Java Dilinde Nesne ve Sınıf Kavramı

Bir **nesne** gerçek dünyadaki bir varlığı temsil eder. Örneğin, bir öğrenci, bir masa, bir daire, bir buton vs nesne olarak ifade edilir. **Çünkü** bu varlıkları diğerlerinden ayıran özellikler vardır. O halde nesne eşsizdir.

Nesneni **durumu data field'ler** tarafından, property'ler tarafından temsil edilirken, nesnenin **davranışı** metodlar kümesiyle temsil edilir.
Bir **sınıf** ise nesnenin durumu ve davranışını tanımlayan bir **plan, taslaktır.** Yani sınıfa baktığımız zaman ondan yaratılacak nesnenin nasıl davranacağını, sınıf içindeki metodlara göre anlayabiliriz.

Nesne ile Sınıf arasındaki ilişkiyi şu şekilde örnekleyebiliriz: 50 daireden oluşan bir binanın tasarlanması yapıldıktan sonra bu tasarıma göre dairelerin yapılması işlemini düşünelim. Burada inşaat mühendisi öncelikle binadaki dairelerin tasarımını bir kağıda çizer, daha sonra bu tasarıma göre dairelerın inşaatına başlanır ve bu daireler kullanıma hazır hale getirilir. Nesne tabanlı programa dilinde bu işlemi temsil etmek istersek; İnşaat mühendisi'nin çizdiği tasarım Sınıf, bu tasarımdan oluşan daireler ise nesne olarak ifade edilir. Yani tasarım soyut, tasarımdan oluşturulan yapılar(daireler) ise somuttur. O halde **Sınıf soyut, nesne ise somuttur.**

Örnek:

```
1  public class Circle{
2     private double radius=1.0;
3     public Circle(){
4     }
5     public Circle(double radius){
6        this.radius=radius;
7     }
8     public double getArea(){
9        return radius*radius*Math.PI;
10    }
11 }
```

İkinci satırda tanımlanan radius değişkeni **data field** olarak adlandırılır
Üçüncü satırda **default** constructor tanımlanmıştır
Beşinci satırda parametreli constructor tanımlanmıştır
Diyelim ki bir program yazıyoruz, bu programda sadece yarıçap girip bize dairenin alanını dönderen bir kod lazım. **Circle** isimli bir sınıfımız var. Başka bir sınıf içerisinde **Circle** sınıfında bulunan **getArea()** metodunu çağırmak istiyoruz. Ne yapmamız gerek?
Cevap: **Circle** sınıfından bir tane nesne yaratmalıyız. Sonra **nesneAdi.getArea();** diyerek bu metodu çağırmış oluruz.
Fakat bir sorun var: **getArea()** metodu parametre almıyor. Ben yarıçap değerini nasıl yollayacağım? Hatırlarsak önceki derslerimizde constructor metodlarından bahsetmiştik. Bu metod sınıf ismi ile aynı oluyordu. Parametreli olan constructor ile sınıfların data field'lerine yani değişkenlerine değer atayabiliyorduk. Burada da aynısını yapacağız:

```
1  public class FindCircleArea{
2      public static void main(String[] args){
3          Circle myCircle=new Circle(3.45);//parametreli constructor aracılıyla nesne yaratıldı.
4          double myCircleArea=myCircle.getArea();
5          System.out.println(myCircleArea);
6      }
7  }
```

Hatırlarsak **new** ile yaratılan her şey referans değişkendir. Yani bir pointer'dır. Bu örneğimizde de myCircle bir pointer'dır. **new** Circle() denildiği zaman **Circle** sınıfında tanımlanan tüm kodlar memory'e yükleniyor. Eğer tekrar **new Circle()** dersek yine aynı kodlar memory'nin farklı kısmına yüklenecektir.

UML Diagramı:

Circle Sınıfının UML Diagramı:

Circle
radius: double
Circle() Circle(radius: double) getArea(): double

Nasıl ki C kodları yazmadan önce bir plan yapıyorsak yani şu fonksiyonları kullanayım gibi, Java'da da programı yazmadan önce hangi sınıfları kullanacağımıza ve bu sınıflar içerisinde hangi metodlar, değişkenler(data field) olacağını belirlememiz gerekir. UML diagram bunu sağlamaktadır.

