**super() metodu**

Bir alt-sınıf ne zaman üst-sınıfına erişmek isterse super anahtar sözcüğünü kullanabilir. super ‘in kullanımı iki türlü olur. Birincisi, üst-sınıfa ait nesne yaratmak içindir. İkincisi, üst-sınıfın öğelerine erişmek içindir. Bundan sonraki örneklerde, bu işlerin nasıl yapıldığını göreceğiz.

**a.**       **Üst-sınıfa ait nesne yaratma:**

Bir alt-sınıf, super() metodunu kullanarak, üst sınıfının bir nesnesini yaratabilir ve onun değişkenlerine değer atayabilir. AltKutu sınıfında ust-kutuya ait anlık değişkenleri kullanarak nesne yarattık. Üst-sınıfın iç-değişkenleri private olmadığı sürece, bunu yapmak mümkündür. Ama, üst-sınıfın değişkenleri private damgalı olduğunda, alt-sınıftaki kodlar, onlara erişemeyecektir. Çoğunlukla, üst-sınıfın öğelerinin private olmasını isteriz. Böylece üst-sınıfın yapısını diğer sınıflardan saklarız. Buna encapsulation denir. Bu durumlarda, üst-sınıfın öğelerine erişmenin yolu super()metodunu kullanmaktır.

Bir alt-sınıf içinde üst-sınıfa ait bir nesne yaratmak için sözdizimi şöyledir:

        super(formal\_parametreler\_listesi);

Üst-sınıfta overload edilmiş constructor metotları varsa, alt-sınıftan super() metodu ile onların her birisi çağrılabilir.  Çağrıda, formal parametreler listesine, üst-sınıfa ait olarak yaratılacak nesneyi belirleyecek gerçek parametreler konur. Bu demektir ki, üst-sınıfta overload edilen constructor’lardan hangisi isteniyorsa, onun parametreleri yerine gerçek değerler konulur. Ancak, super() metodu alt-sınıftaki constructor içine ilk deyim olarak yazılmalıdır; çünkü constructor çağrılınca ilk işi nesneyi yaratmaktır. Nesne yaratılmadan, onunla ilgili hiçbir iş yapılamaz.

Aşağıdaki örnek, AltKutu sınıfında bir yapıcı (constructor metodu) tanımlamaktadır. Yapıcı içinde tanımlanan AltKutu constructoru, super() metodunu çağırmaktadır. super() metodu, üst-sınıfa ait bir nesne constructor yerine geçer. Üst-sınıfta overload edilmiş constructorlar tanımlı ise, hangisini çağıracağını, kullanılan parametreler belirler. Çünkü, java derleyicisi overload edilen fonksiyonları, parametreleri yardımıyla birbirinden ayırır.

/\* super() metodunun kullanılışı:

   alt-sınıf, super() metodunu kullanarak

   üst-sınıfın bir nesnesini yaratabilir ve

   onun değişkenlerine değer atayabilir.

\*/

class **AltKutu extends Kutu** {

  double agr;  //kutu’nun ağırlığını tutacak değişken

  /\* super() metodu ile en, boy ve yukseklik

     değerlerine atama yapıyor. \*/

**AltKutu**(double e, double b, double y, double a) {

    super(e, b, y); // ust-sınıfa ait constructor’u çağrılıyor

    agr = a;

  }

}

Dikkat edilirse, yukarıdaki AltKutu sınıfı içinde tanımlanan AltKutu() constructoru super(e,b,y)metodunu çağırmaktadır. Bu metot, üst-sınıftaki Kutu(e,b,y) constructor’unun yerine geçmektedir. Bunu daha iyi anlamak için, önce yazdığımız Kutu sınıfını anımsamak yetecektir. Tabii, uygulama programı bu satıra gelmeden önce e,b,y değerlerine gerçek parametre değerlerini atamış olacaktır. Dolayısıyla, constructor, gerçek parametrelerle çağrılıyor. Aksi halde nesneyi yaratamaz, derleme hatası oluşur.

class **Kutu** {

double en;

double boy;

double yukseklik;

**Kutu**(double e, double b, double y) {

en = e;

boy = b;

yukseklik = y;

}

AltKutu constructoru gerçek parametrelerle çağrılınca Kutu sınıfına ait bir nesne yaratılmış ve onun en, boy ve yükseklik değerleri nesneye girilmiş olacaktır. AltKutu alt-sınıfı üst-sınıfında olmayan agr(ağırlık) değişkenine sahiptir. Dolayısıyla AltKutu() constructoru nesneyi yaratırken ona da bir değer atamalıdır. Constructor içindeki son deyim olan agr = a; deyimi bu atamayı yapacaktır.

