrice() {
    }
    public String getModel() {
        return model;
    }
    public void setModel(String model) {
        this.model = model;
    }
}

public class hatchback extends Car {
    private String model;
    public void printPrice() {
        System.out.println("Hatchback Price");
    }
    public String getModel() {
        return model;
    }
    public void setModel(String model) {
        this.model = model;
    }
}
```



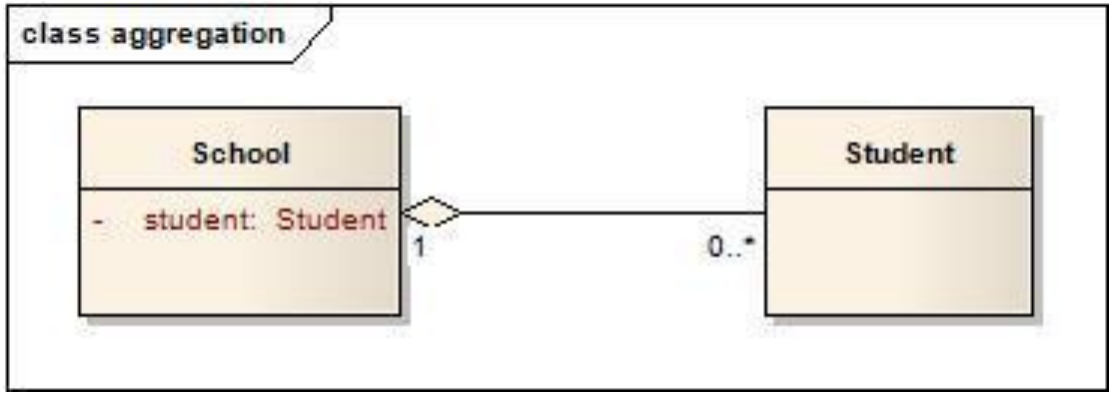
Sınıf Diyagramları – İçerme (Aggregation)

Bazı sınıflar birden fazla parçadan oluşur.Bu tür özel ilişkiye "Aggregation" denir.Mesela ,bir TV 'yi ele alalım.Bir televizyon çeşitli parçalardan oluşmuştur.Ekran,Uzaktan Kumanda,Devreler vs.. Bütün bu parçaları birer sınıf ile temsil edersek TV bir bütün olarak oluşturulduğunda parçalarını istediğimiz gibi ekleyebiliriz. Aggregation ilişkisini 'bütün parça' yukarıda olacak şekilde ve 'bütün parça'nın ucuna içi boş elmas yerleştirecek şekilde gösteririz.

Sınıf Diyagramları – İçerme (Aggregation)

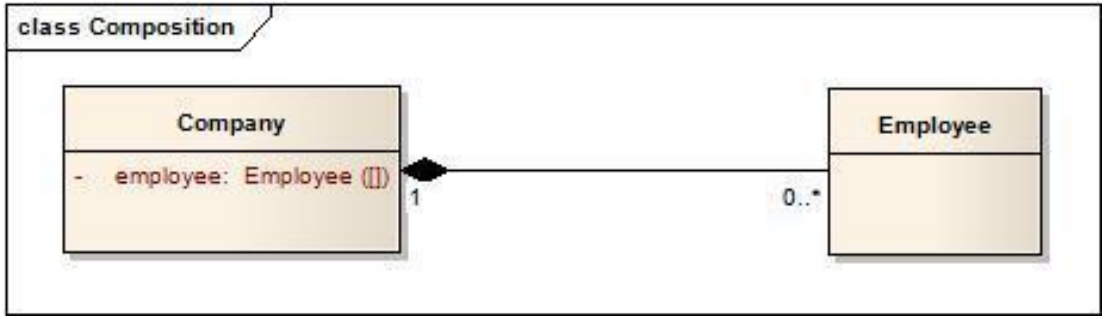
İçerme: Bütün yok olsa da içerilen yaşamaya devam eder.

