**JAVA KURULUMU VE EĞİTİMİ**

Java Sun firması tarafından desteklenen ve her platformda çalışan bir dildir. Sun firması tarafından ortaya çıkan daha sonra Oracle tarafından alınan Java hemen hemen tüm platformları desteklenir.

Java eğitimi: Java Tutorial (Oracle sayfasında) Burda Java ile alakalı her türlü bilgiye erişebilirsiniz.

Java http://www.eclipse.org indirilerek Eclipse ortamında Java programlama dili kullanılır. Eclipse için arayüz (IDE) ve Java Development KIT (JDE) indirilmesi gerekiyor. Hangi işletim sistemi kullanıyorsanız kullandığınız işletim sistemine göre indirin.

Netbeans için de Java Developer Kit- JDK indirilir. Aynı işlemler yapılır.

**Örnek 1:**

```
public class insan {

int boy;

int kilo;

int yas;

      void yemek( ) {

        kilo++;

        }

}
```

Burda en basit bir şekilde insan sınıfını oluşturduk. İnsan sınıfının boy, kilo, yas özellikleri vardır. yemek ( ) isimli bir de methodu vardır. C,C++ gibi programlarda fonksiyon olarak adlandırılan yapılar Java'da methodlar olarak tanımlanır ve nesnelerin davranışlarını, eylemlerini, fiillerini gösterir. boy, kilo, yas ise nesnelerin özelliklerini oluşturur.

```
public class insan {

     public static void main (String args [ ] ) {

          insan ali=new ali ( ); //insan sınıfı türünde ali isimli nesne yaratılıyor. Bu nesne sınıf yapısı gibi
//soyut değil Ram'de yer kaplayan bir yer tutan somut özel bir varlıktır.

ali.boy=1.80; //ali nesnesi insan sınıfından yaratıldığı için insan sınıfına ait boy özelliğine değer atanıyor.

ali.kilo= 70; //aynısı kilo ve yaş için de geçerlidir.

ali.yas=30;

ali.yemek( ); //oluşturduğumuz ali nesnesi için insan sınıfına ait yemek methodu çağrılıyor

          System.out.println ( "alinin yaşı"+ ali.yas+ "alinin kilosu"+ ali.kilo+"alinin boyu"+ ali.boy);

//System.out.println C'de printf gibi ekrana bir şey yazdırmak için kullanılır

}

}
```

**Örnek 2:** Aynı örneğe insan sınıfının maaşına belli oranda zam yapan bir method daha ekleyelim. Ayrıca veli isimli insan sınıfından bir nesne daha oluşturalım. Buradaki amaç bir sınıftan istediğimiz kadar nesne oluşturabiliriz.

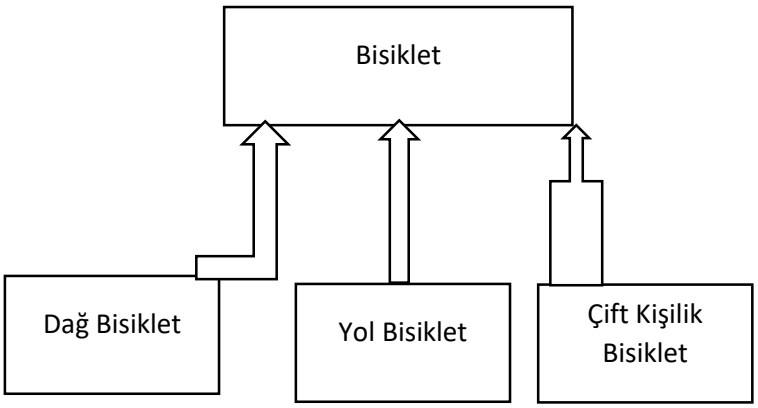
```
public class insan {

int boy;

int kilo;

int yas;

      void yemek( ) {

        kilo++;

        }

        void zam (int oran) {

        maas=maas+maas*zam/100;

       }

}
```

Burda en basit bir şekilde insan sınıfını oluşturduk. İnsan sınıfının boy, kilo, yas özellikleri vardır. yemek ( ) isimli bir de methodu vardır. C,C++ gibi programlarda fonksiyon olarak adlandırılan yapılar Java'da methodlar olarak tanımlanır ve nesnelerin davranışlarını, eylemlerini, fiillerini gösterir. boy, kilo, yas ise nesnelerin özelliklerini oluşturur.

```
public class insan {

     public static void main (String args [ ] ) {

           insan ali=new insan ( ); //insan sınıfı türünde ali isimli nesne yaratılıyor. Bu nesne sınıf yapısı gibi //soyut değil Ram'de yer kaplayan bir yer tutan somut özel bir varlıktır.

insan veli=new insan ( );

ali.boy=1.80; //ali nesnesi insan sınıfından yaratıldığı için insan sınıfına ait boy özelliğine değer atanıyor.

ali.kilo= 70; //aynısı kilo ve yaş için de geçerlidir.

ali.yas=30;

ali.maas=100;

           System.out.println ( "alinin yaşı"+ ali.yas+ "alinin kilosu"+ ali.kilo+"alinin boyu"+ ali.boy);

//System.out.println C'de printf gibi ekrana bir şey yazdırmak için kullanılır

ali.zam(20);

System.out.println ( "alinin maaşı"+ ali.zam );

} }
```

**Kalıtım (Inheritence):** Bir sınıftaki tüm özellik ve methodların diğer sınıfa geçmesidir. Üst sınıftan kalıtım yoluyla alt sınıfa iletilir.