Örneğimizde, super() metodu 3 gerçek parametre ile çağrılmaktadır. Gerektiğinde bu sayı değişebilir. Üst-sınıfta constructor metodu overload edildiği için, aynı adlı farlı constructorları çağırmak mümkündür. Tabii, üst-sınıfta overload edilen bu metotların önceden tanımlanmış olması gerekir.

**Liste 4:**

Yukarıda söylenenleri topluca görebilmek için Kutu sınıfını, overload edilen constructorlarını içerecek biçimde yeniden yazalım.

           // Kutu ve AltKutu’nun tanımları.

class Kutu {

  private double en;

  private double boy;

  private double yukseklik;

 /\* Parametreli constructor.

     Constructor metodunun parametresi kendi

     sınıfının tipindendir. \*/

  Kutu(Kutu ob) {

    Width  = ob.en;

    boy    = ob.boy;

    yukseklik  = ob.yukseklik;

  }

  /\*

   \* Parametreli constructor.

   \* Constructor çağrılırken formal parametreler

     yerine gerçek parametreler girilir.

  \*/

  Kutu(double e, double b, double y) {

    en  = e;

    boy = b;

    yukseklik  = y;

  }

  /\*

   \* Parametresiz constructor.

   \* Değişkenlere geçici -1 değeri atanmıştır.

   \* Program koşarken gerçek değerleri alacaktır.

  \*/

  Kutu() {

    en  = -1;         // geçici değer

    boy = -1;         // geçici değer

    yukseklik  = -1;  // geçici değer

  }

  /\*

   \* Tek parametreli constructor.

   \* Boyutlar eşit kılınarak, bir küp yaratılıyor.

  \*/

  Kutu(double uzunluk) {

    en = boy = yukseklik = uzunluk;

  }

  // Kutu’nun hacmini hesaplayan metodun tanımı

  double hacim() {

    return en \* boy \* yukseklik;

  }

}

**Alıştırmalar:**

1.        Bu sınıfı derleyiniz ve Kutu.class  bytekodunu ayrı bir dizine koyunuz. Derlemek için

javac Kutu.java

komutunu çalıştırınız.

**Liste 5:**

İkinci iş olarak, Kutu sınıfı için yaptıklarımızı  AltKutu alt-sınıfı için de yaparak onu yeniden yazalım. Alt-sınıfta overload edilen constructorlara dikkat ediniz.

  // AltKutu bir alt-sınıftır; Kutu adlı üst-sınıfın bütün

  // constructorlarını etkin kılar.

class AltKutu extends Kutu {

  double agr; // kutu’nun ağırlığı

/\* Bu constructor, super() metodu ile üst-sınıfa ait

 \* bir nesne yaratıyor. Yaratılan nesne’nin

 \* adresi ob ile gösteriliyor (referans, pointer).

 \* constructor’un parametresi kendi sınıfının tipindendir.

\*/

  AltKutu(AltKutu ob) {

    super(ob);

    agr = ob.agr;

  }

  /\* Bu constructor çağrılınca bütün

   \* değişkenlere değer atar

  \*/

  AltKutu(double e, double b, double y, double a) {

    super(e, b, y); // superclass constructor çağrımı

    agr = a;

  }

          // default constructor

  AltKutu() {

    super();

    agr = -1;

  }

          // Bu constructor küp yaratır

  AltKutu(double uzunluk, double a) {

    super(uzunluk);

    agr = a;

  }

}

**Alıştırmalar:**

1.       Bu sınıfı derleyiniz ve AltKutu.class  bytekodunu Kutu.class bytekodunun bulunduğu dizine koyunuz.

2.       Aşağıdaki program parçası yukarıdaki  AltKutu sınıfından alınmıştır. Bu program parçasını çözümleyiniz (yani, içerdiği her sözcüğün ve deyimin anlamını açıklayınız).

