

1 A. Java Temelleri – Bazı Temel Sorular

Java'nın en önemli özelliği nedir?

Java platformdan bağımsız bir dildir.

Platform bağımsızlığından kastınız nedir?

Platform bağımsızlığı, Java kodunu tek bir platformda (örn. Windows) yazıp derleyebileceğimiz ve sınıfı desteklenen herhangi bir platformda (örn. Linux, Solaris, vb.) çalıştırabileceğimiz anlamına gelir.

Java neden platformdan bağımsızdır?

Java'nın en eşsiz özelliği platformdan bağımsız olmasıdır. Herhangi bir programlama dilinde kaynak kodu çalıştırılabilir koda derlenir. Bu, tüm platformlarda çalıştırılmaz. Javac bir Java programını derlediğinde, .class dosyası adı verilen yürütülebilir bir dosya oluşturur. class dosyası byte kodları içerir. Byte kodları yalnızca JVM'ler tarafından yorumlanır. Bu JVM'ler tüm platformlarda mevcut olduğundan, bu byte kodunu herhangi bir platformda çalıştırabiliriz. Windows ortamında üretilen byte kodlar Linux ortamında da çalıştırılabilmektedir. Bu, Java platformunu bağımsız kılar.

JVM nedir?

JVM, derlenmiş Java sınıfı dosyaları için bir çalışma zamanı ortamı olan Java Sanal Makinesidir. JVM, derleyiciler tarafından üretilen sınıf dosyalarını çalıştırır.

JVM'nin platformu bağımsız mı?

JVM'ler platformdan bağımsız değildir. JVM'ler satıcı tarafından sağlanan platforma özel çalışma zamanı uygulamasıdır.

JDK ile JVM arasındaki fark nedir?

JDK, geliştirme amaçlı olan Java Geliştirme Kitidir ve yürütme ortamını da içerir. Ancak JVM tamamen bir çalışma ortamıdır ve dolayısıyla kaynak dosyalarınızı JVM kullanarak derleyemezsiniz.

İşaretçi nedir ve Java işaretçileri destekler mi?

İşaretçi, bir bellek konumuna yönelik bir başvuru tanıtıcısıdır. İşaretçilerin yanlış kullanılması bellek sızıntılarına ve güvenilirlik sorunlarına yol açar, dolayısıyla Java işaretçilerin kullanımını desteklemez.

Kavanoz nedir?

Jar, Java arşiv dosyası anlamına gelir. Kavanozlar Jar.exe aracı kullanılarak oluşturulur. Jar dosyaları, .class dosyalarını, uygulamamızda kullanılan diğer kaynakları ve manifest dosyasını içerir. Manifest dosyası, ana yöntemle sınıf adını içerir.jar, sıkıştırılmış .class dosyalarını içerir. Jvm, bu .class dosyalarını bu jar dosyasını açmadan bulur.

Java'daki farklı bellek türleri nelerdir?

Java'da kullanılan iki tür bellek vardır. Bunlara yığın belleği ve yığın belleği denir. Yığın belleği, nesnelerin ilkel türlerini ve adreslerini saklar. Nesne değerleri yığın hafızasında saklanır. Yığındaki bir nesne referansı, yalnızca o nesnenin yığın hafızasında tutulduğu yeri belirten bir adrestir.

Test test1 = yeni Test();

```
Test test2 = yeni Test();
test2 = test1;
```

Bunu yazarken aslında yaptığınız şey test1 nesnesinin adresini test2 nesnesine atamaktır. Test1'in bellek adresinin 0x33d444 ve test2'nin adresinin 0x99f775 olduğunu varsayalım. Yukarıdaki atamayı gerçekleştirdikten sonra test2 artık bu adresi yığın belleğinde tutar: 0x99f775, bu test1 ile aynı nesneye karşılık gelir. Yığındaki test2 nesnesi hâlâ mevcut ancak ona erişilemiyor. Bunun nedeni, bu yeniden atamanın test2'nin yığında tuttuğu eski adresin üzerine yazmasıdır. Bu tür bir yeniden atama, yığındaki aynı nesneye iki yığın referansı yapar.

Java'da bu iki farklı bellek türünün bulunduğunu bilmekte fayda var. Yığın belleği programın belleğidir ve yığın belleği programın dışında bulunur.

Bir Java programcısı olarak, C++ programcılarının ortak baş ağrısı olan bellek tahsisi ve bellek alanının kurtarılması konusunu doğrudan ele almanıza gerek yoktur. Yeni bir nesneye ihtiyaç duyduğunuzda Java gerekli belleği ayırır. Bir nesneyle işiniz bittiğinde, Java'nın çöp toplama özelliği aracılığıyla bellek sizin için otomatik olarak geri kazanılır.

Çöp toplama, artık kullanılabilir bir referansı olmayan nesneleri arayarak arka planda bir iş parçası olarak çalışır. Onları bulduğunda yok eder ve hafızayı geri alır.

2. Java'da Sınıf ve Nesne Nedir?

Java'daki İlk Program:

```
halk sınıf İlkProgram {
    halk statik void main(String[] args ) {
        Sistem. out .println( "Merhaba Dünya!" );
    }
}
```

Sınıf: Java'da bir program sınıf bildirimiyle yazılır. Bu, nesneler yaratabileceğimiz temel bir plandır. Tüm program bir sınıf tanımının içine alınmıştır.

Programın gövdesi bir ana yöntemde bulunur. Java'da yürütme ana yöntemden başlar.