```
1  public class Student {
2
3  }
4
5  public class School {
6      private Student student;
7  }
```



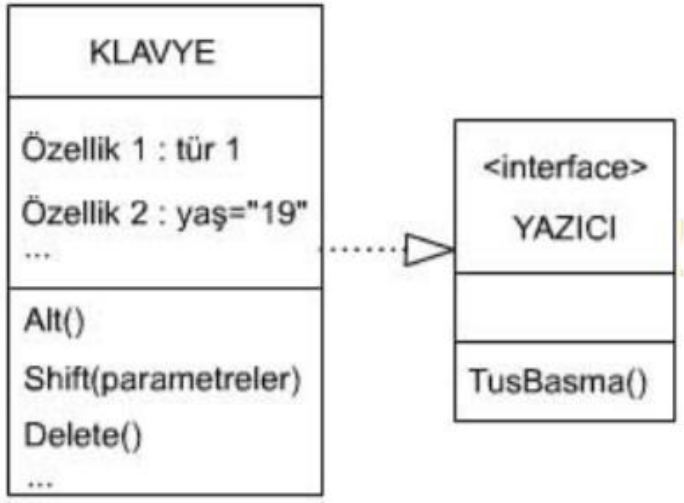
Birleşim: Bütün yok olursa içerilen de yok olur

```
1  public class Employee {
2
3  }
4
5  public class Company {
6      private Employee[] employee;
7  }
```



Sınıf Diyagramları – Arayüz (Interface)

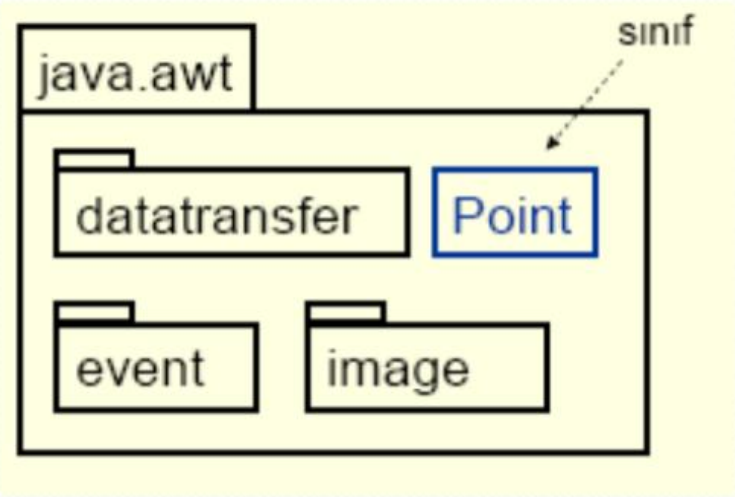
* Arayüzlerin özellikleri yoktur. Yalnızca bir takım işleri yerine getirmek için başka sınıflar tarafından kullanılırlar.
* Mesela, bir "TuşaBasma" arayüzü yaparak ister onu "KUMANDA" sınıfında istersek de aşağıdaki şekilde görüldüğü gibi "KLAVYE" sınıfında kullanabiliriz.



