**Java Ders Notlarım**

**JVM(Java Virtual Machine)=**Java kodlarını bütün işletim sistemlerinde kullanabilmemiz için kodları Java Byte Code’e çevirir, .java uzantılı dosyaları değilde .class uzantılı okuyabilir.Bunun sayesinde “Kodu bir kere yaz,bütün işletim sistemlerinde çalıştır” sözünün mantığını anlıyoruz.

**JRE(Java Runtime Environment)=**JVM sayesinde java kodlarından dönüştürülmüş byte code’leri diğer makinalarda kullanmamızı sağlar.Bunun yanısıra bazı temel kütüphaneleride barındır.

**JDK(Java Development Kit)=**Java dili ile geliştirme yapamız içi gereken araçları içeren pakettir.

KISACA: JRE=JVM + Java Kütüphaneleri JDK=JRE + Compiler + debugger

**///Klavyeden Veri Alma///**

Scanner scan = new Scanner(System.in);

System.out.println("Klavyeden Veri Alma");

int a=scan.nextInt();

System.out.println(a);

System.err.println (“ “)-- (Kırmızı yazdırma)

\*\*\*/// System.err.println("Klavyeden Veri Alma");\*\*\*

--- Klavyeden Veri Alma---

**///MATH METODLARI///**

**///Math.sqrt ve Kök Alma///**

double x = 9;

Math.sqrt(x);

System.out.println(Math.sqrt(x));

yada

double x=Math.sqrt(9);

System.out.println(x); Çıktı: --3--

**///Math.ceil ve bir double sayıyı üst sayıya yuvarlama///**

double sayi= Math.ceil(5.2);

System.out.println(sayi); yada

double sayi=5.2;

Math.ceil(sayi);

System.out.println(Math.ceil(sayi));

--6—

**///Math.pow ve ikinci parametreyi üs alan sınıf///**

double sayi=Math.pow(5,3);

System.out.println(sayi);

5.5.5= ---125---

**///Math.Max/Math.Min ve İki sayıdan büyüğünü yada küçüğünü alan sınıf///**

double sayi=Math.max(76,66);

System.out.println(sayi);

---76---

double sayi=Math.min(76,66);

System.out.println(sayi);

---66---

**///Math.Random ve rastgele sayı döndüren sınıf///**

double sayi=Math.random();

System.out.println(sayi);

---rastgele bir sayı---

double sayi=Math.random() **\* 100**;

System.out.println(sayi);

---0-100 arasında rastgele bir sayı---

double sayi=**100 +** Math.random() **\* 1000**;

System.out.println(sayi);

---100 ile 1000 arasında rastgele bir sayı---

**///Parametresiz Değer Döndürmeyen Metodlar///**

\*Void=Değer Döndürmeyen,bir sonuç üretmeyen

\*Public=Herkese açık metod

\*Private=Coddera özel metod

**public** **static** **void** main(String[] args)

{

*yazdir*();

}

**public** **static** **void** yazdir()

{

System.***out***.println("Samet");

System.***out***.println("Çavur");

System.***out***.println("Akdeniz Üniversitesi");

System.***out***.println("Yönetim ve Bilişim Sistemleri");

System.***out***.println("20184753075");

}

---

Samet

Çavur

Akdeniz Üniversitesi

Yönetim ve Bilişim Sistemleri

20184753075

---

**///Parametreli Değer Döndürmeyen Metodlar///**

**public** **static** **void** main(String[] args)

{

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("İlk Sayi");

**int** a=scan.nextInt();

System.***out***.println("İkinci Sayi");

**int** b=scan.nextInt();

*topla*(a, b);

}

**public** **static** **void** topla(**int** a,**int** b)

{

**int** toplam=a+b;

System.***out***.print("Toplamı="+toplam);

}

---

İlk Sayi

44

İkinci Sayi

44

Toplamı=88

---

\***public** **static** **void** topla(**int** a,**int** b)>>buraya int c,int d ekleyerek parametreyi arttırabiliriz.

**///Parametresiz Değer Döndüren Metodlar///**

\*Void yerine metodun döndüreceği sonuç değişkeninin türünü yazacağız.void>int,double,str.

\*Değer döndüren metodlarda sona her zaman return yazarız.

**public** **static** **void** main(String[] args)

{

System.***out***.println(*topla*());

System.***out***.println(*carp*());

}

**public** **static** **int** topla()

{

**int** a=10;

**int** b=20;

**int** toplam=a+b;

**return** toplam;

}

**public** **static** **double** carp()

{

**double** a=10.5;

**double** b=50.123;

**return** a\*b;

}

---

30

526.2914999999999

---

**///Parametreli Değer Döndüren Metodlar///**

**public** **static** **void** main(String[] args)

{

System.***out***.println(*topla*(50, 60));

}

**public** **static** **int** topla(**int** a,**int** b)

{

**return** a+b;

}

**ÖRNEK**

**Kullanıcı adı ve Parola doğru ise true yanlış ise false döndüren metod**

**public** **static** **void** main(String[] args)

{

**boolean** durum =*ıdpass*();

System.***out***.print(durum);

}

**public** **static** **boolean** ıdpass(){

Scanner scan = **new** Scanner(System.***in***);

**boolean** ıdpass;

System.***out***.print("Kullanıcı Adı Giriniz");

**int** ıd=scan.nextInt();

System.***out***.print("Password Adı Giriniz");

**int** pass=scan.nextInt();

**if**(ıd==654321 && pass==123456) {

**return** **true**;

}

**else** {

**return** **false**;

}

}

**ÖRNEK**

**Metodlarla yapılmış bir hesap makinesi**

**public** **static** **void** main(String[] args)

{

Scanner scan = **new** Scanner (System.***in***);

**int** a,b;

**int** islem;

System.***out***.println("İlk Sayiyi Giriniz.");

a = scan.nextInt();

System.***out***.println("İkinci Sayiyi Giriniz.");

b = scan.nextInt();

System.***out***.println("Yapmak İstediğiniz İşlemi Seçiniz.");

System.***out***.println("1-Topla");

System.***out***.println("2-Çıkar");

System.***out***.println("3-Böl");

System.***out***.println("4-Çarp");

islem = scan.nextInt();

**if**(islem==1) {

System.***out***.println(*topla*(a , b));

}

**else** **if**(islem==2) {

System.***out***.println(*cikar*(a , b));

}

**else** **if**(islem==3) {

System.***out***.println(*bol*(a , b));

}

**else** {

System.***out***.println(*carp*(a , b));

} }

**public** **static** **int** topla(**int** a,**int** b) {

**int** c = a + b;

**return** c;

}

**public** **static** **int** cikar(**int** a,**int** b) {

**int** c = a - b;

**return** c;

}

**public** **static** **double** bol(**int** a,**int** b) {

**int** c = a / b;

**return** c;

}

**public** **static** **int** carp(**double** a,**double** b) {

**int** c = (**int**) (a \* b);

**return** c; }

**///DİZİLER///**

**int** [] dizi = **new** **int**[3];

dizi [0]=15;

dizi [1]=4;

dizi [2]=3;

\*3 elemanlı bir dizi

**///Hazır Elemanlı Diziler///**

String [] gunler = {"Pazartesi","Salı","Çarşamba"};

System.***out***.print(gunler[2]);

---Çarşamba---

**///Dizi elemanını kullanıcıdan alma///**

**public** **static** **void** main(String[] args)

{

Scanner scan=**new** Scanner(System.***in***);

String a,b,c,d,e,f,g;

String [] gunler = **new** String[7];

System.***out***.println("günleri yaz");

a=scan.next();

gunler[0]=a;

b=scan.next();

gunler[1]=b;

c=scan.next();

gunler[2]=c;

d=scan.next();

gunler[3]=d;

e=scan.next();

gunler[4]=e;

f=scan.next();

gunler[5]=f;

g=scan.next();

gunler[6]=g;

System.***out***.println(gunler[0]);

System.***out***.println(gunler[1]);

System.***out***.println(gunler[2]);

System.***out***.println(gunler[3]);

System.***out***.println(gunler[4]);

System.***out***.println(gunler[5]);

System.***out***.println(gunler[6]); }

**ÖRNEK**

**Kullanıcının sayısını belirlediğini diziye sayılar atayıp sayıların toplamını ve ortalamasını aldıran ve sayıları metodla sıralayan program**

\*\*\*lenght=dizideki eleman adedini verir.

\*\*\*size=dizinin boyutunu verir.(bunu adedi belli olmayan dizilerde(Array List)kullanacağız.