UML Diagram Çizme Adımları:

1) İlk kısma Sınıf adı yazılır
2) İkinci kısma data field(Sınıf değişkenleri) yazılır
3) Üçüncü kısma ise metod isimleri yazılır.
Örnekte görüldüğü gibi **double** ifadesi radius değişkeninin türünün **double** olduğunu ifade eder. **getArea(): double** demek ise bu metodun **double** değer döndereceğini ifade eder.
Not: UML diagram çizerken şuna dikkat etmeliyiz: Bizim yazdığımız kodları bilmeyen bir kullanıcı çizdiğimiz **UML** diagrama göre metodları kullanabilmelidir. Yani **getArea(): double** yerine sadece **getArea()** demiş olsaydık, kullanıcı bu metodun ne tür değer dönderdiğini nasıl anlayacaktı? Bu metodtan dönen değeri **int** değişkeninde mi tutacak yoksa **double** değişkeninde mi tutacak? Fakat **getArea(): double** denildiği zaman kullanıcı anlar ki bu metod **double** bir değer döndermektedir.

Bazı Önemli Sınıflar

Java **JDK** kütüphanesinde binlerce sınıf bulunmaktadır. Bu sınıflardan biri de Date sınıfıdır. Date sınıfında işimize yaracak bir çok metod bulunmaktadır.

Date Sınıfı:

- java.util.Date **import** edilir.
- Zamanı döndermek için **toString()** metodu kullanılır

Random Sınıfı: Hatırlarsak **Math.random()** metodu bize 0 ile 1 arasında **double** değerler üretiyordu. Rastgele sayılar üretmeyi sağlayan bir çok metod Random sınıfında vardır.

Örnek:

```
1  public class RandomGenerator{
2      public static void main(String[] args){
3          Random rnd=new Random(3);//Buradaki 3 seed'tir.
4          rnd.nextInt();//Rastgele bir integer sayi donderir
5          rnd.nextDouble();//Rastgele bir double sayi donderir
6          rnd.nextFloat();//Rastgele bir float sayi donderir
7          rnd.nextBoolean();//Rastgele bir boolean donderir. true veya false donderir
8          rnd.nextLong();//Rastgele bir long deger donderir.
9      }
10 }
```

Sınıfların Data Field'lerini Static Yapmak

Circle sınıfının data field'inde radius değişkeni vardı. Bu değişken **double** radius şeklinde tanımlanmıştı. Bu değişkeni **static** yapmak için yani nesne yaratmadan bu değişkeni **SınıfAdi.degiskenAdi** seklinde kullanabilmek için **double** keyword'ten önce **static** yazmak gereklidir:

double radius;/*nesne yaratildiktan sonra nesneAdi.radius; şeklinde çağrılır.*/

static double radius;/*nesne yaratilmadan **SinifAdi.radius;** şeklinde çağrılır*/

Hatırlarsak **Math.PI** şeklinde bir ifade kullanmıştık. Sizce bu ifade bize ne dönderir?

Math bir sınıf adı

PI değişkenindeki tüm karakterler büyük harfle yazıldı için bir **CONSTANT** değişken demektir. **SinifAdi.PI** şeklinde kullanılmış. O halde **PI** bir **static** değişkendir. Hem **static** hem de constant bir değişken nasıl tanımlarız?

Cevap: **final static double PI=3.141592653589;**

Math sınıfında **PI** sayısı bu şekilde tanımlanmıştır.

Peki neden constant tanımladık? **Çünkü** bir sınıf içerisinde **PI** sayısını kullanırken bu sayıya yeni bir değer atanmasını önlemek için. Yani **Math.PI=3.14;** tarzında kullanılmasını engellemek için **final** keyword ile **PI** değişkeni constant yapılmıştır.

```
1  public class Foo{
2      int i=5;
3      static int k=2;
4      public static void main(String[] args){
5          int j=i; //Hatalı. Çünkü static metod içinde static olmayan bir değişken çağrılamaz. i static degil
6          m();//Hatalı.Çünkü static metod içinde static olmayan bir metod çağrılamaz. m() static degil
7      }
8      public void m(){
9          i=i+k+m2(i,k);//Hata yok. static olmayan bir metod içinde static metod çağrılabilir
10     }
11     public static int m2(int a, int b){
12         return (int)(Math.pow(i,j));
13     }
14 }
```