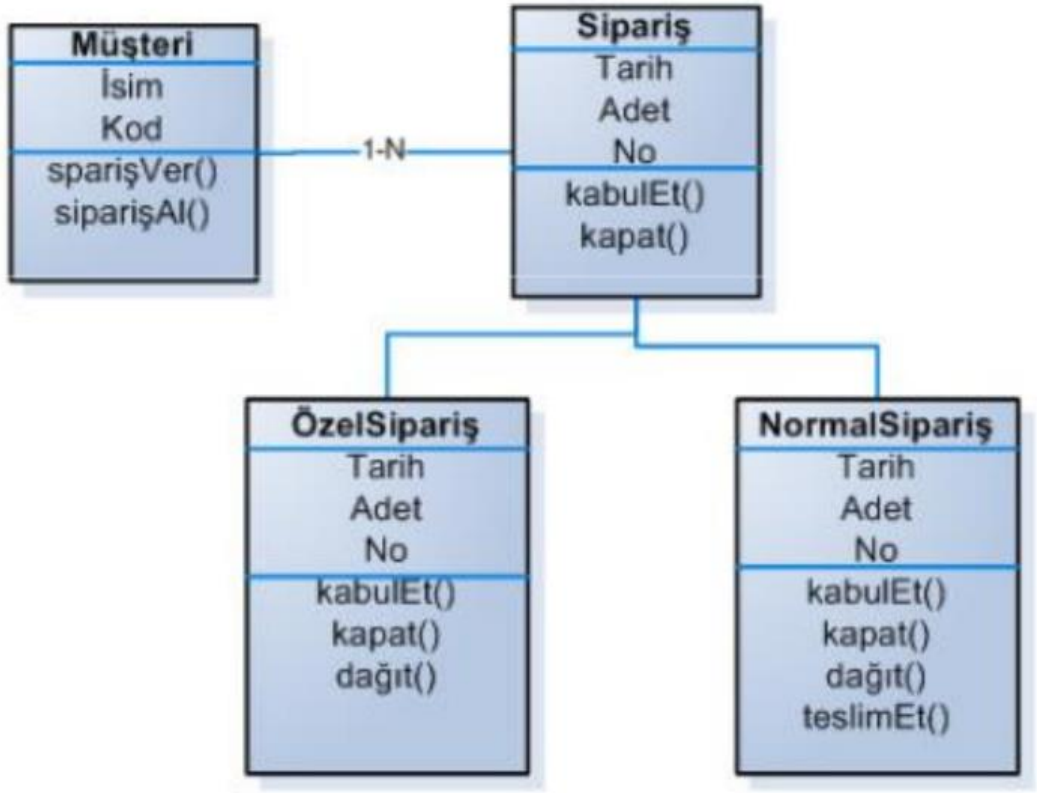
Sınıf Diyagramları – Paket Gösterimi

Birbirleriyle ilişkili sınıflar bir paket (package) içine yerleştirilirler.

Paket isimler küçük harflerle yazılır

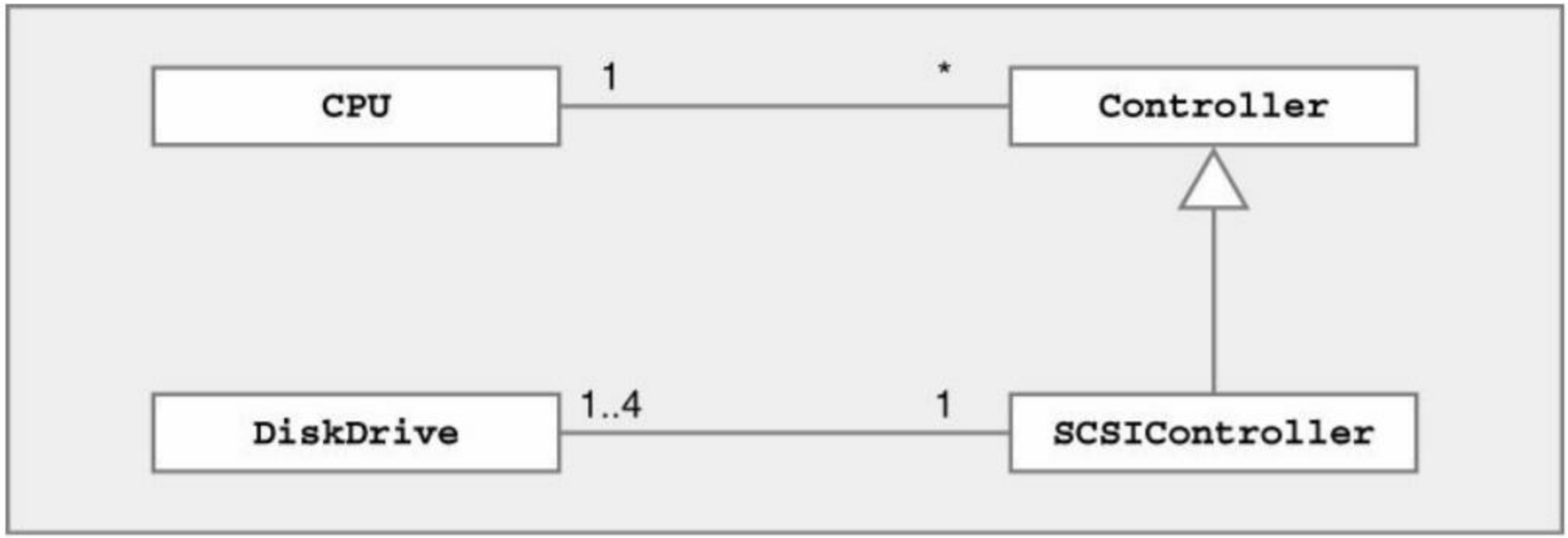


Sınıf Diyagramları – Soru 1



(?) UML diyagramına ait Java Kodunu Yazınız