"Sistem", Java.lang paketindeki bir sınıftır. "out", System sınıfının statik bir üyesidir ve Java.io.PrintStream'in bir örneğidir. "println", Java.io.PrintStream'in bir yöntemidir. Bu yöntem, mesajı çıktı hedefine yazdırmak için aşırı yüklenmiştir.

Nesne: Bir nesne bir sınıfın örneğidir.

Bir nesne oluşturmak için "yeni" anahtar kelimesini kullanırız

```
paket paket1;

halk sınıf nesne örneği {
    int bir = 1;
    int b = 2;
```

```

    halk geçersiz disp() {
        nesne örneği obj = yeni nesne örneği();
        Sistem. out .println( " b değeri " + obj .b );
    }
    halk statik void main(String[] args ) {
        nesne örneği ob = yeni nesne örneği();
        Sistem. out .println( "bir değer " + ob . a );
        ob .disp();
    }
}

```

Yukarıdaki örnekte iki nesne oluşturduk.

```

nesne örneği obj = yeni nesne örneği();
nesne örneği ob = yeni nesne örneği();

```

Genel Örnekler:

Bir ev inşa etmek istiyorsanız bir planınız olacaktır. Plan bir dersten başka bir şey değil. İnşa edilen ev bir nesneden başka bir şey değildir.

Aynı planı kullanarak istediğiniz sayıda ev oluşturabilirsiniz. Bu, bir sınıfı kullanarak istediğiniz sayıda nesne oluşturabileceğiniz anlamına gelir.

Her evin sahip olduğu oda sayısı farklılık gösterebilir. Bu, nesnelerin özelliklerinin farklı olacağı anlamına gelir.

Örnek:

```

paket paket1;

sınıf Ev planı {

    halk int yatak odaları ;
    halk int banyolar ;
}

halk sınıf Ev {
    halk statik void main(String[] args ) {
        Ev planı bir = yeni Ev planı ();
        A . yatak odaları = 2;
        A . banyolar = 1;

        Ev planı b = yeni Ev planı ();
        B . yatak odaları = 3;
        B . banyolar = 2;

        Sistem. out .println( "Evdeki Yatak Odaları a " + a . yatak odaları
    );
        Sistem. out .println( "B Evindeki Yatak Odaları " + b . yatak
    odaları );

        Sistem. out .println( "Evdeki Banyolar a " + a . banyolar );
        Sistem. out .println( "B Evindeki Banyolar " + b . banyolar );

    }
}

```

2A. Java Temelleri – Sınıf ve Nesne Soruları

Sınıf nedir?

Bir sınıf basitçe bir nesne türünün temsilidir. Bir nesnenin ayrıntılarını açıklayan plan/plan/şablondur.

Tüm sınıfların temel sınıfı nedir?

java.lang.Object

Path ve Classpath arasındaki fark nedir?

Yol ve Sınıf Yolu, işletim sistemi düzeyindeki ortam değişkenleridir. Yol, sistemin çalıştırılabilir (.exe) dosyalarını nerede bulabileceğini tanımlamak için kullanılır ve sınıf yolu, .class dosyalarının konumunu belirtmek için kullanılır.

Java'daki main() yöntemini açıklayın?

Main() yöntemi, tüm Java uygulamaları için yürütmenin başlangıç noktasıdır.

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

String args[], komut satırı argümanlarından aktarmamız gereken dize nesneleri dizisidir. Her Java uygulamasında en az bir ana yöntem bulunmalıdır.

Main() yönteminin argümanı nedir?

main() yöntemi, argüman olarak bir String nesnesi dizisini kabul eder.

Main() yönteminde genel ve statik bildirimin sırası önemli mi?

Hayır. Fark etmez ama void her zaman main()'dan önce gelmelidir.

Hangi sınıf diğer tüm sınıflar tarafından genişletilir?

Object sınıfı diğer tüm sınıflar tarafından genişletilir.

Tüm Java sınıflarında zorunlu olarak bir main() yöntemi bildirilmeli mi?

Hayır, gerekli değil. main() yöntemi yalnızca kaynak sınıf bir Java uygulamasıysa tanımlanmalıdır.

Main() yönteminin dönüş türü nedir?

Main() yöntemi, geçersiz ilan edilen hiçbir şeyi döndürmez.

Tek dosyada birden fazla sınıfımız olabilir mi?

Evet. Tek dosyada birden fazla sınıfımız olabilir ancak insanlar bunu nadiren yapar ve önerilmez. File'da birden fazla sınıfımız olabilir ancak yalnızca bir sınıf herkese açık hale getirilebilir. File public'te iki sınıf yapmaya çalıştığımızda aşağıdaki derleme hatasıyla karşılaşırız. "Public türü kendi dosyasında tanımlanmalıdır".

paket soruları;

```
halk sınıf test { //Hata: Dosyada genel tür tanımlanmalı
    halk statik geçersiz ana() {
        Sistem. out .println( "merhaba" );
    }
}
```

```

halk sınıf test sınıfı
{
    halk statik void main(String[] args ) {
        Sistem. out .println( "merhaba" );
        Ölçek. ana ();
    }
}

```

Java'da tanımlayıcılar nelerdir?

Tanımlayıcılar Java programındaki isimlerdir. Tanımlayıcılar sınıf adı, yöntem adı veya değişken adı olabilir.

Sınıf yolu nedir?

.class dosyalarımızın kaydedildiği yola sınıf yolu denir.

JVM, belirtilen sınıf yolunu kullanarak .class dosyalarını arar.

Sınıf yolu CLASSPATH ortam değişkeni kullanılarak belirtilir.

CLASSPATH ortam değişkeni birden fazla değer içerebilir.

