

Nesne Yönelimli Programlama

OOP Giriş

Hüseyin Ahmetođlu

Java OOP Kavramları

- ▶ Popüler nesne yönelimli diller Java , C # , PHP , Python , C ++ vb.
- ▶ Nesneye yönelik programlamanın temel amacı, nesne, sınıflar, soyutlama, kalıtım, polimorfizm vb. gibi gerçek dünya varlıklarını uygulamaktır.

Java Adlandırma kuralları

- ▶ Java programlama dilinin tüm sınıfları, arayüzleri, paketleri, yöntemleri ve alanları Java adlandırma kuralına göre verilir. Bu kurallara uymazsanız, karışıklık veya hatalı kod oluşturabilir.
- ▶ Standart Java adlandırma kurallarını kullanarak, kodunuzu kendiniz ve diğer programcılar için okumayı kolaylaştırırsınız. Java programının okunabilirliği çok önemlidir. Kodun ne yaptığını anlamak için daha az zaman harcadığınızı gösterir.
- ▶ Bazı kurallarda zorunluluk yoktur.
- ▶ Bu kurallar Sun Microsystems ve Netscape gibi birçok Java topluluğu tarafından önerilmektedir.

Java'da Nesnelere

- ▶ Java'daki bir nesne hem fiziksel hem de mantıksal bir varlık iken, Java'daki bir sınıf yalnızca mantıksal bir varlıktır.
- ▶ Durumu ve davranışı olan her varlık bir nesnedir.
- ▶ Bir nesnenin üç özelliği vardır:
 - **Durum:** bir nesnenin verilerini (değerini) temsil eder.
 - **Davranış:** para yatırma, para çekme vb. Gibi bir nesnenin davranışını (işlevselliğini) temsil eder.
 - **Kimlik:** Nesne kimliği genellikle benzersiz bir kimlik aracılığıyla uygulanır. Kimliğin değeri harici kullanıcı tarafından görülemez. Ancak, JVM tarafından her bir nesneyi benzersiz bir şekilde tanımlamak için dahili olarak kullanılır.

Java'da sınıf

- ▶ Sınıf, ortak özelliklere sahip bir nesne grubudur.
- ▶ Nesnelerin oluşturulduğu bir şablon veya taslaktır.
- ▶ Mantıksal bir varlıktır. Fiziksel olamaz.
- ▶ Java'daki bir sınıf şunları içerebilir:
 - Alanlar
 - Yöntemler
 - Kurucular
 - Bloklar
 - İç içe sınıf ve arayüz

```
sınıf <sınıf_adi> {  
    alan;  
    yöntem;  
}
```

Java'da örnek değişkeni

- ▶ Sınıf içinde ancak yöntemin dışında oluşturulan bir değişken, örnek değişkeni olarak bilinir.
- ▶ Örnek değişkeni derleme zamanında bellekte yer almaz.
- ▶ Bir nesne veya örnek oluşturulduğunda çalışma zamanında bellekte yer alır. Bu yüzden bir örnek değişkeni olarak bilinir.

Java'da Yöntem | Method

- ▶ Java'da yöntem, bir nesnenin davranışını ortaya çıkarmak için kullanılan bir işlev gibidir.
- ▶ Yöntemin Avantajı
 - Kod Yeniden Kullanılabilirliği
 - Kod Optimizasyonu

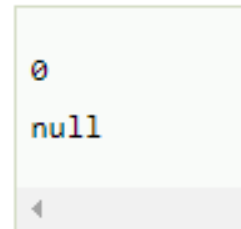
Java'da new anahtar kelimesi

- ▶ new anahtar kelimesi çalışma zamanında bellekte yer ayırmak için kullanılır.
- ▶ Tüm nesnelere Yığın bellek alanında yer alır.
- ▶

Nesne ve Sınıf Örneği: sınıf içindeki main methodu

```
//Java Program to illustrate how to define a class and fields
//Defining a Student class.
class Student{
//defining fields
int id;//field or data member or instance variable
String name;
//creating main method inside the Student class
public static void main(String args[]){
//Creating an object or instance
Student s1=new Student();//creating an object of Student
//Printing values of the object
System.out.println(s1.id);//accessing member through reference variable
System.out.println(s1.name);
}
}
```

Output:



```
0
null
```

Nesne ve Sınıf Örneği: sınıf dışında main methodu

```
//Java Program to demonstrate having the main method in
//another class
//Creating Student class.
class Student{
    int id;
    String name;
}
//Creating another class TestStudent1 which contains the main method
class TestStudent1{
    public static void main(String args[]){
        Student s1=new Student();
        System.out.println(s1.id);
        System.out.println(s1.name);
    }
}
```

Output:

```
0
null
```

Nesneyi yaratmanın 3 yolu

- ▶ Java'da nesne başlatmanın 3 yolu vardır.
 - Referans değişkeni ile
 - Yöntemle
 - Yapıcı tarafından