Sınıf Diyagramları – Soru 2

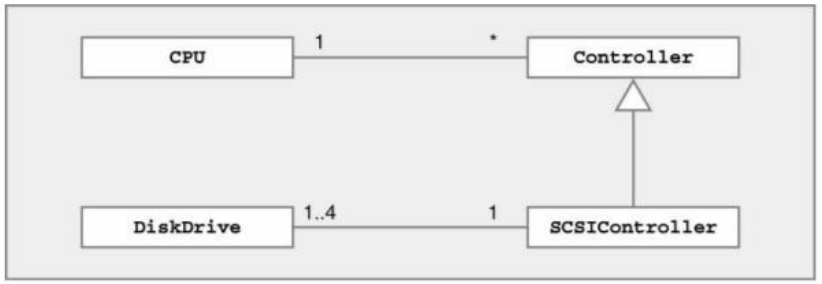


(?) UML diyagramına ait Java Kodunu Yazınız

Sınıf Diyagramları – Soru 2 (Cevap)

- Java

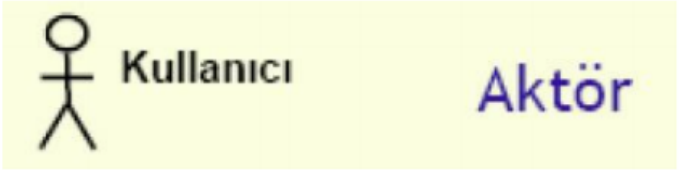
```
class CPU {
   Controller myCtlr;
}
class Controller {
   CPU myCPU;
}
class SCSIController extends Controller {
   DiskDrive myDrive;
}
