Nesneye Yönelik Programlama Dillerinde Encapsulation(Kapsülleme)

Nesneye yönelik programlama dillerinin yapıtaşlarından biri olan **encapsulation,** bir sınıfın **data field(sınıf değişkenleri)'lerine** direkt erişimi engellemeyi sağlar.
Circle sınıfının radius değişkenine, nesne yaratıp, nesneAdi.radius şeklinde erişebiliyorduk. Yazdığımız sınıfların daha güvenli ve daha sağlıklı kullanılmasını sağlamak için, encapsulation yapmak gerekir.
Bu özellik, dilin nesne kullanıcısından gereksiz uygulama ayrıntılarını saklayabilme yeteneği olarak ifade edilebilir. Örnek olarak **.Net Framework** temel sınıf kütüphanesi içerisinde yer alan ve **Open()** ve **Close()** metotlarına sahip **SqlConnection** sınıfını ele alalım.

```
1  //SqlConnection, veritabanına yapılan bağlantının detaylarını kapsüllemektedir.
2  SqlConnection baglanti = new SqlConnection("server = London; database = AdventureWorks; integrated security = true");
3  baglanti.Open();
4  //Burada veri yönetilir.
5  baglanti.Close();
```

SqlConnection sınıfı, veritabanına açılan bağlantının sağlanması, yüklenmesi, yönetimi, kapanması gibi içsel detayları gizlemiştir. Nesne kullanıcısı **kapsüllemeyi** sever; **çünkü** programlama görevlerini daha **kolay** hale getirir. **SqlConnection** sınıfında olduğu gibi, nesnenin görevini yerine getirmesi için arka tarafta çalışan onlarca satır kodu düşünmeye gerek yoktur. Tek yapılması gereken nesne örneğinin oluşturulması ve uygun metotların çağrılmasıdır.

Örnek:

```
1  public class Foo{
2      private int i=5; //Başa private keyword getirilerek Encapsulation yapildi.
3      private static int k=2; //Başa private keyword getirilerek Encapsulation yapildi.
4      public static void main(String[] args){
5      }
6  }
```

Nesneleri Metodlara Aktarmak

```
1  public class TestPassObject{
2      public static void main(String[] args){
3          Circle circle=new Circle(5.2);
4          printCircleArea(circle);
5      }
6      public static void printCircleArea(Circle daire){
7          System.out.println("The area of your circle is "+daire.getArea());
8      }
9  }
```

circle nesnesi **Circle** tipinde olduğu için **printCircleArea()** metoduna parametre olarak gönderilebilmiştir. Hatırlarsak, sınıflardan yaratılan nesneler referans değişkenlerdi. **printCircleArea()** metodunun parametresi olan daire referans değişkenine circle nesnesinin tuttuğu adres atanmaktadır. Bu nedenle **printCircleArea** metodu içindeki kodta daire değişkeni circle değişkeni oldu diyebiliriz.

Nesnelerden Oluşan Array Yaratmak

```
1  public class ClassArrays{
2      public static void main(String[] args){
3          double[] sayilar=new double[10]; //10 tane double türünden değişken tutabilir
4          Circle[] daireler=new Circle[10];//10 tane Circle türünden nesneler tutabilir
5          Circle circle1=new Circle(3.2);
6          Circle circle2=new Circle(5.3);
7          daireler[0]=circle1; //nesne atandi
8          daireler[1]=circle2; //nesne atandi
9      }
10 }
```