1) Nesne ve Sınıf Örneği: Referans yoluyla başlatma

```
class Student{
    int id;
    String name;
}
class TestStudent2{
    public static void main(String args[]){
        Student s1=new Student();
        s1.id=101;
        s1.name="Sonoo";
        System.out.println(s1.id+" "+s1.name);//printing members with a white space
    }
}
```

Output:

```
101 Sonoo
```

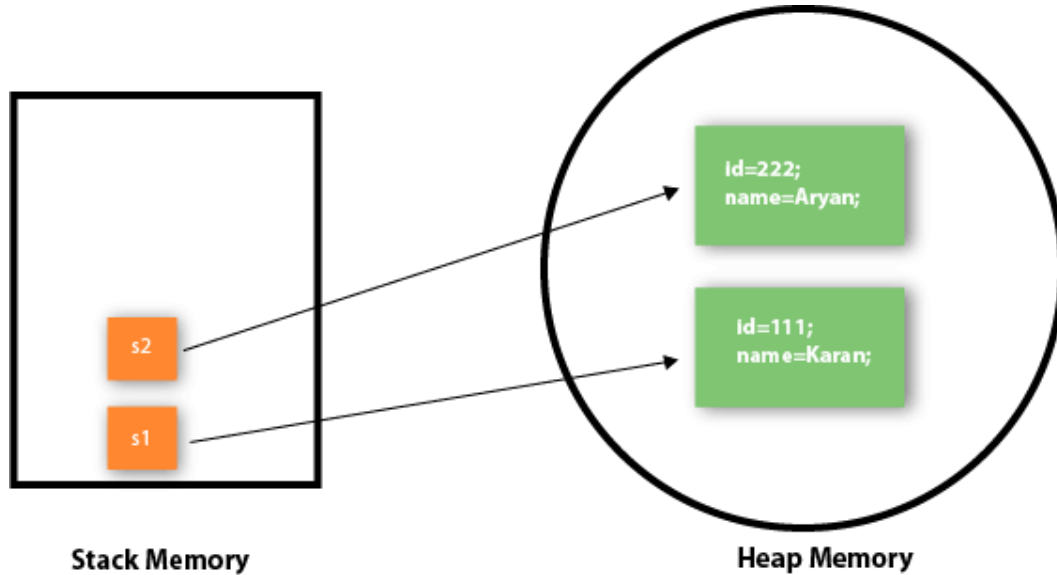
2) Nesne ve Sınıf Örneği: Yöntemle başlatma

```
class Student{
    int rollno;
    String name;
    void insertRecord(int r, String n){
        rollno=r;
        name=n;
    }
    void displayInformation(){System.out.println(rollno+" "+name);}
}
class TestStudent4{
    public static void main(String args[]){
        Student s1=new Student();
        Student s2=new Student();
        s1.insertRecord(111,"Karan");
        s2.insertRecord(222,"Aryan");
        s1.displayInformation();
        s2.displayInformation();
    }
}
```

Output:

```
111 Karan
222 Aryan
```

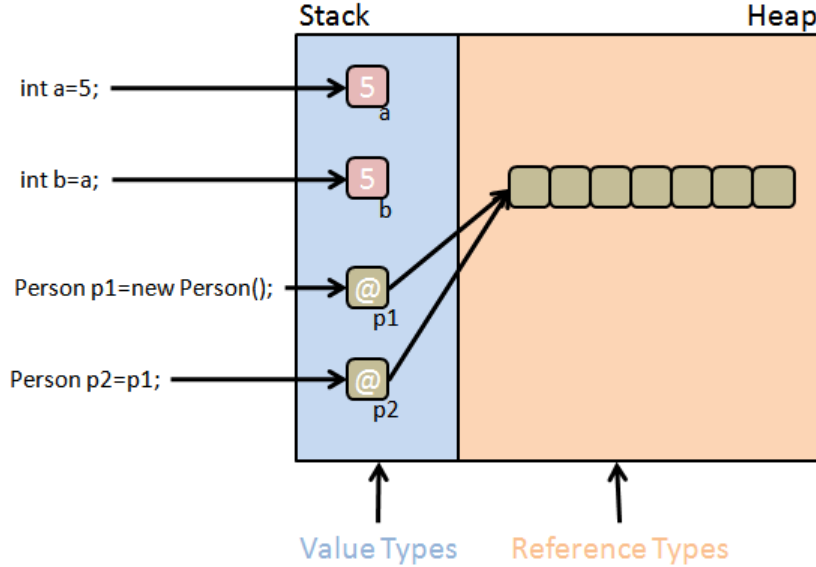
2) Nesne ve Sınıf Örneği: Yöntemle başlatma



Heap ve Stack Bellek

- ▶ **Stack** bellekten statik olarak yer tahsisi için kullanılırken, **Heap** dinamik olarak yer tahsisi içindir. Her ikisi de Ram bölgesinde bulunur.
- ▶ Stack'te yer alan veriler direk bellek içine yerleştirilir dolayısıyla erişimi çok hızlıdır ve programın derleme aşamasında belleğe yerleşirler. **Heap** ise runtime(çalışma zamanında) anında kullanılırlar ve dağınık bir bellek göz yapısı olduğu için erişimi **stack** kadar kolay olmaz dolayısıyla yavaş çalışır. Program esnasında boyutları bildirilmiş, değişmez bir değer kullanacaksak ve buda çok büyük bir veri değilse **stack**, boyutu belli olmayan bir değer kullanıyorsak (ki nesne yönelimli programlama da bunlara obje denir) o zaman derleyici otomatik olarak Heap'ten yer tahsisi yapar. Stack bellekteki veri hemen silinirken Heap bellekteki verinin silinmesi **Garbage Collector'a** (Çöp toplama mekanizmasına) bağlıdır. Stack alanı sınırlı olduğundan çok büyük sayıda ve büyük tiplerde veri atanması belleğin dolmasına sebep olabilir.