Class DiskDrive {
    SCSIController mySCSI;
}
```



Kullanım (Use-case) Diyagramları

- Aktör ve Senaryo Gösterimi

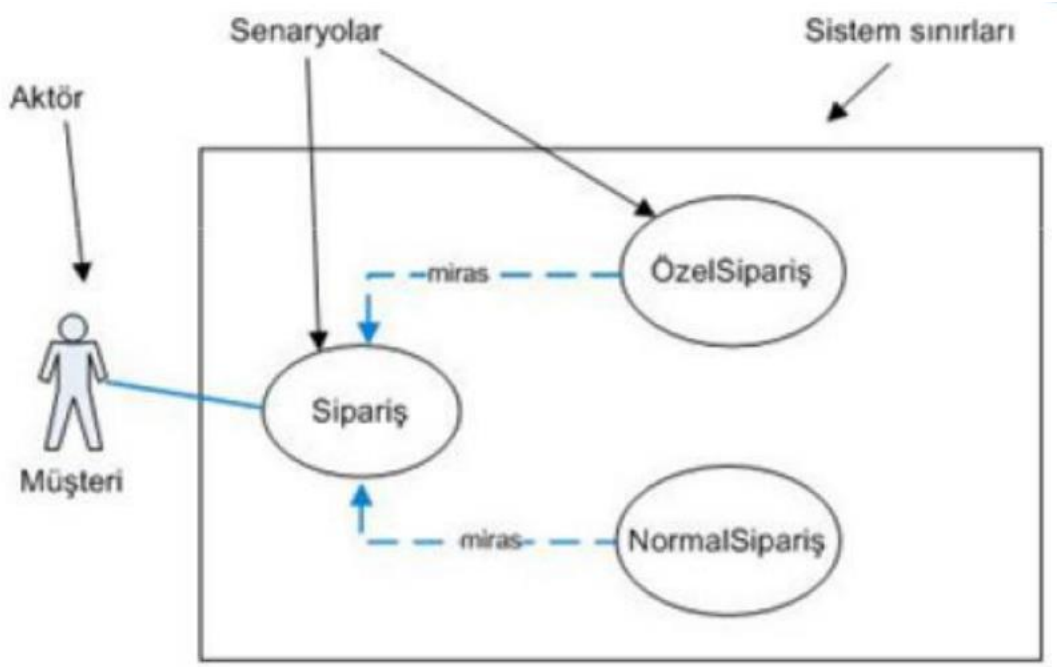
Kullanım Senaryosu Diyagramları



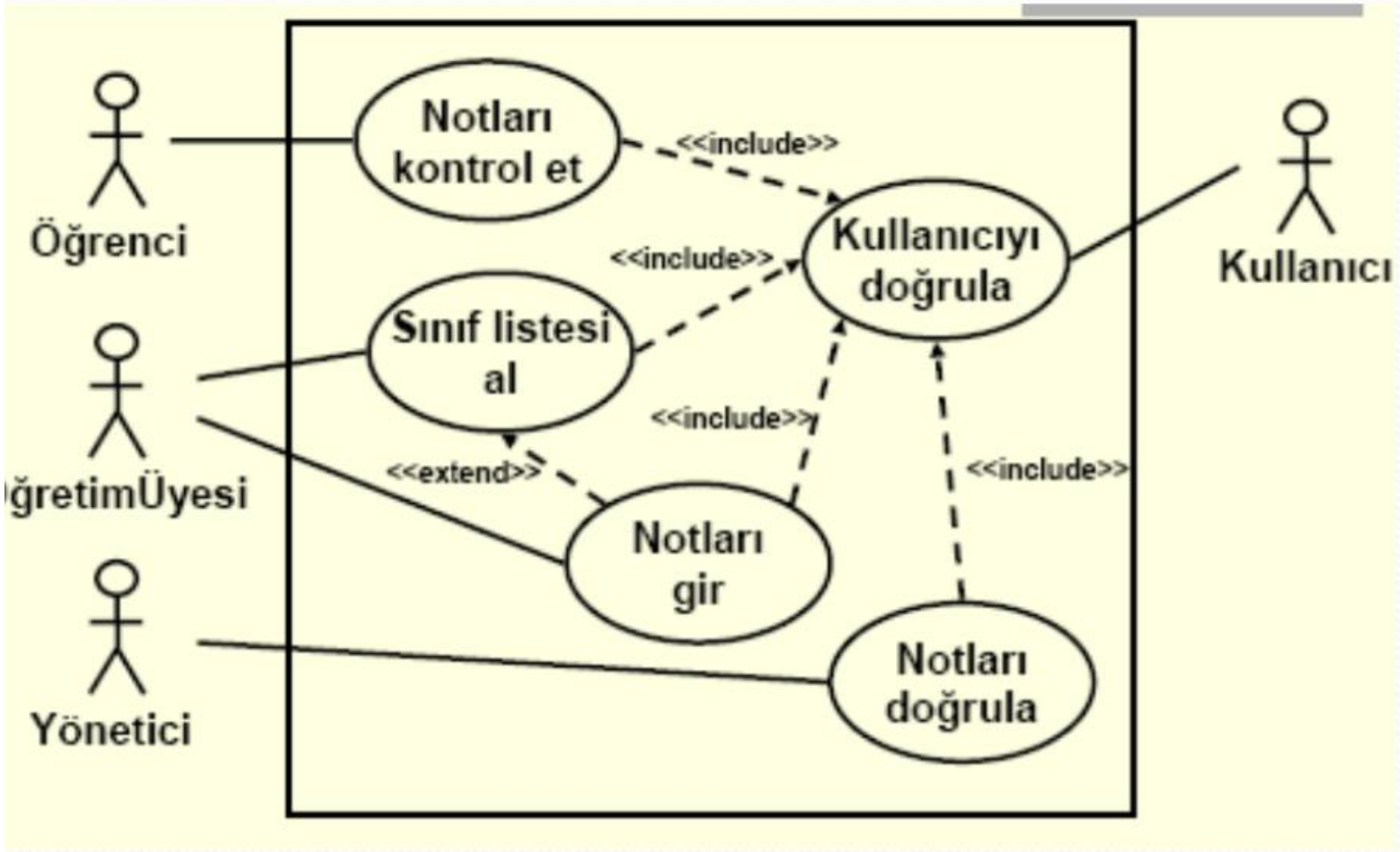
* Aktör genellikle “insan” olarak düşünülmekle birlikte başka “sistem” ve “donanım” da olabilir.