Birden fazla değer içeren CLASSPATH değişkeni noktalı virgülle ayrılır.

Komut isteminden sınıf yolunu ayarlama örneği: set CLASSPATH= C:Program Files\Java\jdk1.6.0_25\bin;.;

classpath'e yalnızca ana dizinlerin eklenmesi gerekir. Java derleyicisi uygun paketleri ve sınıfları arayacaktır.

3. Java Değişkenleri

Değişkenler değer sahipleridir. Bir değer bildirmek için sözdizimi şöyledir

veri_türü değişken_adı = değer;

int a = 9;

karakter c1 = 'c'

Örnek:

paket FPPaket;

```

halk sınıf değişkenleriex {
    halk statik void main(String[] args ) {
        int bir = 10;
        int b = 20;
        karakter c1 = 'j' ;
        Sistem. out .println( "bir değer " + a );
        Sistem. out .println( "b değeri " + b );
        Sistem. out .println( "C1 eşittir " + c1 );
        int c = a * b ;
        Sistem. out .println( "c değeri " + c );
    }
}

```

```
}
```

Farklı Değişken Türleri:

1. Sınıf değişkeni (Statik alanlar) - Sınıf değişkenleri, sınıf gövdesi içinde, herhangi bir yöntem veya blok dışında bildirilen ve 'statik' anahtar kelimeyle bildirilen değişkenlerdir. Sınıf değişkenleri en uzun kapsama sahiptir. Sınıf yüklendiğinde oluşturulurlar ve sınıf JVM'de yüklü kaldığı sürece bellekte kalırlar.
2. Örnek değişkenler (Statik olmayan alanlar) - Örnek değişken, sınıf gövdesi içinde, herhangi bir yöntemin veya bloğun dışında bildirilen ve 'statik' anahtar kelime olmadan bildirilen değişkenlerdir. Örnek değişkenler ikinci en yüksek kapsama sahiptir. Örnek değişkenleri, yeni bir sınıf örneği oluşturulduğunda oluşturulur ve örnek bellekten kaldırılincaya kadar aktif kalır.
3. Yerel Değişkenler - Yerel değişkenler, bir yöntem gövdesi içinde bildirilen değişkenlerdir. Yalnızca bildirildiği yöntem yığında kaldığı sürece yaşarlar.
4. Blok değişkenleri - Blok değişkenleri, init bloğu gibi bir blok içinde veya for döngüsü içinde bildirilen değişkenlerdir. Yalnızca bloğun yürütülmesi sırasında yaşarlar ve en kısa yaşayan değişkenlerdir.

Örnek:

```

halk sınıf DeğişkenlerÖrnek {
    statik int c = 6; //Sınıf Değişkeni
    int d = 8; //Örnek değişkeni
    halk statik geçersiz main (Dize[] args ) {
        int bir = 5; //Yerel değişken
        int b = a * 5; //Yerel değişken
        Sistem. out .println( "bir değer " + a );

        Sistem. out .println( "b değeri " + b );

        için ( int ben = 0; ben <4; ben ++ ) {
            int j = 2; //Değişkeni Blokla
            Sistem. out .println( a * j );
        }
    }
}

```

3 A. Java Değişkenleri – Soruları – Bölüm 1

Değişken ne anlama geliyor?

Değişkenler hafızada değerleri tutabilen konumlardır.

Değişkenler nasıl bildirilir?

Değişkenler, yöntem tanımının herhangi bir yerinde bildirilebilir ve bildirimleri sırasında başlatılabilir. Genellikle tanımın başında kullanımdan önce bildirilirler. Aynı veri tipine sahip değişkenler birlikte bildirilebilir. Yerel değişkenlere kullanımdan önce bir değer verilmelidir.

Değişkenlere nasıl değer atarsınız?

Değerler atama operatörü = kullanılarak değişkenlere atanır.

Java'daki değişken türleri nelerdir ve bunların kullanımları nelerdir?

Java'nın dört tür değişkeni vardır.

- Değişkenleri Blokla
- Yerel Değişkenler
- Örnek değişkenler
- Sınıf Değişkenleri

Blok Değişkenleri, init blokları veya for döngüleri gibi blokların içinde kullanılır ve tek bir bloğun ihtiyaç duyduğu bilgileri depolamak için kullanılır.

Yerel değişkenler yöntemlerin içinde geçici değişkenler olarak kullanılır ve tek bir yöntemin ihtiyaç duyduğu bilgileri depolamak için kullanılır.

Örnek değişkenler (üye değişkenler), belirli bir nesnenin niteliklerini veya durumunu tanımlamak için kullanılır ve nesnelerde birden fazla yöntemin ihtiyaç duyduğu bilgileri depolamak için kullanılır.

Sınıf değişkenleri bir sınıf ve sınıfın tüm örnekleri için geneldir ve aynı sınıfın farklı nesneleri arasında iletişim kurmak veya genel durumları takip etmek için kullanışlıdır.

Gerçek nedir? Kaç çeşit harf vardır?

Bir değişmez değer, türün bu değerinin nasıl davrandığını açıkladığı belirli bir türün değerini temsil eder. Değişmez değerlerin farklı türleri vardır:

- a) sayı değişmezleri (1,23)
- b) karakter değişmezleri ('c', 'd')
- c) boolean değişmezleri (doğru, yanlış)
- d) dize değişmezleri vb. ("merhaba")

Yerel değişkenler nelerdir?

Yerel değişkenler yöntemlerin içinde bildirilen değişkenlerdir. Yerel değişkenlere erişmeden önce başlatılmalıdır.

