Class (Sınıf) ve Nesne (Object) Kavramları

Class (Sınıf) ve Nesne (Object) Kavramları

Burada iki farklı tanım yapılabilir.

Sınıf dediğimiz kavram aslında bizim kendi veri türümüz, nesne ile bu veriye değer atadığımız varlıklardır.
int sayi = 5 ; int veri türüne 5 değerini atadık. Sınıfları kullanarak kendi veri türümüzü (int gibi) oluşturup, nesneler oluşturup değerler atayabilirz. (5 gibi);

Sınıflar ile veri yapısının taslağını oluştururuz(mesela bir binanın projesi), soyut yapılardır. Sınıfları kullanarak ortaya bir ürün çıkardığımızda nesne üretmiş oluruz(bina projesini kullanarak ortaya bir yapı çıkarmak), somut yapılardır.

Aynı projeyi kullanarak birden fazla bina inşa edilebilir. Yani bir sınıfı kullanarak birden fazla nesne üretilebilir.

Eğer daha önce anlatılan metot, dizi ve bellek durumlarını anladıysanız class ve nesne kavramlarını anlamanız çok zor olmayacaktır.
Sınıflar referans tipli veri türleridir (dizi gibi) ve nesneler heap alanında tutulur.

Bir nesne bir şeyler bilir ki bunları değişkenler ile gösteririz, bir şeyler yapar ki bunları da metotlarda belirtiriz. Sınıflar nesnelerin taslakları olduğu için bu değişken ve metotları tanımladığımız yerler sınıflar olmalıdır.

Bir sınıftan bir nesne üretildiğinde bu nesnenin heap alanında saklandığını söylemiştik. Peki bu kendi oluşturduğumuz veri için heap alanında ne kadar yer ayrılacağını ve bu alanın ne zaman ayrılacağını nasıl belirtiriz ?

new anahtar sözcüğü ile heap alanında belli bir alan nesnelerimiz için ayrılır. Peki ne kadar alan ayrılacak ?

Bu sorunun yanıtı için constructor yani kurucu metot konusuna bakalım.

Constructor Kurucu Metot Kavramı

Constructor Kurucu Metot Kavramı

Şu ana kadar bilmediğimiz bir konudan bahsedilmedi. Yeni öğreneceğimiz constructor da aslında daha önceden öğrendiğimiz metot kavramından çok da farklı değil. Sadece biraz daha özel bir metot. Bu metotu özel yapan nedir ?

- Yeni bir nesne oluşturulduğunda otomatik olarak çağrılır.
- Heap alanında ne kadar yer ayrılacağı bu metot sayesinde anlaşılır.
- Geriye bir değer dönmez ve void ile belirtilmez.
- Sınıf adı ile birebir aynı olmalıdır.
- Bu metot sınıf içindeki değişkenlere ilk değerler atanırken de kullanılır.

Java da her sınıf için varsayılan olarak ve herhangi bir parametre almayan bir constructor bulunur ve buna default constructor denir. Bunu biz yazmasak bile sistem otomatik olarak bunu oluşturur ve new ile nesne ürettiğimiz anda bunu çağırır.

```
//Dikdortgen içinde 2 tane int değer tutmak istediğim ve benim oluşturduğum veri türünün
adıdır. Bundan sonra bu veri türüyle oluşturacağım değerlerin bir taslağıdır.
public class Dikdortgen {
    //nesnenin bildiği şeyler, field variable
    int en;
    int boy;

    //bunu yazmasak da sistem otomatik olarak bu kurucu metotu new ile nesne
    //olusturdugumuz anda çalıştırır.
    public Dikdortgen(){
    }

    //nesnenin yapabildiği şeyler,
    public int alanHesapla(){
        return en * boy ;
    }
}

//main metot içerisinde bu taslaktan somut değerler oluşturmak için
Dikdortgen d1 = new Dikdortgen();
Bu örnekte;
Dikdortgen --> veri türü
d1 --> referans adı
new --> bana heap alanından yer ayırmanı istiyorum.
new Dikdortgen() --> bana heap alanından iki tane int değer tutacak yer ayırmanı istiyorum
anlamına gelir.
```

Bir sonraki sayfada bu işlemleri anlatan şekli bulabilirsiniz.