Aşağıdaki örnekte Bisiklet üst sınıftır. Dağ Bisikleti, Yol Bisikleti ve Çift Kişilik Bisiklet alt sınıftır ve Bisiklet sınıfından kalıtım alır. Bisiklet sınıfının tüm özelliklerine ve methodlarına sahiptir. Ancak alt sınıfların sahip olduğu özellikler ve methodlar üst sınıfa aktarılmaz.




**Örnek:**

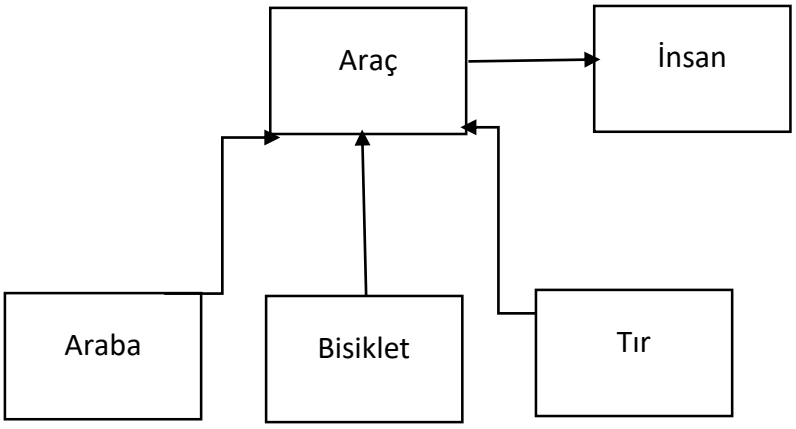
```
public calisan extens insan
{
int maas;
    void zam (int oran) {
     maas=maas+maas*oran/20;
    }
}
public class insan {
int boy;
int kilo;
int yas;
}
     public static void main (String args [ ] ) {
     calisan ali=new calisan ( ); //calisan sınıfından ali nesnesi oluşturuldu.
```

//ali calisan sınıfındadır ve maaşı ile zam mı vardır. Aynı zamanda da ise insan sınıfından kalıtım aldığı için calisan sınıfı insan sınıfına ait boy, kilo, yas özellikleri ile yemek () methodu da vardır.

```
ali.boy=1.80;

ali.yas=30;

ali.kilo=70;

}
```

**Interface (Arayüz):** Nesnelerin birbiriyle haberleşmesini iletişimini sağlar. Interface kullanım amacına örnek verecek olursak; örneğin radar yolunda geçen nesnelerin hızları hesaplanmak istenirse Araba, Tır, Bisiklet sınıflarında oluşan nesnelerin hızları hesaplanır. Ayrıca İnsan sınıfından oluşan nesnelerin de hızları hesaplanmak istenirse, Araç sınıfından kalıtım alan Araba, Tır, Bisiklet sınıfına İnsan sınıfı eklenmez. Aynı özellikler ve methodlar bulunmadığı için İnsan sınıfı interface olarak tanımlanır. Bir sınıfın interface olarak tanımlanması için interface kullanılır. Kalıtım alarak yapılan sınıflanmalarda

UML diyagramı aşağıdaki gibi olur.




**Örnek:**

Aşağıdaki arayüz iki tane metot içermektedir.

```
public interface arayuz01{

    public void topla(int x, int y);

    public void hacim(int x, int y, int z);

}
```

Aşağıdaki arayüz iki tane sabit değişken içermektedir.

```
public interface arayuz02{

    public static final double fiyat = 1250.00;

    public static final int sayac = 5;   }
```

Aşağıdaki sınıf, yukarıdaki iki arayüzü var etmektedir. Sınıf içinde, arayüzün metotlarının serbestçe tanımladığına dikkat ediniz. Aynı arayüzü var edecek başka sınıflar, bu metotları başka başka tanımlayabilirler. Bu, bir metodun farklı işler görmesini sağlar ve polymorphism (çok biçimcilik) diye bilinir.

```
import java.io.*;
class Deneme01 implements arayuz01, arayuz02{

    public void topla(int x, int y){

        System.out.println("x+y = " + (x+y));

    }

    public void hacim(int a, int b, int c){

        System.out.println("Hacim = " + (a*b*c));

    }


    public static void main(String[] args){

        Demo d01 = new Deneme01();

        D01.topla(12,15);

        D01.hacim(3,5,7);

    }
```

**Package:** Java sınıflarının hepsinin aynı yerde bulunmasını sağlar. Aynı package içinde tüm sınıflar yer almazsa program o sınıfları tanımlayamaz. Aynı package olursa java o sınıfı bulur ve compiler eder.

**Constructor (Yapıcı Methodlar ya da inşa methodlar):** Özel tipte fonksiyondur, methodtur.

**Örnek:**

```
class insan {

int boy;

int kilo;

int yas;

public insan () { //public insan( ) constructor tanımlanmış. Constructor sınıfla aynı isim olmak zorunda

System.out.printl (“bir insan nesnesi olabilir”);

}
```

public insan (int boy, int kilo, int yas ) { //parametre alınan constructor polymorphism (çok biçimcilik)

//oluşmuştur. İki constructor var biri parametre alan diğeri almayan constructor dur.

this.boy=boy;

this.kilo=kilo;

this.yas=yas;

}

Public static void main (String [] args) {

insan ali=new ali (1.70, 70, 20); //default constructor. New özelliği ile constructor u çağırıyor

}

this ifadesi ali nesnesinin main fonksyionu içerisinde aldığı değerlerin alınmasını sağlar. Constructor her nesne oluşturulduğunda çalıştırılan fonksiyonlardır. Bir nesne oluşturulduğunda new ile constructor ‘un çağrılması sağlanır.