**public** **static** Scanner *scan* = **new** Scanner(System.***in***);

**public** **static** **void** main(String[] args) {

System.***out***.println("Kaç boyutlu bir dizi oluşturucaksınız");

**int** a=*scan*.nextInt();

**int** dizi[] = **new** **int**[a];

**int** bolum=0;

**int** toplam=0;

**for**(**int** i=0 ; i < a ; i++) {

System.***out***.print((i+1)+"inci sayı=");

**int** b = *scan*.nextInt();

dizi[i]=b;

toplam+=b;

bolum=toplam/a;

}

*diziSirala*(dizi);

System.***out***.println("Sayıların toplamı="+toplam);

System.***out***.print("Sayıların ortalaması="+bolum);

}

**public** **static** **void** diziSirala (**int** s[]) {

**for**(**int** i=0;i<s.length;i++) {

System.***out***.println("Dizinin"+(i+1)+ "inci elemanı= "+s[i]);

}}

**///ÇOK BOYUTLU DİZİLER///**

System.***out***.println("Satır Sayısını Gir");

**int** x = *scan*.nextInt();

System.***out***.println("Sutun Sayısını Gir");

**int** y = *scan*.nextInt();

**int** cokBoyutluDizi[][] = **new** **int** [x][y];

**///For Döngüsü ile Çok boyutlu bir diziye sayı atayıp yazdırma///**

**public** **static** Scanner *scan* = **new** Scanner(System.***in***);

**public** **static** **void** main(String[] args)

{

System.***out***.println("Satır Sayısını Gir");

**int** x = *scan*.nextInt();

System.***out***.println("Sutun Sayısını Gir");

**int** y = *scan*.nextInt();

**int** cokBoyutluDizi[][] = **new** **int** [x][y];

**for**(**int** i = 0 ; i < cokBoyutluDizi.length ; i++) {

**for**(**int** j = 0 ; j < cokBoyutluDizi[i].length ; j++) {

cokBoyutluDizi[i][j]=27;

}

}

**for**(**int** i = 0 ; i < cokBoyutluDizi.length ; i++) {

**for**(**int** j = 0 ; j < cokBoyutluDizi[i].length ; j++) {

System.***out***.println(cokBoyutluDizi[i][j]);

**/// DİZİ KOPYALAMA ///**

\*\*\*System.*arraycopy*(dizi1, 4, dizi2, 3, 2)>>>dizi1’in 4. İndeksinden 2 veri kopyala ve dizi2 in 3.indeksinden sonra yapıştır.

ÖRNEK

{

**int** dizi1[]= {9,1,2,7,3,4,6,5,2,0};

**int** dizi2[]=**new** **int**[dizi1.length];

System.*arraycopy*(dizi1, 4, dizi2, 3, 2);

**for**(**int** i=0 ; i<dizi2.length ; i++) {

System.***out***.println(dizi2[i]);

}}

--- 0 0 0 3 4 0 0 0 0 0 ---

**/// DİZİ SIRALAMA ///**

Arrays.*sort*(dizi); >> Diziyi küçükten büyüğe doğru sıralayan metod.

**int** dizi[]= {9,12,2,74,3,4,63,5,32,0};

Arrays.*sort*(dizi);

**for**(**int** i=0 ; i<dizi.length ; i++) {

System.***out***.println(dizi[i]);

**/// ARRAY LİST (GELİŞMİŞ-DİNAMİK DİZİLER///**

İçerisindeki veri sayısı bilinmeyen dizileri,sonradan içindeki verileri çıkarmak eklemek yada değiştirmek istediğimiz dizileri array list ile yapıyoruz.

Add ile diziye veri ekleme

Remove ile diziden veri silme

Get ile diziden veri çekme

Set ile dizideki veriyi değiştirme

İndexOf ile dizideki verinin indeks numarasını alma işlemlerini yaparız.

Contains ile aradığınız veri dizide var mı yok mu onu kontrol edebiliriz.

ArrayList<String> Derslistesi = **new** ArrayList<>();

Derslistesi.add("Matematik");

Derslistesi.add("Fizik");

Derslistesi.add("Beden Eğitimi");

Derslistesi.add("Müzik");

**for** (String i : Derslistesi) {

System.***out***.println(i);

---

Matematik Beden Eğitimi

Fizik Müzik ---

**///ÖRNEK UYGULAMA///**

İçindeki veri sayısı belli olmayan bir diziye 3 farklı eleman atayacağız,sonrasında bu verilerin kökü alınmış haliyle değiştirip yazdıracağız.

ArrayList<Integer> sayiDizisi = **new** ArrayList<>();

sayiDizisi.add(9);

sayiDizisi.add(16);

sayiDizisi.add(25);

**for**(**int** i : sayiDizisi) {

System.***out***.println(i);

}

System.***out***.println("---");

**int** x;

x=sayiDizisi.get(0);

sayiDizisi.set(0,(**int**) Math.*sqrt*(x));

**int** y;

y=sayiDizisi.get(1);

sayiDizisi.set(1,(**int**) Math.*sqrt*(y));

**int** z;

z=sayiDizisi.get(2);

sayiDizisi.set(2,(**int**) Math.*sqrt*(z));

**for**(**int** i : sayiDizisi) {

System.***out***.println(i);

---

9

16

25

---

3

4

5

---

**///String Sınıfı Metodları///**

String isimSoyisim = "Samet Çavur";

String okulAdi = "Akdeniz Üniversitesi";

String deneme = " Samet ÇavÇav";

**char** a = isimSoyisim.charAt(3);

System.***out***.println(" 3.indekste hangi harf var =" +a);

3.indekste hangi harf var =e

**int** b = isimSoyisim.indexOf("avu");

System.***out***.println("avu harfleri kaçıncı indeksten sonra başlıyor= " + b);

avu harfleri kaçıncı indeksten sonra başlıyor= 7

String c = isimSoyisim.replace("a", "i");

System.***out***.println("A yerlerine İ yazdırılmış hali =" + c);

A yerlerine İ yazdırılmış hali =Simet Çivur

String d = isimSoyisim.toLowerCase();

System.***out***.println("Kelimeyi küçük harflerle yazdırma =" + d);

Kelimeyi küçük harflerle yazdırma =samet çavur

String e = isimSoyisim.toUpperCase();

System.***out***.println("Kelimeyi büyük harflerle yazdırma= "+ e);

Kelimeyi büyük harflerle yazdırma= SAMET ÇAVUR

String f = isimSoyisim.substring(3 ,9);

System.***out***.println("İndeks numaralarını verdiğimiz aralıktaki harfleri çıkarır=" + f);

İndeks numaralarını verdiğimiz aralıktaki harfleri çıkarır=et Çav

Boolean g = isimSoyisim.contains("Samet");

System.***out***.println("Samet kelimesi cümlede var mı kontrol ediyoruz=" + g);

Samet kelimesi cümlede var mı kontrol ediyoruz=true

Boolean h = isimSoyisim.equals(okulAdi);

System.***out***.println("İki cümle aynı değil mi?Aynı ise true,değilse false="+ h);

İki cümle aynı değil mi?Aynı ise true,değilse false=false

Boolean j = isimSoyisim.equals(isimSoyisim);

System.***out***.println("İki cümle aynı değil mi?Aynı ise true,değilse false="+ j);

İki cümle aynı değil mi?Aynı ise true,değilse false=true

String yeniCumle = isimSoyisim.concat(" " + okulAdi);

System.***out***.println("İki cümleyi birleştiriyoruz =" + yeniCumle);

İki cümleyi birleştiriyoruz =Samet Çavur Akdeniz Üniversitesi

String k = deneme.trim();

System.***out***.println("Cümlenin başındaki boşlukları siliyoruz =" +k);

Cümlenin başındaki boşlukları siliyoruz =Samet ÇavÇav

**char** isimDizisi[] = isimSoyisim.toCharArray();

**for**(**char** i : isimDizisi) {

System.***out***.println(i);

**char** isimDizisi[] = isimSoyisim.toCharArray();

**for**(**char** i : isimDizisi) {

System.***out***.println(i);

--- S Ç

a a

m v

e u

t r ---

CharArray ile yeni bir char dizisi üretip kelimeyi yukardan aşağıya yazdırdık.

**///İnteger değerleri Stringe Çevirme(Integer.toString) ve**

**String değerleri İntegera Çevirme(Integer.parseInt-Double.parseDouble)///**

**int** a = 5;

**int** b = 10;

String stringA = Integer.*toString*(a);

String stringB=Integer.*toString*(b);

System.***out***.println(stringA+stringB);

**int** ıntA=Integer.*parseInt*(stringA);

**int** ıntB=Integer.*parseInt*(stringB);

**int** c = 25;

System.***out***.println(ıntA+ıntB+c);

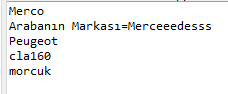
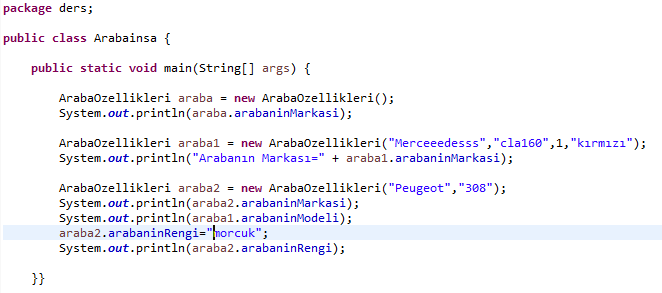
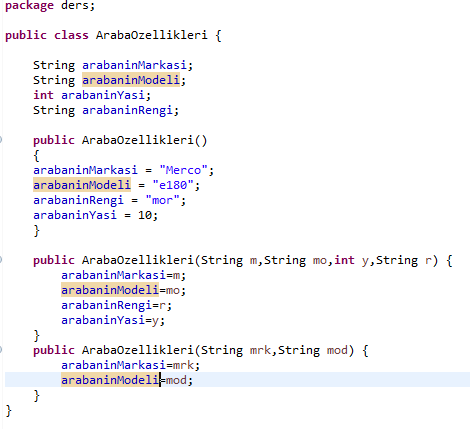
---

510

40

---

**///CONSTUCTOR(YAPILANDIRICI METOTLAR)///**



**///Erişim Belirteçleri ///**

Public=Herkese açık classlar

Private=Gizli class,herhangi bir paketten yada classtan ulaşamayız.

Friendly=Hiçbir şey tanımlamadığımız classlardır.Aynı paket içindeki classlardan ulaşılabilir.

Protected=Korumalı.Başka paketten ulaşılamaz.

**///Kapsülleme (Encapsulation)///**

Kapsülleme kullanıcıdan bazı şartlar altında bir şey istediğinde kullanılır.Örneğin bir isim text boxuna sayı girilmesini engeller.Paketin içindeki kapsüllemek istediğimiz değişkenin başına private yazıyoruz.Bu şekilde dışardan erişime engelliyoruz.Dışarıdan bazı kısıtlamalar ile almak içinse set/get metodları ile kısıtlayarak public classların içine koyuyoruz.