Sınıf nesne ilişkisi

Emre Altunbilek Java Dersleri

Sınıf ve Nesne İlişkisi

```java
public class Dikdortgen {
    //nesnenin bildiği şeyler, field variable
    int en;
    int boy;

    //bunu yazmasak da sistem otomatik olarak bu kurucu metotu new ile nesne
    //olusturdugumuz anda çalıştırır.
    public Dikdortgen(){
    }

    //nesnenin yapabildiği şeyler,
    public int alanHesapla(){
        return en * boy ;
    }
}
```

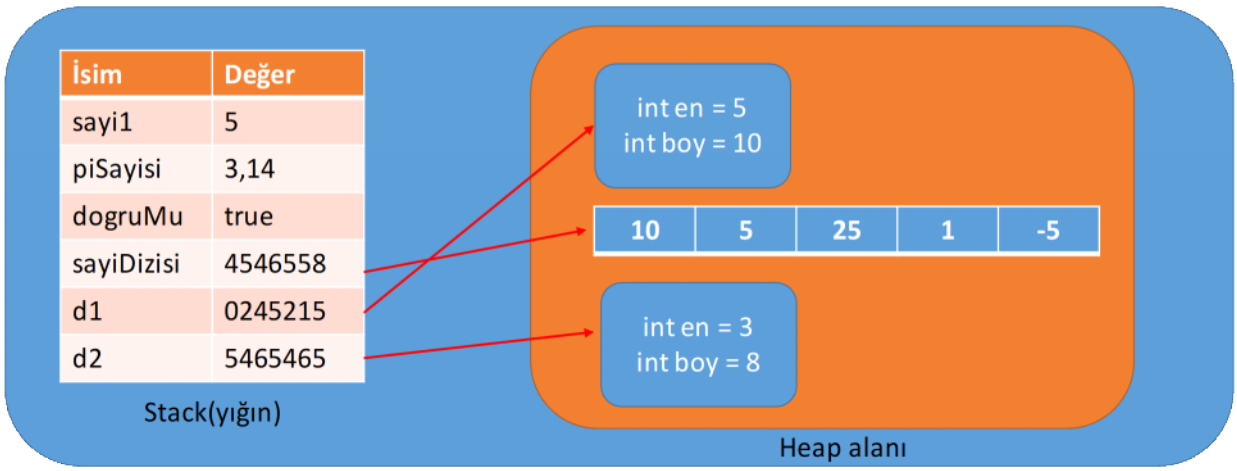
```java
public class OopGirisKavramlar {

    public static void main(String[] args) {

        Dikdortgen d1 = new Dikdortgen();
        d1.en = 5;
        d1.boy = 10;

        Dikdortgen d2 = new Dikdortgen();
        d2.en = 3;
        d2.boy = 8;

    }
}
```



Kendi Kurucu Metotlarımızı Yazmak

Bazı durumlarda bir nesne üretilirken sınıf değişkenlerimize varsayılan olarak bazı değerler atamak isteyebiliriz.Bu durumlarda kurucu metotlar işimize yarar.

Ozaman şunu söylemek yanlış olmaz kurucu metotlar bellekten yer ayırmak ve bellekte ayrılan yerde bulunan değişkenlere ilk değerlerini atamak için kullanılabilir.

Ayrıca farklı kurucu metotlarda bir nesne üretilirken farklı değerlerin atanması mümkündür. Bir sınıfın birden fazla kurucu metotu olabilir. Sistem hangisinin kullanılacağına bu metotun aldığı parametrelerin sayısına ve türüne bakarak karar verir. Method Overloading sayesinde bu işlem gerçekleşir.

Not: Herhangi bir constructor tanımladığınızda artık Java sizin için default constructorı otomatik olarak oluşturmayacaktır.

```
//Ogrenci emre=new Ogrenci() diyebilmek için
//varsayilan constructorı artık elle belirtmeliyiz.
public Ogrenci(){
        System.out.println("Ogrenci nesnesi oluşturuluyor");
        aktif = true;
        ogrenciNo = 9999;
        isim = "Henüz isim verilmemiş";
        sinif = 127;
    }

    public Ogrenci(int ogrenciNo, String isim, byte sinif, boolean aktif){
        this.ogrenciNo = ogrenciNo;
        this.isim = isim;
        this.sinif = sinif;
        this.aktif = aktif;
    }
```

```
Ogrenci emre=new Ogrenci("emre");
Ogrenci hasan = new Ogrenci("hasan");

emre=hasan;

sout(emre.isim)  Sonuç ne olur ?
```

Çözümü videoda bulabilirsiniz.

Bir kurucu metottan baska bir kurucu metotu this ile çağırabilirsiniz. Böyle bir kullanım varsa this ifadesi ilk satırda yazılmalı. Ve bir kurucu metotta birden fazla this ile başka yapılandırıcılar çağrılamaz.

```
Kısacası this.degisken --> o anki nesnenin değişkenlerine erişir
        this() --> o anki nesnenin kurucu metotlarına erişir.
```

Encapsulation

Sınıf içindeki değişkenleri dışarıdan direk olarak erişime kapatıp, bu değerlere metotlar üzerinden erişme kavramına denir.