Örnek değişkenler nelerdir?

Örnek değişkenler, statik anahtar kelime olmadan sınıf düzeyinde tanımlanan değişkenlerdir . Örnek değişkenleri, otomatik olarak varsayılan değerlerine başlatıldıkları için kullanılmadan önce başlatılmalarına gerek yoktur.

Örnek değişkenlerin kapsamı veya yaşam süresi nedir?

Nesne yeni operatör kullanılarak başlatıldığında, bellekte değişkenler tahsis edilir. Örnek değişkenleri, örnek çöp toplanana kadar bellekte kalır.

Java'da yerel değişkenlerin kapsamını veya ömrünü açıklayın?

Yerel değişkenler, bir yöntemin içinde tanımlanan değişkenlerdir. Yöntem oluşturulduğunda yığın belleğinde yerel değişkenler oluşturulur ve yöntemin yürütülmesi tamamlandıktan sonra bu değişken bellekten silinir.

Java'daki referans değişkenleri nelerdir?

Java'da nesnelere erişmek için kullanılan değişkenlere referans değişkenleri denir.

Örn: Çalışan emp=yeni Çalışan();

Yukarıdaki örnekte emp referans değişkenidir.

İlkel değişkenler bellekte ne kadar süreyle bulunur? İlkel değişkenlerin çeşitli kapsamlarını tanımlayın?

İlkel bir değişken bildirilip başlatıldıktan sonra; bellekte ne kadar süre kalacağı değişkenin kapsamına bağlıdır. Bir değişkenin kapsamı, Java sınıfında nerede bildirildiğine göre belirlenir. Aşağıda, bir Java programındaki bir değişkenin, bildirildiği yere bağlı olarak çeşitli kapsamaları verilmiştir.

1. Sınıf değişkeni (Statik alanlar) - Sınıf değişkenleri, sınıf gövdesi içinde, herhangi bir yöntem veya blok dışında bildirilen ve 'statik' anahtar kelimeyle bildirilen değişkenlerdir.

Sınıf değişkenleri en uzun kapsama sahiptir. Sınıf yüklendiğinde oluşturulurlar ve sınıf JVM'de yüklü kaldığı sürece bellekte kalırlar.

2. Örnek değişkenler (Statik olmayan alanlar) - Örnek değişken, sınıf gövdesi içinde, herhangi bir yöntemin veya bloğun dışında bildirilen ve 'statik' anahtar kelime olmadan bildirilen değişkenlerdir.

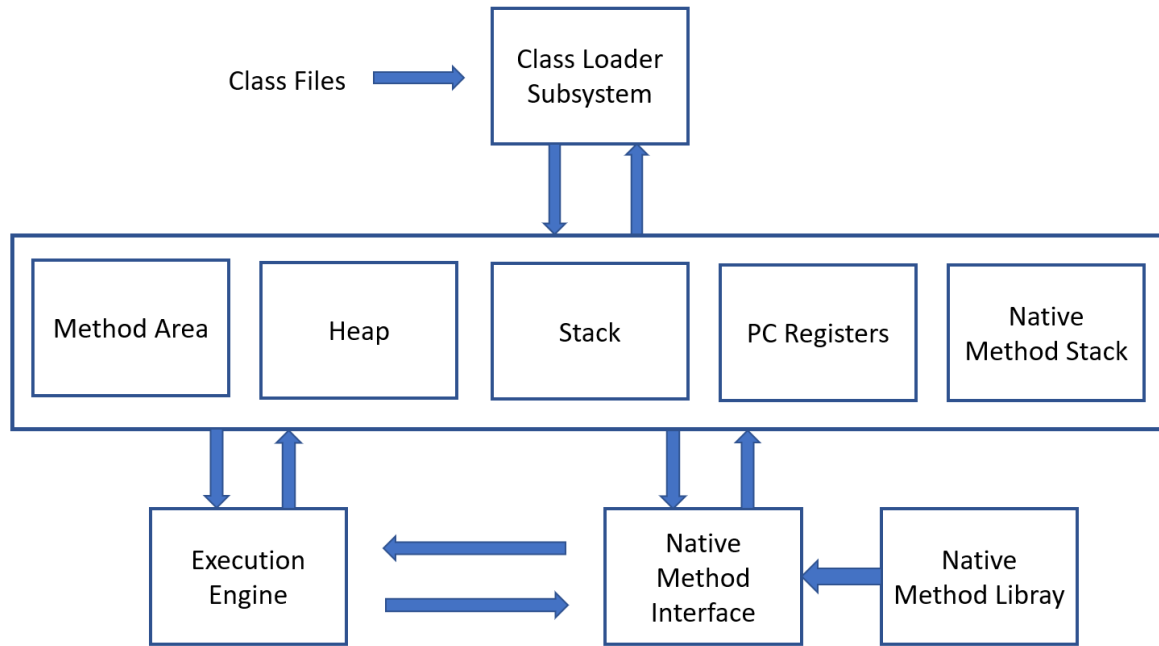
Örnek değişkenler ikinci en yüksek kapsama sahiptir. Örnek değişkenleri, yeni bir sınıf örneği oluşturulduğunda oluşturulur ve örnek bellekten kaldırılıncaya kadar aktif kalır.

3. Yerel Değişkenler - Yerel değişkenler, bir yöntem gövdesi içinde bildirilen değişkenlerdir. Yalnızca bildirildiği yöntem yığında kaldığı sürece yaşarlar.