**private** String isim;

**public** String getIsim() {

**return** **this**.isim;

}

**public** **void** setIsim(String isim) {

**if**(isim.length()<3) {

System.***out***.println("Geçersiz bir isim kullandın.İsmin Kullanıcı olarak değiştirildi");

**this**.isim="Kullanıcı";

}

**else** {

**this**.isim=isim;

}

}

MAİN CLASS

ogrenci ogr1 = **new** ogrenci();

ogr1.setIsim("Metin");

System.***out***.println(ogr1.getIsim());

**///Set,Get,This///**

Üst Örnekte var.

Set aslında metodun ismidir ve kapsüllenmiş classa veri yollamak için kullanılıyor

Get ise kapsüllenmiş classtan veriyi almak için kullanıyoruz.

This ise bir paketin içindeki değişken ile classın içindeki değişken ismi aynı olduğunda ikisininde nereye ait olduğunu belli etmek için kullanıyoruz.başına this gelen değişken paketin,gelmeyen ise classın değişkenidir.

**///Static Sınıflar///**

İskeleti oluşturduğumuz classın içerisinde bir kere tanımlıyoruz ve bütün classlarda ortak olarak kullanıyoruz.Aşağıda okulun Akdeniz olduğunu bir kere tanımladık ve main classımızda hangi nesnede çağırırsak çağıralım aynı sonucu verdi.

**private** **static** String *okul*;

**public** **static** **void** setOkul(String okul) {

ogrenci.*okul*="Akdeniz Üniversitesi";}

**public** **static** String getOkul() {

**return** ogrenci.*okul*;

MAİN CLASS

ogrenci ogr1 = **new** ogrenci();

ogrenci ogr2 = **new** ogrenci();

System.***out***.println(ogr1.*getOkul*());

System.***out***.println(ogr2.*getOkul*());

**///To.String Metodu///**

İskeleti oluştuğumuz classın içine toStr+ctrl+space basınca otomatik oluşturuyor.Return’un içine de çağırmamızı istediğimiz değişkenleri yazıyoruz.Ve main classta sadece nesnemizi syso içinde çağırınca returnda istediğimiz tüm veriler geliyor.

**public** String toString() {

**return** "İsminiz="+ isim + ",Soyisminiz=" + soyisim + ",Okulunuz=" + *getOkul*() + ",OkulNo="+ okulno + ",Yaşınız=" + yas ;

}

MAİN CLASS

ogrenci ogr1 = **new** ogrenci("Samet","Çavur","1234",25);

System.***out***.println(ogr1);

--- İsminiz=Samet,Soyisminiz=Çavur,Okulunuz=Akdeniz Üniversitesi,OkulNo=1234,Yaşınız=25

---

**///Final Metodu///**

Değişkenin türüne bir kere final yazılınca değiştirilemez anlamına gelir.Binevi son noktayı koyduk artık budur demek gibi bir şey.

**private** **static** **final** String ***okul*** = "Akdeniz Üniversitesi";

**///ÖRNEK UYGULAMA///**

public class ogrenci {

private static String okul="Akdeniz Üniversitesi";

private String isim;

private String soyisim;

private String okulno;

private ArrayList<String> dersler;

public String toString() {

return "ogrenci [isim=" + isim + ", soyisim=" + soyisim + ", okulno=" + okulno + ", dersler=" + dersler + "]";

}

public ogrenci() {

}

public ogrenci(String isim,String soyisim,String okulno,ArrayList<String>dersler) {

this.isim=isim;

this.soyisim=soyisim;

this.okulno=okulno;

this.dersler=dersler;

}

public void setIsim(String isim) {

this.isim=isim;

}

public String getIsim() {

return isim;

}

public void setSoyisim(String soyisim) {

this.soyisim=soyisim;

}

public String getSoyisim() {

return soyisim;

}

public void setOkulno(String okulno) {

this.okulno=okulno;

}

public String getOkulno() {

return okulno;

}

public void setDersler(ArrayList<String> dersler) {

this.dersler=dersler;

}

public ArrayList<String> getDersler(){

MAİN CLASS

**public** **class** ogrenciTest {

**public** **static** **void** main(String[] args) {

ogrenci ogr1 = **new** ogrenci();

ogr1.setIsim("Samet");

ogr1.setSoyisim("Çavur");

ogr1.setOkulno("1234");

ArrayList<String> ogr1ders = **new** ArrayList<>();

ogr1ders.add("Matematik");

ogr1ders.add("Fizik");

ogr1ders.add("Kimya");

ogr1.setDersler(ogr1ders);

ogrenci ogr2 = **new** ogrenci("Metin","Taşmaz","4321",**null**);

ArrayList<String> ogr2ders = **new** ArrayList<String>();

ogr2ders.add("İngilizce");

ogr2ders.add("Fransızca");

ogr2ders.add("İspanyolca");

ogr2.setDersler(ogr2ders);

System.***out***.println(ogr1);

System.***out***.println(ogr2);

*dersAra*(ogr1,"Matematik");

}

**public** **static** **void** dersAra(ogrenci o,String d) {

ArrayList<String> dersler = o.getDersler();

**int** x=1;

**for**( String ders : dersler) {

**if**(d.equals(ders)) {

System.***out***.println("Öğrenci bu dersi alıyor.");

x=0;

}

}

**if**(x==1) {

System.***out***.println("Öğrenci bu dersi almıyor.");

}

}

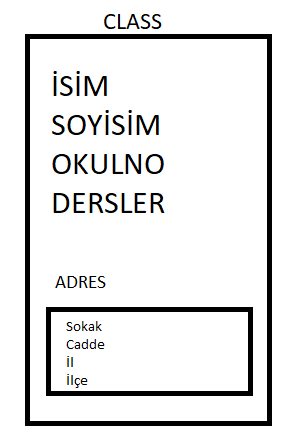
---

ogrenci [isim=Samet, soyisim=Çavur, okulno=1234, dersler=[Matematik, Fizik, Kimya]]

ogrenci [isim=Metin, soyisim=Taşmaz, okulno=4321, dersler=[İngilizce, Fransızca, İspanyolca]]

Öğrenci bu dersi alıyor. ---

**///NESNELER ARASI İLİŞKİ (ASSOCATİON)///**

****

Assocation mantığı iki classı birleştirip main classta

Kullanmaktır.Bunu yapma sebebimiz yandaki resimden anlaşılır.İlk önce isim soyisim okulno dersler gibi verileri aldığımız class oluşturuyoruz.Sonrasında adresi alacağımız classı oluşturuyoruz.Sonra ikisini main classta kullanıyoruz.

public class ogrenci {

private String isim;

private String soyisim;

private String okulno;

private ArrayList<String> dersler;

private adress adres;

Öğrenciyi yarattığımız class

public ogrenci() {

}

public ogrenci(String isim, String soyisim, String okulno, ArrayList<String> dersler, adress adres) {

this.isim = isim;

this.soyisim = soyisim;

this.okulno = okulno;

this.dersler = dersler;

this.adres = adres;

}

public String getIsim() {

return isim;

}

public void setIsim(String isim) {

this.isim = isim;

}

public String getSoyisim() {

return soyisim;

}

public void setSoyisim(String soyisim) {

this.soyisim = soyisim;

}

public String getOkulno() {

return okulno;

}

public void setOkulno(String okulno) {

this.okulno = okulno;

}

public ArrayList<String> getDersler() {

return dersler;

}

public void setDersler(ArrayList<String> dersler) {

this.dersler = dersler;

}

public adress getAdres() {

return adres;

}

public void setAdres(adress adres) {

this.adres = adres;

}

@Override

public String toString() {

ADRESS CLASS

public class adress {

private String sokak;

private String cadde;

private String il;

private String ilce;

public adress() {

}

public adress(String sokak, String cadde, String il, String ilce) {

this.sokak = sokak;

this.cadde = cadde;

this.il = il;

this.ilce = ilce;

}

public String getSokak() {

return sokak;

}

public void setSokak(String sokak) {

this.sokak = sokak;

}

public String getCadde() {

return cadde;

}

public void setCadde(String cadde) {

this.cadde = cadde;

}

public String getIl() {

return il;

}

public void setIl(String il) {

this.il = il;

}

public String getIlce() {

return ilce;

}

public void setIlce(String ilce) {

this.ilce = ilce;

}

@Override

public String toString() {

return "adress [sokak=" + sokak + ", cadde=" + cadde + ", il=" + il + ", ilce=" + ilce + "]";

}

MAİN CLASS

**public** **class** ogrenciTest {

**public** **static** **void** main(String[] args) {

ogrenci ogr1 = **new** ogrenci();

adress adres = **new** adress("113/1 sk","Rauf Denk Cd","İst","Zb");

ogr1.setAdres(adres);

ogr1.setIsim("Samet");

ogr1.setSoyisim("Çavur");

ogr1.setOkulno("123");

ArrayList<String> dersler= **new** ArrayList<>();

dersler.add("Mate");

dersler.add("fizik");

ogr1.setDersler(dersler);

System.***out***.println(ogr1);

ÇIKTI

--- ogrenci [isim=Samet, soyisim=Çavur, okulno=123, dersler=[Mate, fizik], adres=adress [sokak=113/1 sk, cadde=Rauf Denk Cd, il=İst, ilce=Zb]]---

**///CONSTRUCTOR,ENCAPSULATİON ve ASSOCATİON İÇEREN UYGULAMA///**

Bu örnekte öğrenci tanımladık ve adres ile aldığı kursları farklı bi classtan alarak assocation ile yaptık,İsim ve Soyisim 3 harften küçük olmaması için kapsülledik.