* Aktör sistemi “uyarır” ,işlevleri haricen “tetikler”(aktif) yada sistemden “uyarıcı alır”(pasif).
* Aktör sistemin parçası değildir, “harici” dir.

Kullanım Senaryosu Diyagramları

* Aktörler belirlenir
* Her aktörün “ne” yapmak istediği belirlenir
* Her aktörün “ne” si için “ana senaryo özeti” çıkarılır
* Tüm sistemin ana senaryo özetleri incelenir, ayıklanır, birleştirilir
* Her senaryo için ana işlem adımları belirtilir.



Kullanım Senaryosu Diyagramları



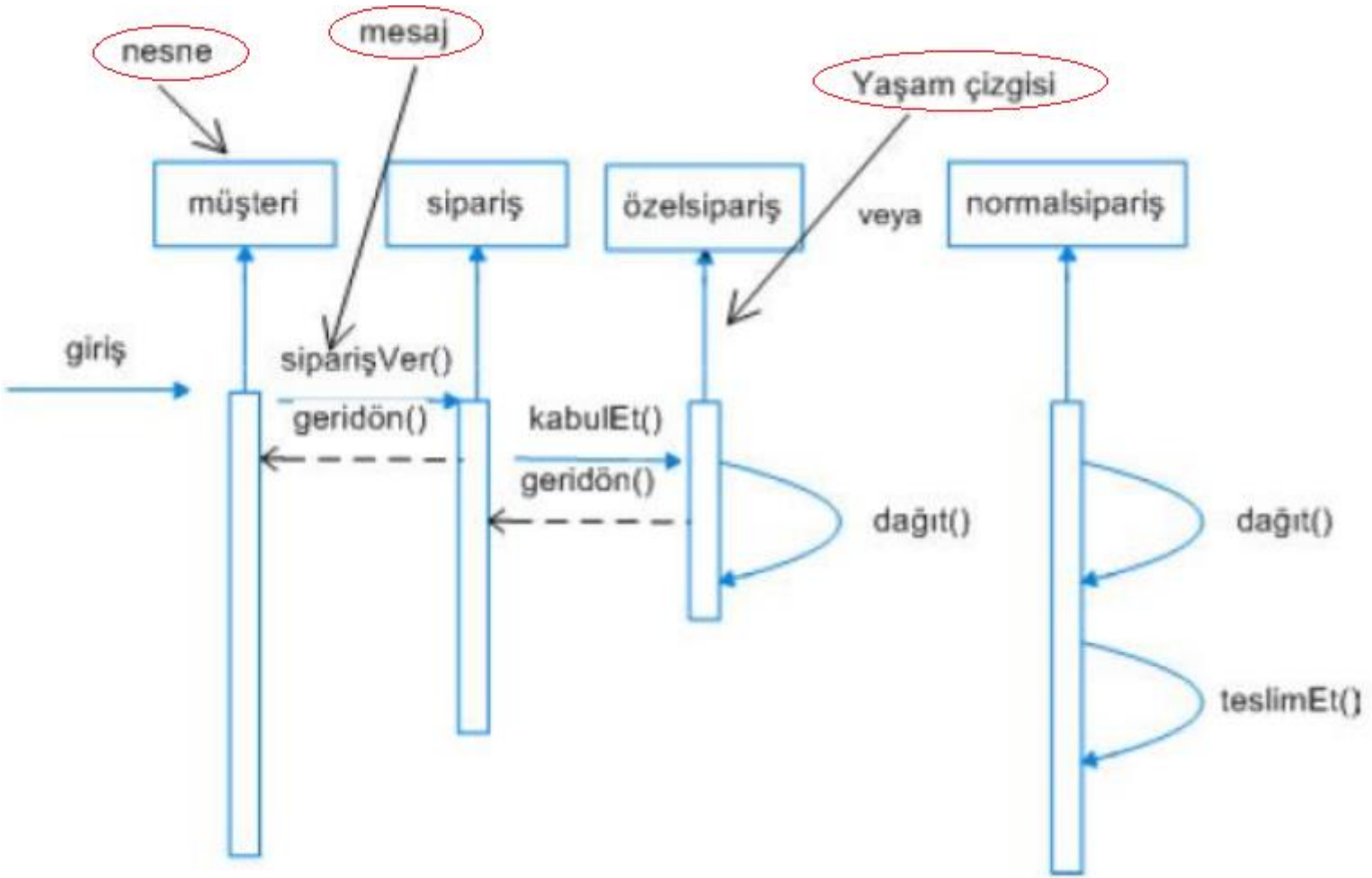
Ardışık Diyagramlar

- Ardışık = Sequence

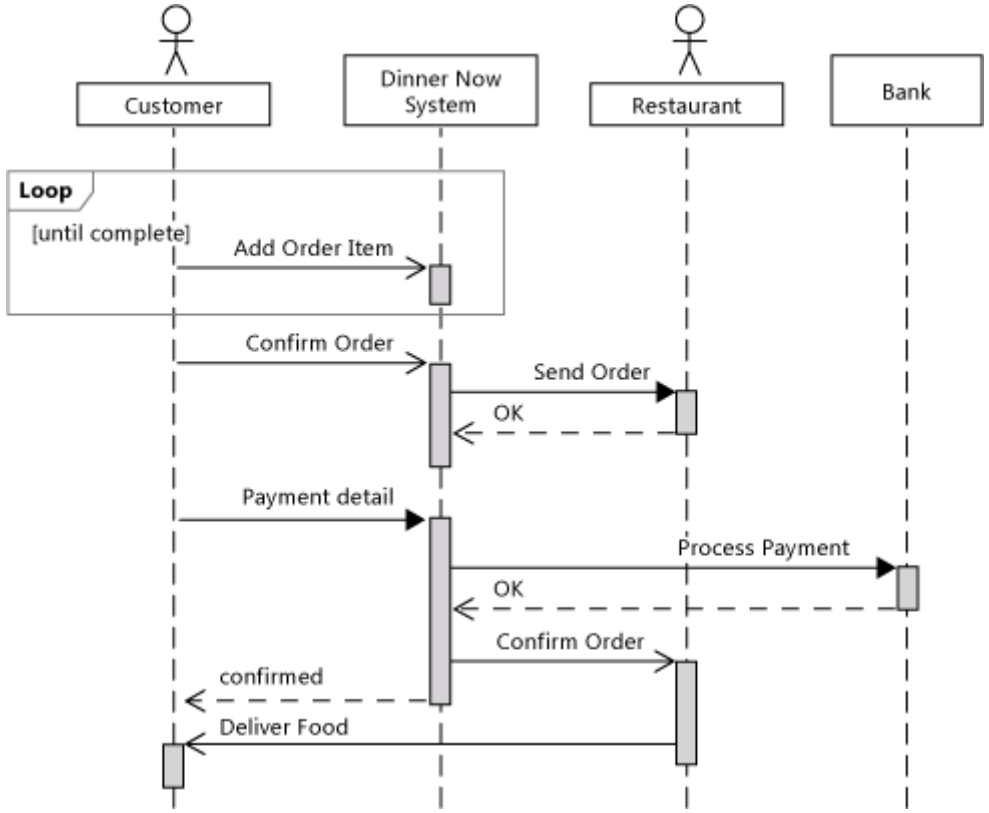
Ardışık Diyagramlar

* Sistem içindeki nesnelerin zaman içindeki ardışık aksiyonlarını gösterirler
* Sistemin dinamik bir resmini çizer
* Aktörün hayat süresi boyunca gerçekleştirdiği işlemler gösterilir.