4. Blok değişkenleri - Blok değişkenleri, init bloğu gibi bir blok içinde veya for döngüsü içinde bildirilen değişkenlerdir. Yalnızca bloğun yürütülmesi sırasında yaşarlar ve en kısa yaşayan değişkenlerdir.

3B. Java Değişkenleri – Soruları – Bölüm 2

Değişkenlerin bellekte nerede oluşturulduğunu açıklayın?



Metod Alanı: Bildirilen tüm sınıf kodunu, metod kodunu ve sınıf değişkenlerini saklar.

Heap Memory: Oluşturulan tüm nesneleri saklar.

JVM Stack Alanı: Metodların çalışma zamanındaki bilgilerini saklar. Örneğin, çalışma zamanında başlatılan yöntem değişkenleri. Ayrıca oluşturulan nesnelere yapılan referansları da saklar.

Nihai değişken nedir?

Bir değişken son değişken olarak bildirilirse değerini değiştiremezsiniz. Sürekli hale gelir.

Örnek:

```

halk sınıf Java Örnekleri {
    statik son int bir = 6;

    halk statik void main(String[] args ) {
        bir = 7; //Hata: Javaexamples.a'nın son alanı atanamıyor
        Sistem. out .println( "bir değer " + a );
    }
}
  
```

Java'da sabit bir değişken nasıl tanımlanır?

static ve **final** olarak bildirilmelidir . Dolayısıyla sınıfın tüm örnekleri için değişkenin yalnızca bir kopyası bulunur ve değer de değiştirilemez.

statik final int MAX_LENGTH = 50; sabite bir örnektir.

4. Java Veri Türleri

Veri türleri, bir değişkenin alabileceği değer türlerini tanımlar.

byte , **short** , **int** ve **long** veri türleri kullanılır.

float ve **double** kullanılır.

char, karakterleri (harfleri) saklamak için kullanılır.

Boolean veri türü, doğru veya yanlış değerini tutan değişkenler için kullanılır.

Byte -128 ile 127 arasındaki tam sayıyı saklayabilir

Short -32.768 ile 32767 arasındaki tam sayıyı saklayabilir

Int -2,147,483,648 ila 2,147,483,647 arasındaki tam sayıyı saklayabilir

Long, -9,223,372,036,854,775,808 ila 9,223,372,036,854,775,807 arasındaki tam sayıyı saklayabilir

Double 15 ondalık rakamı saklayabilir

Float 6 ila 7 ondalık basamak saklayabilir

Boolean doğru ya da yanlış değerini tutar

Char karakterleri tutar.

Türler	Boyut (bayt)	En az değer	Maksimum değer	Kesinlik
Bayt	1	-128	127	+127'den -128'e
Kısa	2	-2^{15}	$2^{15} - 1$	+32.767'den -32.768'e
Dahili	4	-2^{31}	$2^{31} - 1$	+2,147,483,647'den -2,147,483,648'e
Uzun	8	-2^{63}	$2^{63} - 1$	+9,223,372,036,854,775,807'den -9,223,372,036,854,775,808'e
Batmadan yüzmek	4	2^{-149}	$(2 - 2^{-23}) \cdot 2^{127}$	3.402.823,5 E+38'den 1.4 E-45'e
Çift	8	2^{-1074}	$(2 - 2^{-52}) \cdot 2^{1023}$	1,797,693,134,862,315,7 E+308'den 4,9 E-324'e
Karakter	2	0	$2^{16} - 1$	Tüm Unicode karakterler

Örnek:

paket FPPaket;

halk sınıf değişkenleriörnek {

```

    halk statik void main(String[] args ) {
        byte bir = 9;
        int b = 20;
        karakter c1 = 'j' ;
        Sistem. out .println( "bir değer " + a );
        Sistem. out .println( "b değeri " + b );
        Sistem. out .println( "C1 eşittir " + c1 );
        int c = a * b ;
        Sistem. out .println( "c değeri " + c );
    }
}

```

JAVA'da İlkel Döküm:

İlkel dönüşüm, ilkel değerleri bir veri türünden başka bir veri türüne dönüştürmek için kullanılır.

Örneğin, bir float değerine bir int değeri atanabilir, bir int değerine vb. bir double değeri atanabilir.

Büyük bir veri türü değerine küçük bir veri türü değeri atıyorsak herhangi bir dönüşüme gerek yoktur. Ancak küçük bir veri tipine büyük bir veri tipi değeri atıyorsak, değer dönüşümüne ihtiyaç vardır.

Örnek:

Kısa bellekte 2 bayt, int ise bellekte 4 bayt yer kaplar. Bir int'ye kısa bir değer atayabiliriz, ancak bir kısaya int değeri atamak için onu atamamız gerekir.

```

halk sınıf Java Örnekleri {

    statik kısa bir = 1;
    statik int B ;

    statik int c = 2;
    statik kısa d;
    halk statik void main(String[] args ) {

        b = bir ;

        Sistem. out .println( "b değeri " + b );

        //d = c; //Hata int'den short'a dönüştürülemiyor

        d = ( kısa ) c ;

        Sistem. out .println( "d değeri " + d );

    }

}

```

4A. Java Veri Türleri – Soruları – Bölüm 1

Değişken türleri nelerdir?

Değişken türleri, sekiz temel veri türünü, bir sınıfın veya arayüzün adını ve bir diziyi içeren, Java'nın desteklediği herhangi bir veri türü olabilir.

Sekiz ilkel Java türünü adlandırın.

Sekiz temel tür şunlardır: byte, char, short, int, long, float, double ve boolean.