**ADRES CLASS**

public class adres {

private String sokak;

private String mahalle;

private String ilce;

private String il;

public adres(){

}

public adres(String sokak, String mahalle, String ilce, String il) {

this.sokak = sokak;

this.mahalle = mahalle;

this.ilce = ilce;

this.il = il;

}

public String getSokak() {

return sokak;

}

public void setSokak(String sokak) {

this.sokak = sokak;

}

public String getMahalle() {

return mahalle;

}

public void setMahalle(String mahalle) {

this.mahalle = mahalle;

}

public String getIlce() {

return ilce;

}

public void setIlce(String ilce) {

this.ilce = ilce;

}

public String getIl() {

return il;

}

public void setIl(String il) {

this.il = il;

}

public String toString() {

return "[sokak=" + sokak + ", mahalle=" + mahalle + ", ilce=" + ilce + ", il=" + il + "]";

}

**KURS CLASS**

public class kurslar {

private String kursAdi;

private String kursOgr;

private int kursSaati;

public kurslar(){

}

public kurslar(String kursAdi, String kursOgr, int kursSaati) {

this.kursAdi = kursAdi;

this.kursOgr = kursOgr;

this.kursSaati = kursSaati;

}

public String getKursAdi() {

return kursAdi;

}

public void setKursAdi(String kursAdi) {

this.kursAdi = kursAdi;

}

public String getKursOgr() {

return kursOgr;

}

public void setKursOgr(String kursOgr) {

this.kursOgr = kursOgr;

}

public int getKursSaati() {

return kursSaati;

}

public void setKursSaati(int kursSaati) {

this.kursSaati = kursSaati;

}

public String toString() {

return "[kursAdi=" + kursAdi + ", kursOgr=" + kursOgr + ", kursSaati=" + kursSaati + "]";

}

**ÖĞRENCİ CLASS**

public class ogrenci {

private static String okul = "Akdeniz Üniversitesi";

private String isim;

private String soyisim;

private String okulno;

private int yas;

private adres adres;

private ArrayList<kurslar> kurslar;

public ogrenci(){

}

public ogrenci(String isim, String soyisim, String okulno, int yas, ders.adres adres,ArrayList<ders.kurslar> kurslar) {

this.isim = isim;

this.soyisim = soyisim;

this.okulno = okulno;

this.yas = yas;

this.adres = adres;

this.kurslar = kurslar;

}

public adres getAdres() {

return adres;

}

public void setAdres(adres adres) {

this.adres = adres;

}

public ArrayList<kurslar> getKurslar() {

return kurslar;

}

public void setKurslar(ArrayList<kurslar> kurslar) {

this.kurslar = kurslar;

}

public static String getOkul() {

return okul;

}

public static void setOkul(String okul) {

ogrenci.okul = okul;

}

public String getIsim() {

return isim;

}

public void setIsim(String isim) {

if(isim.length()<3) {

System.out.println("Geçersiz isim");

this.isim="Kullanıcı";

}

else {

this.isim = isim;

}}

public String getSoyisim() {

return soyisim;

}

public void setSoyisim(String soyisim) {

if(soyisim.length()<3) {

System.out.println("Geçersiz soyisim");

this.soyisim="SoyisimKullanıcı";

}

else {

this.soyisim = soyisim;

}

}

public String getOkulno() {

return okulno;

}

public void setOkulno(String okulno) {

this.okulno = okulno;

}

public int getYas() {

return yas;

}

public void setYas(int yas) {

this.yas = yas;

}

public String toString() {

return "ogrenci [okul="+ okul +" isim=" + isim + ", soyisim=" + soyisim + ", okulno=" + okulno + ", yas=" + yas + ", adres="+ adres + ", Kurs" + kurslar + "]";

}

}

**MAİN CLASS**

**public** **class** ogrenciTest {

**public** **static** **void** main(String[] args) {

adres adres = **new** adres("113/1", "Beştelsiz","Zeytinburnu","İst");

kurslar kurs1 = **new** kurslar("Java","MetinHoca",32);

kurslar kurs2 = **new** kurslar("C#","EnginHoca",24);

kurslar kurs3 = **new** kurslar("Javascript","KemalHoca",48);

ArrayList<kurslar> kurslar = **new** ArrayList<kurslar>();

kurslar.add(kurs1);

kurslar.add(kurs2);

kurslar.add(kurs3);

ogrenci ogr2 = **new** ogrenci();

ogr2.setIsim("Sa");

ogr2.setSoyisim("La");

ogr2.setKurslar(kurslar);

ogr2.setOkulno("1231");

ogr2.setYas(22);

ogr2.*setOkul*(ogrenci.*getOkul*());

ogr2.setAdres(adres);

*ogrenciBilgileriYazdir*(ogr2);

}

**public** **static** **void** ogrenciBilgileriYazdir(ogrenci o) {

System.***out***.println("Okul="+o.*getOkul*());

System.***out***.println("İsim="+o.getIsim());

System.***out***.println("Soyisim="+o.getSoyisim());

System.***out***.println("Okul No="+o.getOkulno());

System.***out***.println("Yaş="+o.getYas());

System.***out***.println("Adres="+o.getAdres());

ArrayList<kurslar> kurslar = o.getKurslar();

**for** (kurslar kurs : kurslar) {

System.***out***.println(kurs);

}}}

**ÇIKTI**

---

Geçersiz isim

Geçersiz soyisim

Okul=Akdeniz Üniversitesi

İsim=Kullanıcı

Soyisim=SoyisimKullanıcı

Okul No=1231

Yaş=22

Adres=[sokak=113/1, mahalle=Beştelsiz, ilce=Zeytinburnu, il=İst]

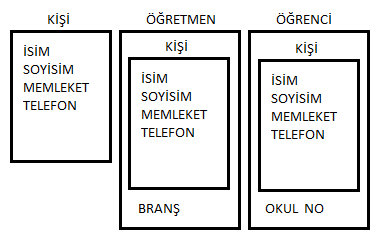
[kursAdi=Java, kursOgr=MetinHoca, kursSaati=32]

[kursAdi=C#, kursOgr=EnginHoca, kursSaati=24]

[kursAdi=Javascript, kursOgr=KemalHoca, kursSaati=48]

---

**///KALITIM (İNHERİTANCE)///**



Yukarıdaki resim mantığı ile çalışır.Kişi adında bir class oluştururuz.İsim,Soyisim gibi değişkenler alırız.Sonrasında Öğretmen ve öğrenci içinde aynı değişkenler almamız gerektiği için bir daha yazıp set,get kullanmamıza gerek kalmaz.Sadece Constructor kullanırız.Bunun için ilk önce normal bir kişi classı oluştururuz.Ardından öğretmen ve öğrenci classını oluşturdumuz zaman yanına extends kisi yazarız.Böylelikle kişi classının içindeki veriler öğretmen ve öğrenci classınada dahil olur.

**KİSİ CLASSI**

private String isim;

private String soyisim;

private String telefonNo;

public kisi() {

}

public kisi(String isim, String soyisim, String telefonNo) {

this.isim = isim;

this.soyisim = soyisim;

this.telefonNo = telefonNo;

}

public String getIsim() {

return isim;

}

public void setIsim(String isim) {

this.isim = isim;

}

public String getSoyisim() {

return soyisim;

}

public void setSoyisim(String soyisim) {

this.soyisim = soyisim;

}

public String getTelefonNo() {

return telefonNo;

}

public void setTelefonNo(String telefonNo) {

this.telefonNo = telefonNo;

}

@Override

public String toString() {

return "kisi [isim=" + isim + ", soyisim=" + soyisim + ", telefonNo=" + telefonNo + "]";}

**ÖĞRETMEN CLASSI**

private String brans;

public ogretmen() {

}

public ogretmen(String brans) {

this.brans = brans;

}

public String getBrans() {

return brans;

}

public void setBrans(String brans) {

this.brans = brans;

}

**ÖĞRENCİ CLASSI**

private String okulNo;

public ogrenci() {

}

public ogrenci(String okulNo) {

this.okulNo = okulNo;

}

public String getOkulNo() {

return okulNo;

}

public void setOkulNo(String okulNo) {

this.okulNo = okulNo;

}

**TEST CLASSI**

**public** **static** **void** **main**(String[] args) {

ogrenci **ogr1** = **new** ogrenci();

ogr1.setIsim("Samet");

ogr1.setSoyisim("Çavur");

ogr1.setTelefonNo("31233211");

ogr1.setOkulNo("999999");

System.***out***.println(ogr1.getOkulNo());

ogretmen **ogt1** = **new** ogretmen();

ogt1.setIsim("Murat");

ogt1.setSoyisim("Hoca");

ogt1.setTelefonNo("685854");

ogt1.setBrans("YBS");

System.***out***.println(ogt1.getBrans());

**ÇIKTI**

---

999999

YBS

---

**///KALITIMDA SUPER KULLANIMI///**

Parametresiz constructor oluştururken super() kullanmamıza gerek yok,fakat çoğunlukla parametreli constructorda kullandığımız için superi çokça kullanmammız gerekli,superin amacı extend ettiğimiz classtaki değişkenleri extend edilen classtada kullanmak için parametreli constructor içine super(isim,soyisim); açıyoruz ve parantezin için extend ettiğimiz classtaki değişkenleri ve verileri yazıyoruz.Aynı zamanda constructorun içinede değişken türüyle adını belirliyoruz.