**AltKutu**(AltKutu ob) {

  super(ob);

  agr = ob.agr;

}

3.     Aşağıdaki programı çözümleyiniz.

class **ColorKutu extends Kutu** {

  int color;                // Kutu’ın rengi

**ColorKutu**(double e, double b, double y, int c) {

    en = e;

    boy = b;

    yukseklik = y;

    color = c;

  }

}

**Liste 6:**

Son iş olarak, Kutu sınıfı ile AltKutu alt-sınıfı ile ilgili uygulama yapmak için aşağıdaki UygulamaPrg uygulama programını yazalım. Uygulama programları mutlaka main() metodunu içermelidir.

class UygulaKutu {

  public static void main(String args[]) {

    AltKutu kutu1 = new AltKutu(10, 20, 15, 34.3);

    AltKutu kutu2 = new AltKutu(2, 3, 4, 0.076);

    AltKutu kutu3 = new AltKutu();        // default

    AltKutu mycube = new AltKutu(3, 2);

    AltKutu myclone = new AltKutu(kutu1);

    double vol;

    vol = kutu1.hacim();

    System.out.println("kutu1 in hacmi = " + vol);

    System.out.println("kutu1 in hacmi = " + kutu1.agr);

    System.out.println();

    vol = kutu2.hacim();

    System.out.println("kutu2 nin hacmi = " + vol);

    System.out.println("kutu2 nin hacmi = " + kutu2.agr);

    System.out.println();

    vol = kutu3.hacim();

    System.out.println("kutu3 ‘ün hacmi = " + vol);

    System.out.println("kutu3 ‘ün hacmi = " + kutu3.agr);

    System.out.println();

    vol = myclone.hacim();

    System.out.println("myclone ‘ın hacmi = " + vol);

    System.out.println("myclone ‘ın hacmi = " + myclone.agr);

    System.out.println();

    vol = mycube.hacim();

    System.out.println("mycube ‘ün hacmi = " + vol);

    System.out.println("mycube ‘ün hacmi = " + mycube.agr);

    System.out.println();

  }

}

**Alıştırmalar:**

1.              Bu sınıfı derleyiniz, çıkan UygulaKutu.class bytekodunu AltKutu.class  bytekodu ile Kutu.class bytekodunun bulunduğu dizine koyunuz. Sonra java interpreter ile çalıştırınız:

                                               java UygulaKutu

2.              Çok basamaklı alt-sınıflar tanımlanmışsa, super() metodu daima en yakın üst-sınıfa ait nesne yaratır. Yaratılmak istenen nesnenin ait olduğu sınıf içinde super() metodu kullanılamaz. Bunu görmek için, yukarıdaki Kutu sınıfında super() metodunu kullanmayı deneyiniz.

3.

AltKutu(AltKutu ob) {

  super(ob);

  agr = ob.agr;

}

constructor’undaki super(ob) metodu, Kutu sınıfı içindeki Kutu(Kutu ob) constructor’unu etkin kılar; yani onu çalıştırır. Bu metottaki son deyim olan agr = ob.agr; deyimi  kaldırılırsa ne olur? Programı değiştirerek deneyiniz. Nedenini açıklayınız.

**b.**        **Üst-sınıftaki öğelere erişme:**

Bir alt-sınıf, super() metodunu kullanarak, üst sınıfının bir nesnesini yaratabilir ve onun değişkenlerine erişebilirdi.  Şimdi, üst-sınıfa ait nesne yaratmadan onun öğelerine super anahtar sözcüğü ile erişilebileceğini göstereceğiz.

**Liste 7:**

Bir sınıftaki öğelere erişmek için this anahtar sözcüğünü kullanıyorduk. Ona benzer olarak super anahtar sözcüğünü kullanarak üst-sınıfın öğelerine erişebiliriz. Bunu aşağıdaki örnek üzerinde görelim.

 // Üst sınıfta aynı adlı öğeye erişim.

class A {

  int i;

}

    // Alt-sınıf tanımı

class B extends A {

  int i;       // i değişkeni B nin bir iç öğesii olduğundan,

                                   //B alt-sınıfı A ‘nın i öğesini doğrudan göremez.

  B(int a, int b) {

    super.i = a;      //  A içindeki  i

    i = b;            //  B içindeki i

  }

  void show() {

    System.out.println("i nin A ust-sınıfındaki degeri : " + super.i);

    System.out.println("i nin B alt-sınıfındaki degeri : " + i);  }

}

class UygulamaPrg {

  public static void main(String args[]) {

    B subOb = new B(1, 2);

    subOb.show();

  }

}