Bölümde neler var?

- Sınıf(Class) ve Nesne(Object)
- Nesne türetme/nesne başlatma
- Yapıcı Metodlar (Constructor)
- Kapsülleme (Encapsulation)
- Soyutlama (Abstraction)

Object

Nesne yönelimli programlamada her şeyin atası Object/nesne olarak kabul edilir. Bu, her nesnenin Object sınıfından türediği anlamına gelir. Object sınıfı, tüm nesnelerin ortak özelliklerini ve davranışlarını tanımlar. Her şeyin Object'ten türemesi, programlamada tutarlılık, genişletilebilirlik ve yeniden kullanılabilirlik sağlar.

Class

- Sınıflar nesnelerin yapısını tanımlamak için kullanılan taslak (template) veya şablonlardır(blueprint).
- Classlar kendi özel tiplerimizdir(custom type).
- Classlar mantıksal varlıklardır. Bir tanım içerirler ancak bellekte bir yer kaplamazlar. Bu yüzden soyutlardır.
- Biz ne zaman mantıksal varlık olan classlarımızdan örnekler üretmeye başladığımız zaman fiziksel varlıklar elde ederiz. Bunlara ise object(nesne) diyoruz.
- Nesneler sınıfların örnekleridir(instance), sınıflardaki özellik ve davranışları taşırlar. Entity(varlık),

alt text Örnek olması adına Vehicle adında bir sınıf oluşturalım. Bu yapı sınıf olduğundan yeni bir dosyada oluşturulur. Main isimli dosyada sınıfımızı yazamayız ordada main metodunda programımızı çalıştırcaz. Yeni oluşturduğumuz dosya ile sınıfımızın adı aynı olmalıdır.

```
public class  Vehicle {

}
```

- Burada public erişim belirtecimiz, class bu yapının sınıf olduğunu belirten keyword, Vehicle ise bizim sınıfa verdiğimiz isim.
- Sınıf isimleri CamelCase(kelimelerin birbirine bitişik yazıldığı ve ilk kelime haricinde kalan tüm kelimelerin ilk harfinin büyük yazıldığı bir yazım şeklidir.) şeklinde yazılırlar.
- Sınıfımıza ait kodları süslü parantezler içerisine yazıcaz.

Şimdi ise sınıfımıza öznitelik/özellik/alan(field/feature) ve davranış(method) ekleyelim.

```
public class  Vehicle {
    public String manufacturer;
    public int year;
    public String color;
}
```

- Öznitelik tanımlanırken ilk önce erişim belirteci yazılır burada public kullanmışız sonradan değiştircez, ardından veri tipi en son ise isim veririz.

```
public class  Vehicle {
    //fields
    public String manufacturer;
    public int year;
    public String color;

    //methods
    public void start(){
        System.out.println("Araç başlatıldı.");
    }
    public void stop(){
        System.out.println("Araç durduruldu.");
    }
    public void drive(){
        System.out.println("Araç sürüşte.");
    }
}
```

- Sınıfımıza start(), stop() ve drive() adında 3 tane davranış(method) eklemesi yaptık.
- Normal method yazımı şekilinde. Tabi isteğe bağlı parametre alan ya da geriye değer döndüren metodlarda yazalabiliriz.
- Ancak bu yapılarda şimdilik static keywordü kullanmayalım.

Temel bir sınıf yazdık şimdi bu sınıftan bir nesne(object) üretelim/türetelim. Dikkat etmeniz gereken kısım nesne oluşturma işlemini main metodunun olduğu Main dosyasında oluşturmamız gerekir.

```
public class Main {
    public static void main(String[] args){
        Vehicle vehicle = new Vehicle();
    //      1       2   3   4   5
    }
}
```