Diyelim ki extend ettiğimiz classımızda şu değişkenler var:

**private** String isim;

**private** String soyisim;

**private** String telefonNo;

Extend edilen classta da şu değişkenler var;

**private** String brans;

**private** String statu;

Bu classın içindeki constructor şöyle olmalı:

**public** **ogretmen**() {

}

**public** **ogretmen**(String brans,String statu,String isim,String soyisim,String telefonNo) {

**super**(isim, soyisim, telefonNo); (Şuraya şöyle bir super belirledik)

**this**.brans = brans;

**this**.statu = statu;

**///KALITIMDA ToString KULLANIMI///**

Kalıtım konusunda toString normal toStringe göre biraz farklı oluyor.Extend ettiğimiz classtaki değişkenler extend edilen classtaki toString in içine yazılıyor fakat get ile beraber.

Üstteki Örneğin toStringi şu şekilde olucak:

**public** String toString() {

**return** "ogretmen [brans=" + brans + "statu = " + statu + "İsim" + getIsim() + ", Soyisim=" + getSoyisim()+ ", TelefonNo=" + getTelefonNo();

**///POLYMORPHİZM(ÇOK BİÇİMLİLİK)///**

Polymorfizimin mantığı şudur diyelim bir insan tanımladık,bu insanın ismi,soyismi,yaşı,memleketi onun kişisel özellikleridir,bir de doktor öğrenci öğretmen tanımladık bu meslekler onların vasıflarıdır.Bizim İnsan yada kişi diye bir classımız var,ve öğretmen,öğrenci,doktor diye classlarımız var,öğretmen classımızın içinde branşı,öğrenci classımızın içinde okul no,doktor classımızın içinde de alanı var,main classta öğretmen öğrenci yada doktoru çağırırsak bütün özellikleri gelir,fakat insan classından çağırırsak bir tek ismi soyismi yaşı memleketi gelir.Mantık budur.Bir nevi ana classta ki metodları diğer classlarda override etme işlemidir.Öğrenci doktor yada öğretmenin nesnesini oluştururken Ogrenci ogr1 = new Ogrenci(); değilde Insan ogr1 = new Ogrenci(); olarak oluştururuz.Böylece insanın(öğrenci öğretmen doktor) herhangi bi özelliğini almak istediğimizde insan classından çağırırız.

**ÖRNEK**

**İNSAN CLASSI**

**public** **class** insan {

**private** String isim;

**private** String soyisim;

**private** **int** yas;

**public** **insan**() {

}

**public** **insan**(String isim,String soyisim,**int** yas) {

**this**.isim=isim;

**this**.soyisim=soyisim;

**this**.yas=yas;

}

**public** String **getIsim**() {

**return** isim;

}

**public** **void** **setIsim**(String isim) {

**this**.isim = isim;

}

**public** String **getSoyisim**() {

**return** soyisim;

}

**public** **void** **setSoyisim**(String soyisim) {

**this**.soyisim = soyisim;

}

**public** **int** **getYas**() {

**return** yas;

}

**public** **void** **setYas**(**int** yas) {

**this**.yas = yas;

}

@Override

**public** String **toString**() {

**return** "insan [isim=" + isim + ", soyisim=" + soyisim + ", yas=" + yas + "]";

}

**ÖĞRETMEN CLASSI**

**public** **class** ogretmen **extends** insan{

String brans;

**public** **ogretmen**() {

}

**public** **ogretmen**(String isim, String soyisim, **int** yas) {

**super**(isim, soyisim, yas);

// **TODO** Auto-generated constructor stub

}

**public** **ogretmen**(String brans,String isim, String soyisim, **int** yas) {

**super**(isim, soyisim, yas);

**this**.brans=brans;

// **TODO** Auto-generated constructor stub

}

**public** String **getBrans**() {

**return** brans;

}

**public** **void** **setBrans**(String brans) {

**this**.brans = brans;

}

@Override

**public** String **toString**() {

**return** "ogretmen [brans=" + brans + ", İsim=" + getIsim() + ", Soyisim=" + getSoyisim()+ ", Yas=" + getYas() + "]";

}

**ÖĞRENCİ CLASSI**

**public** **class** ogrenci **extends** insan {

String okulNo;

**public** **ogrenci**() {

}

**public** **ogrenci**(String okulNo,String isim,String soyisim,**int** yas) {

**super**(isim,soyisim,yas);

**this**.okulNo = okulNo;

}

**public** **ogrenci**(String isim,String soyisim,**int** yas) {

**super**(isim,soyisim,yas);

}

**public** String **getOkulNo**() {

**return** okulNo;

}

**public** **void** **setOkulNo**(String okulNo) {

**this**.okulNo = okulNo;

}

@Override

**public** String **toString**() {

**return** "ogrenci [okulNo=" + okulNo + ", Isim=" + getIsim() + ", Soyisim=" + getSoyisim()

+ ", Yas=" + getYas() + "]";

}

**TEST CLASSI**

**public** **class** Test {

**public** **static** **void** **main**(String[] args) {

insan **ogt1**=**new** ogretmen("YBS", "Resul", "Kara", 35);

insan **ogr1**= **new** ogrenci("1234", "Samet", "Çavur",23);

*insanBilgileriniGoster*(ogt1);

System.***out***.println();

*insanBilgileriniGoster*(ogr1);

System.***out***.println();

*ogrenciBilgileriniGoster*(ogr1);

System.***out***.println();

*ogretmenBilgileriniGoster*(ogt1);

}

**public** **static** **void** **insanBilgileriniGoster**(insan insan) {

System.***out***.println("İsim=" + insan.getIsim());

System.***out***.println("Soyisim=" + insan.getSoyisim());

System.***out***.println("Yaş=" + insan.getYas());

}

**public** **static** **void** **ogrenciBilgileriniGoster**(ogrenci ogrenci) {

System.***out***.println(ogrenci);

}

**public** **static** **void** **ogretmenBilgileriniGoster**(ogretmen ogretmen) {

System.***out***.println(ogretmen);

}

---

İsim=Resul

Soyisim=Kara

Yaş=35

İsim=Samet

Soyisim=Çavur

Yaş=23

ogrenci [okulNo=1234, Isim=Samet, Soyisim=Çavur, Yas=23]

ogretmen [brans=YBS, İsim=Resul, Soyisim=Kara, Yas=35] ---

**///OVERRİDE VE OVERLOAD///**

**Overload**

Overload bir metodun aşırı yüklenmesi anlamına gelir. Java'da overload ile aynı isimde birden fazla metot yazabiliriz. Bu metotları birbirinden ayıran fark aldıkları parametreler olur. Örneğin programımızda selamVer adında bir metot olsun. Bu metodu selamVer() şeklinde yazarak ekrana sadece selam yazdırabiliriz. Aynı şekilde başka bir metot olarak selamVer(String isim) şeklinde tanımlayabiliriz. Programımızda selamVer metodunu bir string değer ile çağırırsak ikinci metot çağırılır eğer parametresiz çağırırsak birinci metot çağırılır. Java burada metodun aldığı parametrelere bakarak hangi metodun çağırılacağına karar verir. Bu işleme overload denir. Burada dikkat edilmesi gereken konular ise parametrelerin sayısı, tipi ve sırası farklı ise kodumuz sorunsuz çalışacaktır. Ancak aynı parametreler ile aynı isimde iki metot yazarsa burada hata alırız. Bir diğer konu ise Java dönüş değerine göre metotları ayırt edemez. Yani void selamVer() metodu ve String selamVer() metodunu programımıza yazarsa Java bu ikisi arasındaki farkı anlayamaz ve hata verir. Çünkü selamVer() metodunu çağırdığımız yerde hangi dönüş değerini beklediğimizi belirtmek zorunda değilizdir.

**Override**

Override bir metodun tekrardan yazılması anlamına gelir. Bildiğiniz gibi Java'da kalıtım vardır. Aynı şekilde interface yapısıda Java dilinde kullanılır. Yazdığımız bir sınıfa kalıtım yolu ile veya interface yolu ile birden çok metot dahil edebiliriz ve bu metotları kullanabiliriz. Ancak kalıtım ile aldığımız bir metodu değiştirmek istersek o zaman override etmemiz yani yeniden yazmamız gerekir. Interface yapısı ile aldığımız metotları da zorunlu olarak tekrardan yazmamız zaten gereklidir. İşte bu işleme override denir.

public class Ornek extends Ornek2 {

class Ornek2 {

public void hareketEt() {

System.out.println("hareket edildi");

}

}

**// Overload**

public void selamVer() {

System.out.println("Selam");

}

public void selamVer(String isim) {

System.out.println("Selam " + isim);

}

public void selamVer(String isim, String soyisim) {

System.out.println("Selam " + isim + " " + soyisim);

}

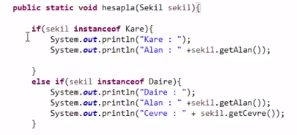
**// Override**

public void hareketEt() {

System.out.println("hareketEt metodu override edildi");

}}

**///İnstanceOf///**



İnstance of şudur;yukarıdaki örneği incelediğimiz de polymorfizm uygulanan bir proje görüyoruz.Fakat bir classın içerisinde bazı koşullarda bazı sonuçlar için instanceOf kullanırız.Bir üst örnektene uyarlicaksak eğer şöyle diyebiliriz.Diyelim insan classı var içinde isim soyisim ve yaş değişkenleri var ve biz öğrenciyi bir insan classı ile yazdırmak istersek öğrencinin ismi soyismi ve yaşını alırız.Okul numarasını almayız çünkü öğrenci classı ile değil insan classı ile yazdırdık.Fakat biz öğretmen yazdırmak istediğimizde sadece isim ve soyismini alacağız,yaşını almak istemiyoruz.O zaman yazdırmak istediğimiz classın içerisinde if(insan instanceof öğretmen) println(isim,soyisim) gibi yazarız.