Değişkenleri private yaparak sadece o sınıf içinde tanımlı olmasını sağlarız. Sonra bu değişkenlerden veri okurken getter dediğimiz, bu değişkenlere veri atarken de setter dediğimiz metotlardan faydalanırız.

Böylece değişkenlerimiz üzerinde yanlış veri olmasının önüne geçer ve bu değerlerin okunması yazılması esnasında daha fazla kontrol gücümüz olur.

Ayrıca encapsulation kullanarak bu kontrolleri sadece sınıfımızda yazdığımız için kod kalabalığından ve gereksiz kod karmaşasından da kurtulmuş oluruz.

```
public class Televizyon {

    private int kanal;
    private boolean acik;

    public void ac(){

        if(acik == false){
            System.out.println("Televizyon açılıyor");
            acik = true;
        }else{
            System.out.println("Televizyon zaten açık");
        }

    }

    public void kapat(){

        if(acik){
            System.out.println("Televizyon kapatılıyor");
            acik = false;
        }else{
            System.out.println("Televizyon zaten kapalı");
        }

    }

    public void setKanal(int yeniKanal){
        if(acik && yeniKanal > 0 && yeniKanal < 500){
            kanal = yeniKanal;
        }else{
            System.out.println("tv kapalı veya yanlış bir kanal değeri yazdınız");
            kanal = 1;
        }
    }
    public int getKanal() {
        return kanal;
    }
```

Static Değişkenler ve Metotlar

static Nedir ?

Bir dikdortgen sınıfı oluşturalım. Bu bölümdeki konular bu örnek üzerinden anlatılacaktır.

```
public class Dikdortgen {
    int en;
    int boy;
    static String ozellik="En ve boy degerleri birbirinden farklı olan geometrik şekildir.";

    public int alanHesapla(){

        return en * boy ;
    }

    public static void ozellikSoyle(){
      System.out.println("Ozellik:" + ozellik);
      }
}

Dikdortgen d1=new Dikdortgen();
d1.en = 10;
Dikdortgen d2 = new Dikdortgen();
d2.en=15;
```

Burada d1 nesnesinin en değeri ile d2 nesnesinin en değeri bellekte farklı yerlerde tutulduğu için birbirinden bağımsızdır. d1'de yapılan değişiklikler d2yi etkilemez. Peki ya tüm sınıf nesnelerinin aynı veriyi paylaşmasını istiyorsak ne yapmalıyız ?

static ile tanımlanan değişkenler o sınıftan türetilen tüm nesnelerce kullanılabilir. Bu yüzden bunlara sınıf değişkeni, en ve boy değişkenlerine ise nesne değişkeni denir.

static ile tanımlanan değişkenlere her nesne erişip kullanabilir. Ortak bir alanda olduğu için bir nesnenin yaptığı değişikliği diğer nesne görebilir.

d1.ozellik = "Boy ve eni eşit olmayan geometrik şekildir"; yapılırsa artık d2 de bu değeri "Boy ve eni eşit olmayan geometrik şekildir" olarak görecektir.
Ayrıca static değişkenlere erişmek için nesne üretmek gerekmez.
Sınıf adı ile de erişilebilir. Dikdortgen.ozellik = "Boy ve eni eşit olmayan geometrik şekildir"; ifadesi doğrudur.

static olarak metot tanımı da yapılabilir. Yine bunlar da tüm nesneler ile erişilebilir ve bunlara erişmek için sınıf adı da kullanılabilir.
d1.ozellikSoyle(); d2.ozellikSoyle(); Dikdortgen.ozellikSoyle(); hepsi doğrudur.

Static olmayan (instance method) bir metot ile;
static olan ve olmayan tüm değişken ve metotlara erişebilirsiniz.

Static olan (class method) metot ile ;
static olan değişken ve metotlara erişebilirken, static olmayan yani instance yani nesneye özgü olan değişken ve metotlara erişemezsiniz.

Immutable Nesneler ve Metotlara Nesne Gönderme

Immutable Class ve Objects

Genellikle bir nesne oluşturur ve içeriğini değiştiririz. Ama bazı durumlarda bir nesneyi oluşturup, değerlerini atayıp sonrasında o değerlerin değiştirilmemesini isteyebiliriz.

Böyle nesnelere immutable object ve bu nesnelerin sınıflarına da immutable class denir.