  * Aksiyonlar->dikdörtgen
  * Etkileşimler->mesajlar

Ardışık Diyagramlar



Ardışık Diyagramlar



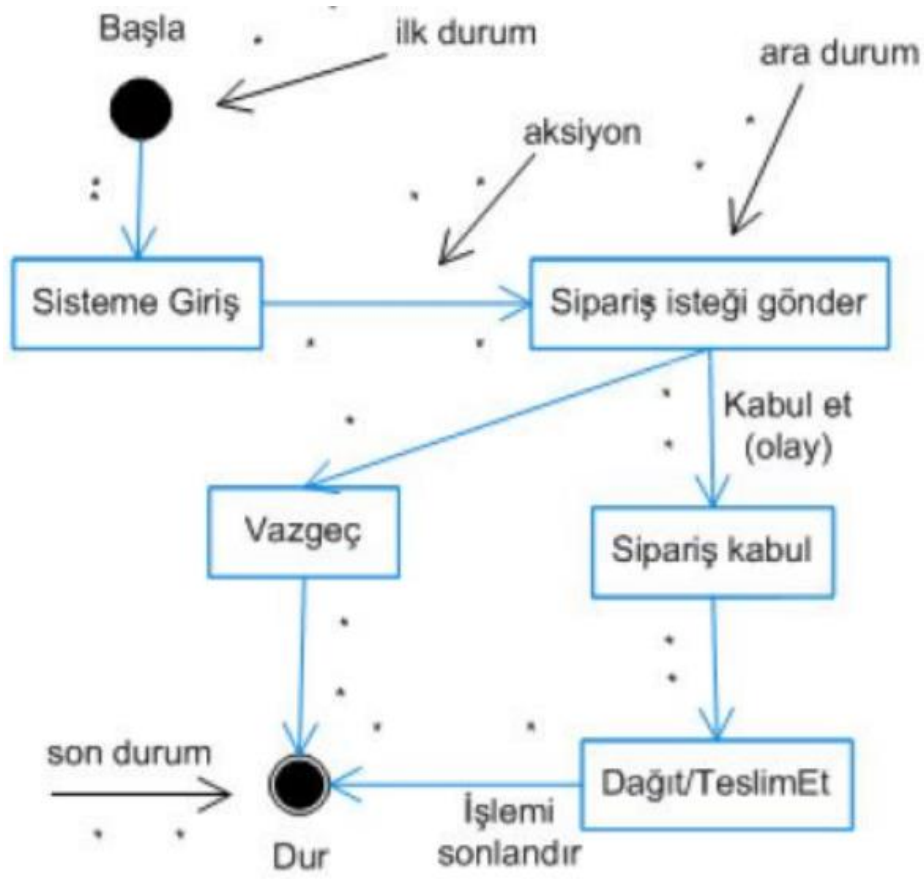

```
class Customer {
     public void islemeBasla() {
        DinnerNowSystem dns =new DinnerNowSystem();
        Restaurant r = new Restaurant();
        Bank b = new Bank();
        while (...) dns.addOrderItem();
        dns.confirmOrder();
        dns.paymentDetail();
        Food f = r.receiveFood();
     }
}

class DinnerNowSystem() {
       Bank b;
       Restaurant r;
       void addOrderItem();
       boolean confirmOrder() {
           return r.sendOrder();
       }
       boolean paymentDetail() {
           if (b.processPayment())
               r.confirmOrder()
           else return false;
       }
}

class Restaurant() {
       boolean sendOrder() {
           return true;
       }
       boolean confirmOrder() {
           r.deliverFood();
       }
       void deliverFood() {
           // send to customer
       }

}

class Bank() {
       void processPayment() {
              return true;
       }
}
```

Durum Diyagramları

- Durum = State

Durum Diyagramları

* Sistemdeki nesnelerin anlık durumlarını göstermek amacıyla kullanılırlar.
* Sistemin küçük alt sistemlere veya nesnelere ilişkin dinamik davranışlarının ortaya çıkartılması amacıyla yararlanılır
* Sistemin durumları ve bunların birbirlerini tetikleme ilişkileri belirtilir



Aktivite Diyagramları

- Etkinlik Diyagramları da denir

Aktivite Diyagramları