**/// ABSTRACT(SOYUTLAMA) ///**

Abstractların mantığı şudur;diyelim bir şirkette çalışanlar var bu çalışanlarda işçi,mühendis,yönetici diye 3’e ayrılıyor.Yani şirketteki her çalışan 3 vasıftan birine sahip,biz bu durumda kod yazarken 5 tane class açarız.Bunlar çalışan,işçi,mühendis,yönetici ve test classıdır.Test classına herhangi bir çalışanın nesnesini oluşturmak istersek mühendis,işçi yada yönetici oluşturabiliriz,fakat çalışan diye de bir nesne oluşturabiliriz.Fakat bu yanlış olur.Çünkü çalışan şirketteki bir eleman değil,mühendisin,işçinin yada yöneticinin genel adıdır.Bu sebepten ötürü Çalışanı abstract ederiz ve test classında çalışan adında artık bir nesne olmaz.Ayrıca bu 3 elemanda kullanılması zorunlu bir metod olmasını istersek(mesela çalışma saati başına 10 lira mühendis için,çalışma saati başına 8 lira işçi için,çalışma saati başına 12 lira yönetici için) bunun için çalışan classının içine bir abstract metodta açabiliriz.Böylelikle calismaSaati adındaki metodu mecburen işçi,mühendis ve yönetici classlarında da kullanmak zorunda kalırız.Abstractlar görev olarak polymorfizme benzer önemli bir farkı abstract ettiğimiz classın nesnesini oluşturamayız.

ÖRNEK

**ARABA CLASSI**

**public** **abstract** **class** Araba {

**private** **int** yas;

**private** ArrayList<String> kazaDurumu;

**private** **int** km;

**private** String model;

**private** String renk;

**public** **Araba**() {

}

**public** **Araba**(**int** yas, ArrayList<String> kazaDurumu, **int** km,String model,String renk) {

**this**.yas = yas;

**this**.kazaDurumu = kazaDurumu;

**this**.km = km;

**this**.model=model;

**this**.renk=renk;

}

**public** **abstract** **int** **getKmBasiYakit**();

**public** String **getRenk**() {

**return** renk;

}

**public** **void** **setRenk**(String renk) {

**this**.renk = renk;

}

**public** String **getModel**() {

**return** model;

}

**public** **void** **setModel**(String model) {

**this**.model = model;

}

**public** **int** **getYas**() {

**return** yas;

}

**public** **void** **setYas**(**int** yas) {

**this**.yas = yas;

}

**public** ArrayList<String> **getKazaDurumu**() {

**return** kazaDurumu;

}

**public** **void** **setKazaDurumu**(ArrayList<String> kazaDurumu) {

**this**.kazaDurumu = kazaDurumu;

}

**public** **int** **getKm**() {

**return** km;

}

**public** **void** **setKm**(**int** km) {

**this**.km = km;

}

@Override

**public** String **toString**() {

**return** "Araba [yas=" + yas + ", kazaDurumu=" + kazaDurumu + ", km=" + km + ", model=" + model + ", renk=" + renk

+ "]";

}

**BMW CLASSI**

**public** **class** BMW **extends** Araba {

**private** String JantKalinligi;

**public** **BMW** (){

}

**public** **BMW**(**int** yas, ArrayList<String> kazaDurumu, **int** km, String model, String renk, String jantKalinligi) {

**super**(yas, kazaDurumu, km, model, renk);

JantKalinligi = jantKalinligi;

}

//Abstract//

@Override

**public** **int** **getKmBasiYakit**(){

// **TODO** Auto-generated method stub

**return** getKm()\*3;

}

**public** String **getJantKalinligi**() {

**return** JantKalinligi;

}

**public** **void** **setJantKalinligi**(String jantKalinligi) {

JantKalinligi = jantKalinligi;

}

@Override

**public** String **toString**() {

**return** "BMW [JantKalinligi=" + JantKalinligi + ", Renk=" + getRenk() + ", Model=" + getModel()

+ ", Yas=" + getYas() + ", KazaDurumu=" + getKazaDurumu() + "]";

}

**MERCEDES CLASSI**

**public** **class** Mercedes **extends** Araba {

**private** **int** kapiAdedi;

**public** **Mercedes**() {

}

**public** **Mercedes**(**int** yas, ArrayList<String> kazaDurumu, **int** km, String model, String renk, **int** kapiAdedi) {

**super**(yas, kazaDurumu, km, model, renk);

**this**.kapiAdedi = kapiAdedi;

}

**public** **int** **getKapiAdedi**() {

**return** kapiAdedi;

}

**public** **void** **setKapiAdedi**(**int** kapiAdedi) {

**this**.kapiAdedi = kapiAdedi;

}

@Override

**public** String **toString**() {

**return** "Mercedes [kapiAdedi=" + kapiAdedi + ", Renk=" + getRenk() + ", Model=" + getModel()

+ ", Yas=" + getYas() + ", KazaDurumu=" + getKazaDurumu() + ", Km=" + getKm() + "]";

}

// Abstract //

@Override

**public** **int** **getKmBasiYakit**() {

// **TODO** Auto-generated method stub

**return** getKm()\*2;

**TEST CLASSI**

**public** **class** Test {

**public** **static** **void** **main**(String[] args) {

BMW **bmw1** = **new** BMW(5, **null**, 1000, "3.18", "Beyaz", "50");

ArrayList<String> **kazalar** = **new** ArrayList<String>();

kazalar.add("Tampon Kazası");

kazalar.add("Kaldırıma Çarpmış");

kazalar.add("At'a Çarpmış");

bmw1.setKazaDurumu(kazalar);

Mercedes **mer1** = **new** Mercedes(10, **null**,2000, "Cla180", "Siyah", 4);

ArrayList<String> **kazalar1** = **new** ArrayList<String>();

kazalar1.add("Hafif Kaza");

kazalar1.add("Trene Çarpmış");

mer1.setKazaDurumu(kazalar1);

*bilgileriGoruntule*(bmw1);

System.***out***.println("------------");

*bilgileriGoruntule*(mer1);

}

**public** **static** **void** **bilgileriGoruntule**(Araba araba) {

System.***out***.println("Arabanın Modeli=" + araba.getModel());

System.***out***.println("Arabanın Kmsi=" + araba.getKm());

System.***out***.println("Arabanın Rengi=" + araba.getRenk());

System.***out***.println("Arabanın Yaşı=" + araba.getYas());

System.***out***.println("Arabanın Kaza Durumu" + araba.getKazaDurumu());

System.***out***.println("Arabanın Km Başına Yakıtı="+ araba.getKmBasiYakit());

}

**/// İNTERFACE(ARAYÜZLER) ///**

Abstractlar ile çok benzer fakat aralarındaki fark şu birincisi interfaceler kullanmamızın zorunlu olduğu classları bir arada tutuyor.Diyelim Okul projesi yaptık içinde öğrenci öğretmen çalışan diye classlar var ve biz kisilerilistele,adresleri listele,öğrenci isimlerini listele gibi metotları kesin kullanıcaz.İşte o zaman bir interface açıyoruz.Ve o classları yazıyoruz.İkinci fark abstractlarda değişkenler özellikler falan hepsi extends edilen classa gidiyoruz fakat interfacelerde bir tek metot kullanabiliriz.Üçüncü fark interfacelerin içindeki classların için her zaman boş olur,implement ettiğimiz classlarda doldururuz.Dördüncü ve en büyük fark bir classı sadece bir abstract ile extend edebiliriz.Fakar interfacelerde bi classta birden fazla interfaceyi implement edebiliriz.Ayrıca interface içindeki classlar parametreli olabilir.Kullanımı ise şöyle;abstractlar ile aynı fakat bunda abstract yerine interface yazıyoruz ve abstractlarda extends olayı vardı bunda da extends yerine implements yazıyoruz.Bir class bir aynı anda bir abstractı extends ve birden çok interfaceyi implement edebilir.Ayrıca interface bir classın başına her zaman I koyarız.Mesela İslemler diye bir class oluşturucaz IIslemler yapmalıyız.

public interface KisiIslemleri

////

public class Kisiİslemleriİmp implements KisiIslemleril

**/// TRY-CATCH ///**

Bir hata yakalama ve hatayı görmezden gelme olayıdır.Try içine burada hata var mı kontrol et dediğimiz kodları yazıyoruz.Catch içine ise hata neymiş yada hata çıkınca ekranda ne yazsın diye kullanıyoruz.Aşağıdaki kodda 5 elemanlı bir dizi yaptık ama 6 defa yazdırmaya çalıştık.