- (1)hangi sınıftan nesne oluşturacağımızı belirtir, (2) nesnenin ismi, (3) atama operatörü, (4) yeni bir nesne oluşturmak için kullanılan keyword, (5) sınıfın kurucu metodunu çağırır. (5) Bu yapıya daha sonra bakıcaz.
- Bir nesne oluşturmak için bu yapı kullanılır.

```
public class Main {
    public static void main(String[] args){
        Vehicle vehicle = new Vehicle();
        vehicle.manufacturer = "TOGG";
        vehicle.year = 1453;
        vehicle.color = "Kırmızı";
        System.out.println(vehicle.manufacturer);
        System.out.println(vehicle.year);
        System.out.println(vehicle.color);
    }
}
```

- Nesnenin özelliklerine isminden erişiriz. Şimdilik böyle gösterelim daha sonra nesnenin özelliklerine farklı şekillerde erişim yapıcaz.

Yapıcı Metodlar (Constructor)

- Nesne üretildiği/türetildiğinde zaman kurucu metodlar otomatik olarak çalışan özel metodlardır.
- Yani mantıksal bir varlık fiziksel bir varlığa dönüştüğü zamanda kurucu/yapıcı metodlar çaşlışır.
- Genellikle sınıfla aynı ismi alırlar ve geriye değer döndürmezler. Python da bu işlem `__init__` metodu ile yapılır. Java da sadece sınıf ismi ile yapılır.
- Kurucu metodlar, nesnelerin oluşturulma aşamasında önemli bir rol oynar. Nesnelerin başlatılmasını, kaynakların atanmasını, doğrulamaların yapılmasını ve bağımlılıkların yönetilmesini sağlarlar. Bu sayede, kodun daha okunabilir, güvenilir ve sürdürülebilir olmasına katkıda bulunurlar.

```
public class  Vehicle {
    public Vehicle(){}
    public Vehicle(String manufacturer, int year, String color){
        this.manufacturer = manufacturer;
        this.year = year;
        this.color = color;
    }
}
```

- Kurucu metod default olarak vardır. Bunu yukarıda nesne oluştururken `Vehicle vehicle = new Vehicle()` en sonda yazarak gördük.
- Kurucu metod adı sınıfın adı ile aynı olmak zorundadır.
- Yukarıda yazdığımız gibi nesnenin özniteliklerini kurucu metodun çalışma esnasında alırız. Bu yüzden parametre olarak alınan değerleri sınıfın değerlerine atama işlemi yapıyoruz.

```
public class Main {
    public static void main(String[] args){
        Vehicle vehicle = new Vehicle("TOGG", 1453, "Kırmızı");
        //Kurucu metod sayesinde bu şekilde yapmak yerine değerleri kurucu metodun parametrelerine verdik.
        // vehicle.manufacturer = "TOGG";
        // vehicle.year = 1453;
        // vehicle.color = "Kırmızı";

        System.out.println(vehicle.manufacturer);
```

```
        System.out.println(vehicle.year);
        System.out.println(vehicle.color);
    }
}
```

- Kurucu metodlar da parametre vermeden nesne türetmek adına 2 ya da daha fazla kurucu metod tanımı yapılabilinir. Genelde 2 kurucu metod tanımlanır biri parametreli diğeri parametresiz boş metod olur.

```
public class  Vehicle {
    public Vehicle(){

    }
}
```

Neden metodlarda self kullanırız? Python'da bir sınıfın metodunda self anahtar sözcüğü(java da this), metodun hangi nesneye ait olduğunu belirtmek için kullanılır. Bu sayede, metot içerisindeki kod, o nesnenin özelliklerine ve diğer metotlarına erişebilir.

OOP Temel İlkeler

1. Kapsülleme(Encapsulation)
2. Soyutlama(Abstraction)
3. Kalıtım(Inheritance)
4. Çok Biçimlilik(Poliymorphism)

Bir programlama dilinin Nesneye Yönelimli Programlama (OOP) dili olabilmesi için bu 4 temel ilkeyi bünyesinde barındırması gerekir.