Tip	Bitler	Bayt	Min.	Maksimum	Varsayılan
Bayt	8	1	-29	28-01-2001	0
Kısa	16	2	-217	$2^{16-1} - 1$	0
Dahili	32	4	-233	$2^{32-1} - 1$	0
Uzun	64	8	-265	$2^{64-1} - 1$	0
Batmadan yüzmek	32	4	>YOK	Yok	0,0f
Çift	64	8	Yok	Yok	0.0 gün
Boolean	1	Yok	Yok	Yok	YANLIŞ
Karakter	16	Yok	Yok	Yok	"

Döküm nedir?

İki tür dönüştürme vardır; ilkel sayısal türler arasında dönüştürme ve nesne referansları arasında dönüştürme. Sayısal türler arasında dönüştürme, çift değerler gibi daha büyük değerleri bayt değerleri gibi daha küçük değerlere dönüştürmek için kullanılır. Nesne referansları arasında dönüştürme, bir nesneye uyumlu bir sınıf, arayüz veya dizi tipi referansı ile atıfta bulunmak için kullanılır.

Diziler ilkel veri türleri midir?

Java'da Diziler nesnelerdir.

İlkel Değişmezler nedir?

İlkel Değişmez Değerler, ilkel veri türlerinin değerlerinin kod temsilidir. Örneğin 'a' bir char değişmez değeridir, 100 bir int değişmezidir, 'false' bir boole değişmezidir ve 2345.456 bir çift değişmezidir.

Boolean tipindeki bir değişken hangi değere otomatik olarak başlatılır?

Boolean türünün varsayılan değeri false'tur.

Java'da tür dönüşümü nedir?

Bir türden bir değer başka bir türdeki değişkene atanmasına tür dönüşümü denir.

Örnek:

```
int a = 10;
```

```
uzun b = a;
```

Java'da iki tür dönüşüm vardır:

1) Genişletme dönüşümü

2) Daraltma dönüşümü

Java'da Otomatik tür dönüştürmeyi açıklayın?

Aşağıdaki koşullar karşılanırsa Java otomatik tür dönüşümü yapılır:

- 1) İki tür uyumlu olduğunda
Örn: int, kayan nokta
int doğrudan float değişkenine atanabilir.
- 2) Hedef türü kaynak türünden daha büyüktür.
Örn: int, uzun
Int doğrudan long'a atanabilir. Long veri türü int'den daha büyük olduğu için int long'a atanırsa otomatik tür dönüşümü gerçekleşir. Genişletme Dönüşümü, Otomatik tür dönüştürme altında gelir.

Java'da dönüşümü daraltmayı açıklayın?

Hedef türü kaynak türünden küçük olduğunda Java'da daraltma dönüştürme mekanizmasını kullanırız. Hedef türü kaynak türünden küçükse daraltma dönüştürmesinin manuel olarak yapılması gerekir.

Daraltma dönüşümü yapmak için cast kullanıyoruz. Cast, açık tür dönüşümünden başka bir şey değildir.

Örnek:

uzun a;

bayt b;

b=(bayt)a;

Not: Döküm yalnızca geçerli türlerde yapılacaktır, aksi takdirde classcast Exception atılacaktır.

Java'da double ve float değişkenleri arasındaki fark.

Java'da float bellekte 4 bayt alırken Double bellekte 8 bayt alır. Float tek duyarlıklı kayan noktalı ondalık sayıdır, Double ise çift duyarlıklı ondalık sayıdır.

4B. Java Veri Türleri – Soruları – Bölüm 2

Java'da ilkel döküm nedir?

İlkel Döküm, ilkel değerleri bir veri türünden diğerine dönüştürmek için kullanılır. Örneğin, float veri tipine int değeri atanabilir veya int veri tipine double değeri atanabilir. Döküm örtülü veya açık olabilir.

Örtülü Döküm: Örtülü dökümde dönüşüm, dönüşümü gerçekleştirmek için belirli bir kod yazılmadan otomatik olarak gerçekleşir. Örtülü dönüşüm, bayt gibi daha küçük bir değeri int gibi daha büyük bir veri türüne dönüştürdüğünüzde veya atadığınızda gerçekleşir.

Açık Döküm: Açık dökümde, bir ilkel türden diğerine dönüşümü gerçekleştirmek için kodun özel olarak yazılması gerekir. Açık dönüşüm sözdizimi (data_type) kullanılarak yapılır; burada veri_tipi, dönüşümün uygulandığı veri türüdür. Açık atama, daha küçük bir veri türünü dönüştürdüğünüzde veya daha büyük bir değeri atadığınızda gerçekleşir.

```

paket soruları;

sınıf test sınıfı
{
    halk statik void main(String[] args )
    {
        bayt b = 2;
        int ben = b ; //Örtülü döküm

        int j = 3;
        //byte k = j; //Hata atar
        bayt k = ( bayt ) j ; //Açık döküm
    }
}

```

İlkel veri türlerinin varsayılan değerleri nelerdir?