**int** [] **dizi** = **new** **int**[5];

**for** (**int** **i** = 0; i < dizi.length; i++) {

dizi[i]=1;

}

**try**

{

**for** (**int** **i** = 0; i < (dizi.length+1); i++)

{

System.***out***.println(dizi[i]);

}

}

**catch** (Exception **e**)

{

System.***out***.println("Hatalı Kod");

}

----

1

1

1

1

1

Hatalı Kod

---

**///DOSYA OKUMA VE DOSYA YAZMA///**

**package** dosyaYazma;

**import** java.io.\*; //Bunu kesinlikle dahil ediyoruz.

**import** java.util.Scanner;

**public** **class** dosya {

**public** **static** **void** **main**(String[] args) {

*dosyaYazma*();

*dosyaOkuma*();

}

**public** **static** **void** **dosyaYazma**() {

PrintWriter **ourFile** = **null**;

**try**

{

ourFile = **new** PrintWriter("C:/Users/Hp/Desktop/Yesi.txt");

ourFile.println("Samet Çavur");

ourFile.println("Metin kaya");

ourFile.close();

}

**catch** (Exception **e**)

{

System.***out***.println(e.getMessage());

}

}

**public** **static** **void** **dosyaOkuma**() {

File **dosya** = **new** File("C:/Users/Hp/Desktop/Yesi.txt");

**try** {

Scanner **scan** = **new** Scanner(dosya);

String **ekranaYaz**;

**while**(scan.hasNext()) { //hasNextLine yazarsak sonraki satırı alır

ekranaYaz = scan.next(); //Ayrıca hasNext demek sonraki kelime var mı demek gibi

System.***out***.println(ekranaYaz);

}

} **catch** (Exception **e**) {

System.***out***.println("Hata = " + e);

}}}

**///FİLE İŞLEMLERİ///**

**public** **static** **void** **main**(String[] args) {

File **dosya** = **new** File("C:/Users/Hp/Desktop/Dersler");

System.***out***.println(dosya.canExecute()); //Dosya Çalışıyor mu

System.***out***.println(dosya.canRead()); //Dosya okunabilir mi

System.***out***.println(dosya.canWrite()); //Dosya Yazılabilir mi

System.***out***.println(dosya.getAbsolutePath()); //Dosyanın tam yeri

System.***out***.println(dosya.getName()); //Dosyanın Adı

System.***out***.println(dosya.getParent()); //Dosya nereye bağlı

System.***out***.println(dosya.lastModified()); //Dosya en son ne zaman düzenlendi

Date **date** = **new** Date(dosya.lastModified());

System.***out***.println(date); //Dosya en sonra ne zaman düzenlendi.(Tam Tarih)

}

**/// Stack-Heap ///**

Bu ikisi aslında bellekte yer tutan alanlardır.Stack programın içerisindeki bir defa özelleğini tanımladığımız ve bir daha değiştirmediğimiz her kod parçasıdır.Heap ise sonradan değiştirilebilir verilerdir. Stack daha hızlı kullanımı kolay depolama alanıdır.Heap ise daha zahmetlidir.Ayrıca stackte oluşturulan verilen program sonlandırılınca otomatik silinir.Heapte silmeyi manuel olarak yapmalıyız.

**Annotation**=Kod parçasını bir yerden aldıysak,onu belirtmeye denir.Çoğunlukla frameworklerde görüyoruz.Mesela @Override demek annotation demektir.

**Utility Classlar=**İşlemlerin olduğu classlardır.Mesela Topla-Çıkar-Ekle-Sil gibi metodların olduğu classlar.Bu metodlara static yaparız ve diğer classlarda new yazıp nesnesini oluşturmadan da ulaşabiliriz.Static,newlemeden yani nesnesini oluşturmadan o metoda ulaşmaya yarar.Mesela Kisi classı oluşturduk mainde bir kisi oluştururken new kisi1 = new kisi(); yazarız çünkü kisi utility class değildir.Ama topla-çıkar metodlarının olduğu utility bir classı mainde çağırırken islemler islem1 = new islemler(); yazmamıza gerek duymamak için metoda static yazarız ve mainde islemler.topla yazarak ulaşabilir.

## DAL(Data Access Layer) Classlar -“Veri erişim katmanı”=Yazacağımız classın sonuna Dal yazarsak bu classın veri tabanı ile alakalı olduğunu belirtir.sqlDal,postgreDal,msSqlDal gibi.

## Crud Classlar(Create,Read,Update,Delete)=Bu classların içinde yandaki 4 işlemi yaptığımızı belirtiriz.

**///GENERİC CLASSLAR///**

Bu classların mantığı şudur.Diyelim ki Abstract bir kişi classı oluşturduk.Buraya isim,soyisim koyduk.Öğrenci ve Öğretmen diye de başka bir class oluşturduk.Burayada öğrenci no ve branş ekstra koyup kişi classını extend ettik.En son bu kişilerin verilerine ekleme-çıkarma-listeleme gibi crud işlemler yaparken bir tanede Kişiİşlemleri yada kişiManager diye bir class daha oluştururuz ve bu classın adına personManager<HerhangiBisi extends kisi> gibi bir şey yazarız. Bundan sonra o kişiİşlemlerine yada kişiManagera hangi türde veri yollarsak o türde işlem yapar.Örnekte daha iyi anlarsın.Bu sayede öğretmen işlemleri öğrenci işlemleri gibi classlara ihtiyaç kalmıyor,sadece kişi işlemleri yapmış oluyoruz.

(extends kisi yazmamızın sebebi bir tek kişiyi extend etmiş classlar kullanabilsin)

**PERSON CLASS**

**public** **abstract** **class** person {

**private** String name;

**private** String surname;

**private** **int** age;

**public** **person**() {}

**public** **person**(String name, String surname, **int** age) {

**this**.name = name;

**this**.surname = surname;

**this**.age = age;

}

**public** String **getName**() {

**return** name;

}

**public** **void** **setName**(String name) {

**this**.name = name;

}

**public** String **getSurname**() {

**return** surname;

}

**public** **void** **setSurname**(String surname) {

**this**.surname = surname;

}

**public** **int** **getAge**() {

**return** age;

}

**public** **void** **setAge**(**int** age) {

**this**.age = age;

}

@Override

**public** String **toString**() {

**return** "person [name=" + name + ", surname=" + surname + ", age=" + age + "]";

**Student Class**

**public** **class** student **extends** person {

**private** String scNumber;

**public** **student**() {}

**public** **student**(String name, String surname, **int** age, String scNumber) {

**super**(name, surname, age);

**this**.scNumber = scNumber;

}

**public** String **getScNumber**() {

**return** scNumber;

}

**public** **void** **setScNumber**(String scNumber) {

**this**.scNumber = scNumber;

}

@Override

**public** String toString() {

**return** "student [scNumber=" + scNumber + ", Name=" + getName() + ", Surname=" + getSurname()

+ ", Age=" + getAge() + "]";

}

**Teacher Class**

**public** **class** teacher **extends** person{

**private** String branch;

**public** **teacher**() {}

**public** **teacher**(String name, String surname, **int** age, String branch) {

**super**(name, surname, age);

**this**.branch = branch;

}

**public** String **getBranch**() {

**return** branch;

}

**public** **void** **setBranch**(String branch) {

**this**.branch = branch;

}

@Override

**public** String **toString**() {

**return** "teacher [branch=" + branch + ", Name=" + getName() + ", Surname=" + getSurname()

+ ", Age=" + getAge() + "]";

}

**PersonManager Class**

public class personManager<belliOlmayanDegisken extends person> {

private ArrayList<belliOlmayanDegisken> p = new ArrayList<>();

public void add(belliOlmayanDegisken person) {

p.add(person);

}

public void remove(belliOlmayanDegisken person) {

p.remove(person);

}

public void show() {

for (belliOlmayanDegisken person : p) {

System.*out*.println(person);

}

}

**Main Class**

**public** **static** **void** **main**(String[] args) {

personManager<teacher> **teachers** = **new** personManager<>();

personManager<student> **students** = **new** personManager<>();

student **s1** = **new** student("Samet", "Çavur", 23, "12345");

student **s2** = **new** student("Hakkı", "Sini", 32,"62614");

student **s3** = **new** student("Resul","Şaşmaz",28,"13221");

students.add(s1);

students.add(s2);

students.add(s3);

teacher **t1** = **new** teacher("Murat", "Yıldıray", 43, "Matematik");

teacher **t2** = **new** teacher("Engin", "Yıldırmaz", 27, "Fizik");

teacher **t3** = **new** teacher("Tayfun", "Yüksek", 33, "Yazılım");

teachers.add(t1);

teachers.add(t2);

teachers.add(t3);

System.***out***.println("---Students---");

students.show();

System.***out***.println("---Teachers---");

teachers.show();

**///ESNEK BAĞLANTI///**

Bu bir konu değil bir işlem gibi bir şey.Amacı ise diyelim bir interface oluşturduk ve o interfaceyi implement eden bir xyz classı var bu projeyi normalce kullanıyoruz ama bi zaman sonra xyz classını abc olarak değiştirmek,güncellemek zorunda kaldık.İşte o zaman main classta xyz ye ait herşeyi değiştirmek yerine xyz yi oluşturduğumuz nesneyi abc olarak değiştiriyoruz.Güncelleme ve değiştirme işleminde çok fazla zaman ve kolaylık kazanıyoruz.ÖRNEĞİN;

**İNTERFACE CLASSI**

**public** **interface** I\_arabaIslemleri {

**public** ArrayList<araba> ***arabalar*** = **new** ArrayList<>();

**public** **boolean** **arabaKaydet**(araba araba);

**public** **boolean** **arabaSil**(araba araba);

**public** **void** **arabaBilgiGoster**(araba araba);

**public** **void** **listele**();

**INTERFACEYİ İMPLEMENT EDEN CLASS**

**public** **class** arabaIslemleri **implements** I\_arabaIslemleri {

**public** **boolean** **arabaKaydet**(araba araba) {

**boolean** **durum** = ***arabalar***.add(araba);

**return** durum;

}

**public** **boolean** **arabaSil**(araba araba) {

**boolean** **durum** = ***arabalar***.remove(araba);

**return** durum;

}

**public** **void** **arabaBilgiGoster**(araba araba) {

System.***out***.println(araba.getMarka());

System.***out***.println(araba.getModel());

System.***out***.println(araba.getYil());