Bir sınıfı immutable yapmak için ;

Tüm değişkenlerin private olması gerekir.
Setter metotlar olmamalı.
Getter metotlar da herhangi bir referans döndürmemeli.(dizi olarak tutulan bir değişken aslında dizinin kendisidir.)

Metotlara Nesne Gönderme

Nesneler referans tipli veri türü oldukları için metotlara gönderirken değerleri değil bellekte tuttukları adresler gönderilir.
Bundan dolayı metota gönderilen nesne üzerinde yapılan değişiklikler asıl nesnenin içeriğini de değiştirir.

Bölüm Sonu Soruları

Soru 1

CemberDaire isimli sınıf oluşturun. Bu sınıfın yarıçap alan kurucusu olmalı.Ayrıca cevre ve alanını hesaplayan metotlar olmalı.

Soru 2

Ogrenci isimli sınıf oluşturun. Bu sınıfta ogrencinin idsi ve not değeri tutulmalı. 100 elemanlı bir dizide id ve not değerlerini rastgele olusturarak bu ogrencileri saklayın ve bu ogrencileri aldıkları notlara göre azalan sırada yazdıran metotu yazın.

Soru 3

Hesap makinesi sınıfı oluşturun.Bu sınıfta 4 işlem yapabilmek için metotlarınız olsun. Bu metotlara istenilen sayıda parametre geçilebilmeli. Bölme işlemi için 0 değerini için gerekli kontrolü yazın.

Soru 4

Bir banka hesabı için sınıf tasarlayın. Bu sınıfta hesabın kime ait oldugunu tutan hesapNo, hesaptakiPara, para yatirma ve çekme metotları olsun.
Ayrıca bu hesaplarda olan tüm parayı, açılan hesap sayısını, yapılan toplam paracekme ve para yatırma sayısını gösteren bir metot ve iki hesabı para değişkenine göre birbiriyle kıyaslayacak metot bulunmalı.

Çözüm 1


```
public class Soru1 {
    public static void main(String[] args) {

        CemberDaire cember1=new CemberDaire(5);
        System.out.printf("Yarıcapı 5 olan çemberin çevresi : %.2f ",
cember1.cevreBul());
        System.out.println();
        System.out.printf("Yarıcapı 5 olan dairenin alanı : %.2f ",
cember1.alanBul());

    }
}

class CemberDaire{

    private int yariCap;
    public final static double PI = 3.14;

    public CemberDaire(int r){
        this.yariCap = r;
    }

    public double cevreBul(){
        return 2 * PI * yariCap;
    }

    public double alanBul(){
        return PI * Math.pow(yariCap, 2);
    }




}
```

Çözüm 2


```
import java.util.Arrays;

public class Soru2 {
    public static void main(String[] args) {

        Ogrenci tumOgrenciler[]=new Ogrenci[100];

        for(int i=0; i<100; i++){
            int rastgeleID = (int)(Math.random() * 5000);
            int rastgeleNotDegeri = (int)(Math.random() * 100);

            Ogrenci yeni = new Ogrenci(rastgeleID, rastgeleNotDegeri);
            tumOgrenciler[i] = yeni;

        }

        ogrencileriNotlarinaGoreSirala(tumOgrenciler);

        for (int i=0; i<100; i++){
           tumOgrenciler[i].ogrenciBilgileriniYazdir();
        }
    }

    private static void ogrencileriNotlarinaGoreSirala(Ogrenci[] tumOgrenciler) {

        for(int i=0; i<tumOgrenciler.length - 1 ; i++){
            //en küçük elemanın ve indexinin bulunması
            int oankiEnBuyukSayi = tumOgrenciler[i].getNotDegeri();
            int oankiEnBuyukElemaninIndex = i ;

            for(int j = i+1 ; j < tumOgrenciler.length ; j++){
                if(oankiEnBuyukSayi < tumOgrenciler[j].getNotDegeri()){
                    oankiEnBuyukSayi = tumOgrenciler[j].getNotDegeri();
                    oankiEnBuyukElemaninIndex = j;
                }
            }
            //eğer gerekli ise değerler yer değiştirir
            if(oankiEnBuyukElemaninIndex != i){
                tumOgrenciler[oankiEnBuyukElemaninIndex] = tumOgrenciler[i];
                tumOgrenciler[i].setNotDegeri(oankiEnBuyukSayi);
            }


        }

    }
}

class Ogrenci{

    private int id;
    private int notDegeri;

    public Ogrenci(int id, int notDegeri) {
        this.id = id;
        this.notDegeri = notDegeri;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getNotDegeri() {
        return notDegeri;
    }

    public void setNotDegeri(int notDegeri) {
        this.notDegeri = notDegeri;
    }

    public void ogrenciBilgileriniYazdir(){
        System.out.println("id :"+id + " not:"+notDegeri);
    }
}
```