Kapsülleme (Encapsulation)

- Kapsülleme programcıların kodunun belirli bir bölümünü saklamasına ve programın diğer bölümleri tarafından daha az etkileşimli hale getirmesine yardımcı olur.
- Bu programcının daha az hata yapmasına ve kodun daha kolay okunabilir ve anlaşılabilir hale getrimesine yardımcı olur.
- Kapsülleme verilere doğrudan erişim izni verilmeyen bunun yerine verilerin gizlendiği bir süreçtir.
- Bir sınıf içerisindeki private niteliği o sınıf içerisindeki metod ile alıyoruz(getter) ve değiştiriyoruz/güncelleme(setter).

alt text

Getter Methods

- Getter metodlar geriye değer döndürüler. Parametre almazlar.
- Amacı istenilen private niteliği geriye döndürmesidir.
- getName şeklinde isimlendirme jargonu vardır.

Setter Methods

- Setter metodlar parametre alır ve geriye değer döndürmezler.
- Alınan parametreyi sınıfın niteliğine atar.
- Amacı güncelleme/değiştirme işlemi yapmaktır.
- setName seklinde isimlendirme jargonu vardır.

Şimdi bu yapıyı nasıl kullanacağımızı görelim. Vehicle sınıfında 3 tane özellik tanımlamıştık ve bunların erişim belirteçlerini public olarak yazmıştık. OOP yazılımlarda bu yapı kullanılmaz. Sınıfımızın özelliklerine private erişim belirteçlerini vericez. Sonrasında ise getter ve setter metodlarını yazıcaz.

```
public class Vehicle {

    private String manufacturer;
    private int year;
    private String color;
```

```
    public String getManufacturer(){
        return this.manufacturer;
    }
    public void setManufacturer(String manufacturer){
        this.manufacturer = value;
    }

    public int getYear(){
        return this.year;
    }
    public void setYear(int year){
        this.year = value;
    }

    public String getColor(){
        return this.color;
    }
    public void setColor(String color){
        this.color = value;
    }
}
```

Yukarıda getter ve setter yapılarını yazdık. Şimdi ise kullanalım.

```
    public class Main {
        public static void main(String[] args){
            Vehicle vehicle = new Vehicle("TOGG", 1453, "Kırmızı");
            System.out.println(vehicle.getColor());
            vehicle.setColor("Beyaz");
            System.out.println(vehicle.getColor());
        }
        ---Çıktı---
        Kırmızı
        Beyaz
    }
```

- Burada getColor() metodu ile vehicle nesnesinin color özelliği ekrana yazdırılmış.

- Sonra setColor() ile color özelliğinin içeriği değiştirilerek setColor("Beyaz") Beyaz yapılmış.

Soyutlama (Abstraction)

- Soyutlama nesnenin sadece önemli olan kısımlarının görünmesini sağlar. Önemsiz, ayrıntı kısımların görünmesini engeller.
- Soyutlama sadeleşme sağlar ve bilgininde gizlenmesini sağlar.
- Soyutlama için bir örnek: Kahve makinasını bir nesne olarak düşünelim kullanıcılar sadece kahve yapma işlevini düşünür, düğmeye basarak istenilen özelliklerde kahve yapma talimatı veririz, içerideki detaylarla uğraşmaz.
- Daha sonra ayrıntılı görcez.

---

alt text

oluşturduğumuz sınıfın uml diyagramı.

```
public class Vehicle {
    private String manufacturer;
    private int year;
    private String color;

    public String getManufacturer(){
        return this.manufacturer;
    }
    public void setManufacturer(String value){
        this.manufacturer = value;
    }

    public int getYear(){
        return this.year;
    }
    public void setYear(int value){
        this.year = value;
    }
```

```
    public String getColor(){
        return this.color;
    }
    public void setColor(String value){
        this.color = value;
    }

    public Vehicle(){

    }

    public Vehicle(String manufacturer, int year, String color){
        this.manufacturer = manufacturer;
        this.year = year;
        this.color = color;
    }

    public void start(){
        System.out.println("Araç başlatıldı.");
    }
    public void stop(){
        System.out.println("Araç durduruldu.");
    }
    public void drive(){
        System.out.println("Araç sürüşte.");
    }

}
```

oluşturduğumuz sınıfın kaynak kodları.