}

**public** **void** **listele**() {

**for** (araba **araba** : ***arabalar***) {

System.***out***.println(araba);}}

**NESNE CLASSI**

**public** **class** araba {

**private** String marka;

**private** String model;

**private** **int** yil;

**public** **araba**() {}

**public** **araba**(String marka,String model,**int** yil) {

**this**.marka=marka;

**this**.model=model;

**this**.yil=yil;

}

**public** String **getMarka**() {

**return** marka;

}

**public** **void** **setMarka**(String marka) {

**this**.marka = marka;

}

**public** String **getModel**() {

**return** model;

}

**public** **void** **setModel**(String model) {

**this**.model = model;

}

**public** **int** **getYil**() {

**return** yil;

}

**public** **void** **setYil**(**int** yil) {

**this**.yil = yil;

}

**public** String **toString**() {

**return** "araba [marka=" + marka + ", model=" + model + ", yil=" + yil + "]";

}

}

**MAİN CLASSI**

**public** **static** **void** **main**(String[] args)

{ araba **araba1** = **new** araba("bmw", "318", 2015);

araba **araba2** = **new** araba("mercedes", "cla180", 2017);

arabaIslemleri **arabaIslemleri** = **new** arabaIslemleri();

///Burada nesneyi oluştururken referans alınan classı interface yapıyoruz.

arabaIslemleri.arabaKaydet(araba1);

arabaIslemleri.arabaKaydet(araba2);

arabaIslemleri.listele();

}

**İNTERFACEYİ İMPLEMENT EDEN CLASSI GÜNCELLİYORUZ**

**public** **class** arabaIslemleriGuncel **implements** I\_arabaIslemleri{

**public** **boolean** **arabaKaydet**(araba araba) {

System.***out***.println("---KAYDEDİLEN ARABA---");

arabaBilgiGoster(araba);

**boolean** **durum** = ***arabalar***.add(araba);

**return** durum;

}

**public** **boolean** **arabaSil**(araba araba) {

System.***out***.println("---SİLİNEN ARABA---");

arabaBilgiGoster(araba);

**boolean** **durum** = ***arabalar***.remove(araba);

**return** durum;

}

**public** **void** **arabaBilgiGoster**(araba araba) {

System.***out***.println(araba.getMarka());

System.***out***.println(araba.getModel());

System.***out***.println(araba.getYil());

}

**public** **void** **listele**() {

**for** (araba **araba** : ***arabalar***) {

System.***out***.println(araba); }}

**YENİ MAİN CLASSI**

**public** **class** main {

**public** **static** **void** **main**(String[] args) {

araba **araba1** = **new** araba("bmw", "318", 2015);

araba **araba2** = **new** araba("mercedes", "cla180", 2017);

I\_arabaIslemleri **arabaIslemleri** = **new** arabaIslemleriGuncel();

///Burada nesneyi arabaIslemleriGuncel **arabaIslemleri** = **new** arabaIslemleriGuncel(); olarak tanımlamak yerine I\_arabaIslemleri olarak tanımladık ve artık arabaIslemleriGuncel classı değiştikçe yada güncellendikçe o güncellenen classın adını buraya yazarız.

arabaIslemleri.arabaKaydet(araba1);

arabaIslemleri.arabaKaydet(araba2);

arabaIslemleri.listele();

}}

**///COLLECTİON///**

**(SET)**

**import** java.util.\*; (BUNU HER ZAMAN DAHİL EDİYORUZ.)

**public** **class** **mainSet** {

**public** **static** **void** main(**String**[] args) {

/\*SET KULLANIMI

Setin asıl özelliği aynı veriyi tekrar tutmaz.

1.hashset

2.treeset

3.linkedhashset\*/

//----HASH SET----//

//Özelliği sıralama yoktur.

//Hash kodlarına göre sıralar//

//Null bir tane tutar en başa yazar//

**Set** isimler = **new** HashSet();

isimler.add("Samet");

isimler.add("Murat");

isimler.add("Halil");

//İterator ile yazdırma//

**Iterator** isimlerItr = isimler.iterator();

**while**(isimlerItr.hasNext()) {

**System**.*out*.println(isimlerItr.next());

}

//For Each ile yazdırma//

isimler.forEach(item -> **System**.*out*.println(item + " "));

**System**.*out*.println("-------");

//----LİNKED HASHSET----//

//Özelliği sıralama vardır.İlk giren ilk çıkar//

//null bir tane tutar//

**Set** gunler = **new** LinkedHashSet();

gunler.add("Ptesi");

gunler.add("Salı");

gunler.add("Çarş");

**Iterator** gunlerItr = gunler.iterator();

**while**(gunlerItr.hasNext()) {

**System**.*out*.println(gunlerItr.next());

}

**System**.*out*.println("--------");

//----TREE SET----//

//Artan sıralama ile tutar//

//null kabul etmez//

**Set** sayilar = **new** TreeSet();

sayilar.add(4);

sayilar.add(2);

sayilar.add(8);

sayilar.add(6);

sayilar.forEach(sayi -> **System**.*out*.println(sayi+" "));

}

}

**(LİST)**

/\*LİST KULLANIMI

\* Tekrarlı verilere izin verir.

\* İlk giren ilk çıkar.

\* Sıralama var

\* İstenilen indekse veri eklenebilir.

\*/

//----ARRAYLİST LİNKEDLİST İLE AYNI ŞEY----//

//Yukarıdaki özellikler burada geçerli//

**ArrayList** array = **new** ArrayList();

array.add("Samet");

array.add("Samet");

array.add(14);

array.add(**null**);

array.add(**null**);

array.add(1,"Murat" ); 1.indekse Murat’ı koy demek

**Iterator** itr = array.iterator();

**while**(itr.hasNext()) {

**System**.*out*.println(itr.next());

}

**(MAP)**

/\*MAP KULLANIMI

\* Herbir değeri key value ilişkisiyle tutar.

\* Arraylerdeki add yerine put koyarız.

\* Verileri çekmek içinde get koyarız.\*/

**Map** ıdPass = **new** HashMap();

ıdPass.put("ıd","Samet");

ıdPass.put("password","1234");

**System**.*out*.println("Kullanıcı Adı:"+ıdPass.get("ıd"));

**System**.*out*.println("Şifre:"+ıdPass.get("password"));

ÇIKTI

Kullanıcı Adı:Samet

Şifre:1234

//Nesneler ile map kurulabilir.//

**Map**<String,mapOgrenci> ogrenci = **new** HashMap<>();

**mapOgrenci** ogr1 = **new** mapOgrenci("Samet", "Çavur", 1997);

**mapOgrenci** ogr2 = **new** mapOgrenci("Hakkı", "Duymaz", 1995);

**mapOgrenci** ogr3 = **new** mapOgrenci("Resul","Çalışkan", 1993);

ogrenci.put("ilk", ogr1);

ogrenci.put("ikinci", ogr2);

ogrenci.put("ucuncu", ogr3);

**System**.*out*.println(ogrenci.get("ilk"));

ÇIKTI=Kisi [isim=Samet, soyisim=Çavur, dogumYili=1997]

//ogrenci mapindeki tüm keyleri alma//

**System**.*out*.println(ogrenci.keySet());//ÇIKTI = [ilk, ucuncu, ikinci]

//Map içindeki verileri koleksiyon yapabiliriz.//

**Collection** <mapOgrenci> ogrenciListesi = ogrenci.values();

**System**.*out*.println(ogrenciListesi);

ÇIKTI = [Kisi [isim=Samet, soyisim=Çavur, dogumYili=1997], Kisi [isim=Resul, soyisim=Çalışkan, dogumYili=1993], Kisi [isim=Hakkı, soyisim=Duymaz, dogumYili=1995]]

}}

**///Date///**

**Date** date = **new** Date();

**System**.*out*.println(date);

//Sadece gün ay yıl olarak yazdırma//

**SimpleDateFormat** dateFormat = **new** SimpleDateFormat("dd.MM.yyyy");

**System**.*out*.println(dateFormat.format(date));

**///Calender///**

**Calendar** cal = **Calendar**.*getInstance*();

cal.set(2021,1,27);

**System**.*out*.println(cal.getTime());

//Simple Formatla Yazma//

**SimpleDateFormat** simpCal = **new** SimpleDateFormat("dd.MM.yyyy");

**System**.*out*.println(simpCal.format(cal.getTime()));

//gün ay yılı ayrı ayrı ekleme//

**Calendar** cal1 = **Calendar**.*getInstance*();

cal1.set(**Calendar**.*YEAR*, 2016);

cal1.set(**Calendar**.*MONTH*,6);

cal1.set(**Calendar**.*DAY\_OF\_YEAR*, 26);

**System**.*out*.println(cal1.getTime());

**///Threading-MultiThreading///**

Bunun amacı aynı anda 2 işlem yapmak.Örneğin bi for döngüsü oluşturduk ve 10 na kadar sayacak.Bir for döngüsü daha oluşturduk oda 10 dan geri sayacak.Ama normalde ilk 10 na kadar gidip sonra ikinci for çalışır 10 dan geri sayar.Bu multi threading işlemi sayesinde ilk for 10 na kadar sayarken ikinci forda aynı anda 10 dan geri sayıyor.

KRONOMETRE CLASSI

**public** **class** **kronometre** **implements** **Runnable** { Şunu kesin eklemeliyiz

**private** **String** threadName;

**public** kronometre(**String** threadName) {

**this**.threadName = threadName;

}

**public** **void** run() {

**try** {

**for** (**int** i = 1; i <= 10; i++) {

**System**.*out*.println(threadName+ "=" +i);

**Thread**.*sleep*(1000);

}

}

**catch** (**InterruptedException** e)

{e.getMessage();}

}}

MAİN CLASSI

**kronometre** kro1 = **new** kronometre("İlk Kronometre");

**Thread** t1 = **new** Thread(kro1);

**kronometre** kro2= **new** kronometre("İkinci Kronometre");

**Thread** t2 = **new** Thread(kro2);

t1.start();

t2.start();