Heap ve Stack Bellek



Nesne ve Sınıf Örneği: Çalışan

```
class Employee{
    int id;
    String name;
    float salary;
    void insert(int i, String n, float s) {
        id=i;
        name=n;
        salary=s;
    }
    void display(){System.out.println(id+" "+name+" "+salary);}
}

public class TestEmployee {
    public static void main(String[] args) {
        Employee e1=new Employee();
        Employee e2=new Employee();
        Employee e3=new Employee();
        e1.insert(101,"ajeet",45000);
        e2.insert(102,"irfan",25000);
        e3.insert(103,"nakul",55000);
        e1.display();
        e2.display();
        e3.display();
    }
}
```

Output:

```
101 ajeet 45000.0
102 irfan 25000.0
103 nakul 55000.0
```

Nesne ve Sınıf Örneği: Dikdörtgen

```
class Rectangle{
    int length;
    int width;
    void insert(int l, int w){
        length=l;
        width=w;
    }
    void calculateArea(){System.out.println(length*width);}
}
class TestRectangle1{
    public static void main(String args[]){
        Rectangle r1=new Rectangle();
        Rectangle r2=new Rectangle();
        r1.insert(11,5);
        r2.insert(3,15);
        r1.calculateArea();
        r2.calculateArea();
    }
}
```

Output:

```
55
45
```

Anonim nesne

- ▶ Anonim sadece isimsiz anlamına gelir.
- ▶ Referansı olmayan bir nesne anonim bir nesne olarak bilinir.
- ▶ Yalnızca nesne oluşturma sırasında kullanılabilir.
- ▶ Bir nesneyi yalnızca bir kez kullanmanız gerekiyorsa, anonim bir nesne iyi bir yaklaşımdır.

```
class Calculation{
    void fact(int n){
        int fact=1;
        for(int i=1;i<=n;i++){
            fact=fact*i;
        }
        System.out.println("factorial is "+fact);
    }
    public static void main(String args[]){
        new Calculation().fact(5);//calling method with anonymous object
    }
}
```

Output:

```
Factorial is 120
```

Yalnızca tek bir türle birden çok nesne oluşturma

```
//Java Program to illustrate the use of Rectangle class which
//has length and width data members
class Rectangle{
    int length;
    int width;
    void insert(int l,int w){
        length=l;
        width=w;
    }
    void calculateArea(){System.out.println(length*width);}
}
class TestRectangle2{
    public static void main(String args[]){
        Rectangle r1=new Rectangle(),r2=new Rectangle();//creating two objects
        r1.insert(11,5);
        r2.insert(3,15);
        r1.calculateArea();
        r2.calculateArea();
    }
}
```

Output:

```
55
45
←
```

Gerçek Dünya Örneği: Hesap

```
//Java Program to demonstrate the working of a banking-system
//where we deposit and withdraw amount from our account.
//Creating an Account class which has deposit() and withdraw() methods
class Account{
    int acc_no;
    String name;
    float amount;
    //Method to initialize object
    void insert(int a,String n,float amt){
        acc_no=a;
        name=n;
        amount=amt;
    }
    //deposit method
    void deposit(float amt){
        amount=amount+amt;
        System.out.println(amt+" deposited");
    }
}
```

```
//withdraw method
void withdraw(float amt){
    if(amount<amt){
        System.out.println("Insufficient Balance");
    }else{
        amount=amount-amt;
        System.out.println(amt+" withdrawn");
    }
}
//method to check the balance of the account
void checkBalance(){System.out.println("Balance is: "+amount);}
//method to display the values of an object
void display(){System.out.println(acc_no+" "+name+" "+amount);}
}
//Creating a test class to deposit and withdraw amount
class TestAccount{
    public static void main(String[] args){
        Account a1=new Account();
        a1.insert(832345,"Ankit",1000);
        a1.display();
        a1.checkBalance();
        a1.deposit(40000);
        a1.checkBalance();
        a1.withdraw(15000);
        a1.checkBalance();
    }}
}
```

Output:

```
832345 Ankit 1000.0
Balance is: 1000.0
40000.0 deposited
Balance is: 41000.0
15000.0 withdrawn
Balance is: 26000.0
```

KAYNAKLAR

- ▶ Java Tutorial | Learn Java - javatpoint. (2021, March 21). Retrieved from <https://www.javatpoint.com/java-tutorial>