Çözüm 3


```
public class Soru3 {
    public static void main(String[] args) {

        System.out.println("Topla : " + HesapMakinesi.topla(10,12,14));
        System.out.println("Fark : " + HesapMakinesi.cikar(15,4,3));
        System.out.println("Çarp : " + HesapMakinesi.carp(10,12,5));
        if(HesapMakinesi.bol(0,5,2) != -1){
            System.out.println("Böl : " + HesapMakinesi.bol(0,5,2));
        }else{
            System.out.println("Bölme işlemi başarısız");
        }

    }
}

class HesapMakinesi{

    public static int topla(int... parametreler){
        int toplam =0;
        for(int parametre : parametreler){
            toplam = toplam + parametre;
        }
        return toplam;
    }

    public static int cikar(int... parametreler){
        int fark = parametreler[0];
        for (int i=1; i<parametreler.length ; i++){
            fark = fark - parametreler[i];
        }
        return  fark;
    }

    public static int carp(int... parametreler){
        int carpim = 1;
        for(int parametre : parametreler){
            carpim = carpim * parametre;
        }
        return carpim;
    }

    public static double bol(int... parametreler){
        int bolum = parametreler[0];
        for(int i=1; i<parametreler.length; i++){
            if(parametreler[i] != 0){
                bolum = bolum / parametreler[i];
            }else{
                System.out.println("parametrelerden biri 0 bölme yapılamadı");
                return -1;
            }
        }

        return bolum;



}
```

Çözüm 4


```
public class Soru4 {
    public static void main(String[] args) {
        BankaHesap emre=new BankaHesap(123,500);
        BankaHesap hasan=new BankaHesap(456,1500);
        BankaHesap ayse=new BankaHesap(987,200);
        ayse.paraYatir(500);
        emre.paraCek(600);
        hasan.paraCek(450);
        BankaHesap.bankaOzetiGoster();
        ayse.kiyasla(emre);
        emre.kiyasla(hasan);
    }
}

class BankaHesap{
    private int hesapNo;
    private int hesapBakiye;
    private static int tumBankaBakiyesi=0;
    private static int tumHesapSayisi=0;
    private static int operasyonSayisi=0;
    public BankaHesap(int hesapNo, int hesapBakiye) {
        this.hesapNo = hesapNo;
        this.hesapBakiye = hesapBakiye;
        tumBankaBakiyesi += hesapBakiye;
        tumHesapSayisi ++;
    }
    public int getHesapNo() {
        return hesapNo;
    }
    public void setHesapNo(int hesapNo) {
        this.hesapNo = hesapNo;
    }
    public int getHesapBakiye() {
        return hesapBakiye;
    }
    public void setHesapBakiye(int hesapBakiye) {
        this.hesapBakiye = hesapBakiye;
    }
    public void kiyasla(BankaHesap kiyaslanacakHesap){
        if(this.getHesapBakiye() < kiyaslanacakHesap.getHesapBakiye()){
            System.out.println(this.getHesapNo()+" nolu kişinin parası  "+kiyaslanacakHesap.getHesapNo()+" nolu
kişinin parasından azdır");
        }else if(this.getHesapBakiye() > kiyaslanacakHesap.getHesapBakiye()){
            System.out.println(this.getHesapNo()+" nolu kişinin parası  "+kiyaslanacakHesap.getHesapNo()+" nolu
kişinin parasından çoktur");
        }else{
            System.out.println(this.getHesapNo()+" nolu kişinin parası  "+kiyaslanacakHesap.getHesapNo()+" nolu
kişinin parasına eşittir");
        }
    }

    public void paraYatir(int paraMiktari){
        this.hesapBakiye += paraMiktari;
        operasyonSayisi++;
    }

    public void paraCek(int paraMiktari){
        if(this.hesapBakiye >= paraMiktari){
            this.hesapBakiye -= paraMiktari;
            operasyonSayisi++;
            tumBankaBakiyesi -= paraMiktari;
        }else{
            System.out.println("Hesabınızda yeterli para yok");
        }
    }

    public static void bankaOzetiGoster(){
        System.out.println("Bankadaki hesap sayısı :" +tumHesapSayisi);
        System.out.println("Bankadaki toplam para :" +tumBankaBakiyesi);
        System.out.println("Bankada yapılan tüm operasyonların sayısı :" +operasyonSayisi);
    }
}
```