Varsayılan değerler aşağıdaki örnekte verilmiştir.

```

paket soruları;

halk sınıf test sınıfı
{
    bayt B ;
    kısa S ;
    int Ben ;
    uzun ben ;
    batmadan yüzmek F ;
    çift D ;
    boolean bo ;
    karakter ch ;

    halk statik void main(String[] args ) {
        testsınıfı tc = yeni testsınıfı();

        Sistem. out .println( "baytın varsayılan değeri " + tc .b );
        Sistem. out .println( "kısa varsayılan değeri " + tc .s );
        Sistem. out .println( "int varsayılan değeri: " + tc .i );
        Sistem. out .println( "uzun varsayılan değeri: " + tc .l );
        Sistem. out .println( "float'ın varsayılan değeri " + tc .f );
        Sistem. out .println( "çift varsayılan değeri: " + tc .d );
        Sistem. out .println( "boolean varsayılan değeri: " + tc .bo );
        Sistem. out .println( "karakterin varsayılan değeri " + tc .ch );
    }
}

```

Sonuç:

bayt varsayılan değeri 0'dır
 kısa varsayılan değeri 0'dır
 int'in varsayılan değeri 0'dır
 uzun varsayılan değeri 0'dır
 kayan noktanın varsayılan değeri 0,0'dır
 çift varsayılan değeri 0,0'dır
 boolean varsayılan değeri false'tur
 karakterin varsayılan değeri

Tamsayı bölme işleminde yuvarlama nasıl yapılır?

Sonucun kesirli kısmı kesilir. Buna sıfıra yuvarlama denir.

Örnek:

```
halk sınıf Java Örnekleri {

    statik int bir = 5;
    statik int b = 2;

    halk statik void main(String[] args ) {

        Sistem. out .println( "a/b eşittir " +5/2); //Yuvarlama kesirli
        kısım kesilerek yapılır.

    }

}
```

Sonuç: 2

Bir boolean'ı Java'daki bir int değişkeniyle karşılaştırabilir misiniz?

Hayır, derleme hatası alırsınız.

Örnek:

```
halk sınıf Java Örnekleri {

    statik int bir = 5;
    statik boolean b = yanlış ;

    halk statik void main(String[] args ) {

        if ( a == b ) { //Hata: == operatörü int, boolean argüman türleri
        için tanımsız
            Sistem. out .println( "Boolean'ın int ile karşılaştırılması"
        );
        }

    }

}
```

Aşağıdakilerin çıktısı nedir?

```
Sistem . dışarı . println ( 1.0 / 0 );
```

Birçoğumuz ArithmeticException'ı bekleyebilir, ancak bu durumda hiçbir istisna olmayacak, bunun yerine **Infinity yazdıracaktır** .

1.0 çift değişmezdir ve çift veri türü sonsuzluğu destekler.

Örnek:

```
paket soruları;

halk sınıf test sınıfı {

    halk statik void main(String[] args ) {
```

```

        Sistem. out .println( "değer: " +(1.0/0));
    }
}

```

Sonuç:

değer Sonsuzluktur

5. Java Operatörleri - Aritmetik Operatörler ve Atama Operatörleri

Java'daki operatörler:

Java'daki Operatör Türleri:

1. Temel Aritmetik Operatörler
2. Atama Operatörleri
3. Otomatik Arttırma ve Otomatik Azaltma Operatörleri
4. Mantıksal operatörler
5. Karşılaştırma (ilişkisel) operatörler
6. Üçlü operatör

Temel Aritmetik Operatörler

Temel aritmetik operatörler şunlardır: +, -, *, /, %

```
paket FPPaket;
```

```

halk sınıf Aritmetik Operatörler {

    halk statik void main(String[] args ) {
        int bir = 10;
        int b = 9;
        Sistem. out .println( a + b );
        Sistem. out .println( a - b );
        Sistem. out .println( a * b );
        Sistem. out .println(( float ) a / b );
        Sistem. out .println( a % b );
    }
}

```

Atama Operatörleri

Java'daki atama operatörleri şunlardır: =, +=, -=, *=, /=, %=

num2 = num1 değişkene num1 değişkeninin değerini atayacaktır.

sayı2+=sayı1 eşittir **sayı2 = sayı2+sayı1**

sayı2-=sayı1 eşittir **sayı2 = sayı2-sayı1**

sayı2*=sayı1 eşittir **sayı2 = sayı2*sayı1**

sayı2/=sayı1 eşittir **sayı2 = sayı2/sayı1**

num2%=num1 eşittir **num2 = num2%num1**

paket FPPaket;

```
halk sınıf Aritmatik Operatörler {

    halk statik void main(String[] args ) {
        int say11 = 10;
        int say2 = say11 ;
        Sistem. out .println( say12 );
        say12 += say11 ;
        Sistem. out .println( say12 );
        say12 -= say11 ;
        Sistem. out .println( say12 );
        say12 *= say11 ;
        Sistem. out .println( say12 );
        say12 /= say11 ;
        Sistem. out .println( say12 );
        say12 %= say11 ;
        Sistem. out .println( say12 );
    }
}
```

5A. Java Operatörleri - Otomatik Arttırma ve Azaltma Operatörleri ve Mantıksal Operatörler

Java'daki operatörler:

Java'daki Operatör Türleri:

1. Temel Aritmetik Operatörler
2. Atama Operatörleri
3. Otomatik Arttırma ve Otomatik Azaltma Operatörleri
4. Mantıksal operatörler
5. Karşılaştırma (ilişkisel) operatörler
6. Üçlü operatör

Otomatik Arttırma ve Otomatik Azaltma Operatörleri:

++ ve **--**

sayı1++ sayı1 = sayı1+1'dir

sayı1-- sayı1 = sayı1-1'dir

paket FPPaket;

```
halk sınıf Aritmatik Operatörler {

    halk statik void main(String[] args ) {
        int say11 = 10;

        Sistem. out .println( say11 );
        say11 ++;
        Sistem. out .println( say11 );
    }
}
```

```

        say11--;
        Sistem.out.println( say11 );
    }
}

```

Mantıksal operatörler:

Mantıksal operatörler: &&(mantıksal VE), ||(mantıksal VEYA) ve !(Olumsuzlama)

paket FPPaket;

```

halk sınıf Aritmetik Operatörler {

    halk statik void main(String[] args ) {
        int say11 = -10;
        int say12 = -5;
        boolean deneme = yanlış ;
        if ( say11 > 0 && say12 >0) {
            Sistem.out.println( "Her iki değer de 0'dan büyük" );
        }
        başka if ( say11 > 0 || say12 > 0) {
            Sistem.out.println( "Değerlerden biri 0'dan büyük" );
        }
        başka {
            Sistem.out.println( "Her iki değer de 0'dan küçük" );
        }
        eğer ( test ) {
            Sistem.out.println( "Bu doğrudur" );
        }
        eğer (! test ) {
            Sistem.out.println( "Yanlıştır" );
        }
    }
}

```

5B. Java Operatörleri - Karşılaştırma Operatörleri ve Üçlü Operatör

Java'daki operatörler:

Java'daki Operatör Türleri:

1. Temel Aritmetik Operatörler
2. Atama Operatörleri
3. Otomatik Arttırma ve Otomatik Azaltma Operatörleri
4. Mantıksal operatörler
5. **Karşılaştırma (ilişkisel) operatörler**
6. **Üçlü operatör**

Karşılaştırma (ilişkisel) Operatörler:

Java'da altı ilişkisel operatörümüz vardır: ==, !=, >, <, >=, <=

paket FPPaket;

halk sınıf Aritmetik Operatörler {

```

    halk statik void main(String[] args ) {
        int say11 = 10;
        int say12 = 10;

        if ( say11 == say12 ) {
            Sistem. out .println( "Her iki deęer de eřittir" );
        }

        if ( say11 != say12 ) {
            Sistem. out .println( "Deęerler eřit deęil" );
        }

        if ( say11 > say12 ) {
            Sistem. out .println( "Say11, Say12'den büyük" );
        }

        if ( say11 < say12 ) {
            Sistem. out .println( "Say11, Say12'den küçük" );
        }

        if ( say11 >= say12 ) {
            Sistem. out .println( "Say11, Say12'den büyük veya ona eřit"
);
        }

        if ( say11 <= say12 ) {
            Sistem. out .println( "Say11, Say12'den küçük veya ona eřit"
);
        }
    }
}

```

Üçlü operatör:

Bu operatör bir boole ifadesini deęerlendirir ve sonuca göre deęeri atar.

Sözdizimi:

deęişken numarası1 = (ifade) ? doęruysa deęer: yanlıřsa deęer

paket FPPaket;

halk sınıf Aritmetik Operatörler {

```

    halk statik void main(String[] args ) {
        int say11 = 10;
        int say12 = 1;
        int say13 ;
        say3 = ( say11 == say12 ) ? 5 : 6;
        Sistem. out .println( "Say13: " + say13 );
    }
}

```

5C. Java Operatörleri - Soruları

% operatörü nedir?

Modülo veya kalan operatörü olarak adlandırılır. Birinci işlenenin ikinci işlenene bölünmesinden kalan kısmı döndürür.

Örnek:

```
halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {
        int bir = 5;
        int b = 3;

        Sistem. out .println( "% - Hatırlatma: " + a % b );

    }

}
```

Sonuç: Hatırlatıcı 2'dir

++ operatörünün önek ve sonek biçimleri arasındaki fark nedir?

Önek formu, artırma işlemini gerçekleştirir ve artırma işleminin değerini döndürür. Postfix formu, ifadenin tamamının geçerli değerini döndürür ve ardından bu değer üzerinde artırma işlemini gerçekleştirir.

Örnek:

```
halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {
        int bir = 1;
        int B ;

        b = bir ++;

        Sistem. out .println( "b eşittir " + b );
        Sistem. out .println( "a " + a'dır );

        int c = 1;
        int D ;

        d = ++ c ;
        Sistem. out .println( "c eşittir " + c );
        Sistem. out .println( "d eşittir " + d );

    }

}
```

Sonuç:

b 1'dir
a 2'dir
c 2'dir

d 2'dir

Boolean & operatörü ile && operatörü arasındaki fark nedir?

Boolean & operatörünü içeren bir ifade değerlendirilirse, her iki işlenen de değerlendirilir. Daha sonra & operatörü işlenene uygulanır. && operatörünü içeren bir ifade değerlendirilirken ilk işlenen değerlendirilir. İlk işlenen doğru değerini döndürürse ikinci işlenen değerlendirilir. Daha sonra && operatörü birinci ve ikinci işlenenlere uygulanır. İlk işlenen yanlış olarak değerlendirilirse ikinci işlenenin değerlendirmesi atlanır.

Eski:

halk sınıf Java Örnekleri {

```

    halk statik void main(String[] args ) {
        int bir = 2;
        int b = 3;
        int c = 2;
        if ((( bir == b ) & ( bir == c ))) {
            Sistem. out .println( "eşit değerler" );
        }
        başka {
            Sistem. out .println( "değerler eşit değil" );
        }
    }

```

}

Bileşik atama operatörleri nelerdir?

Aritmetik operatörünü atamayla birleştiren operatöre Bileşik Atama operatörü denir.

Örneğin x+=10;

Aşağıdakilerin çıktısı nedir?

1. **int** a , b ;
bir =10;
b =++ bir ; //Sonuç 11
2. **int** a , b ;
bir =10;
b = a++ ; //Sonuç 10
3. **int** bir =20;
bir = ++ bir + 1; //Sonuç 22
4. **int** bir =20;
a = a ++ + 1; //Sonuç 21
5. **int** bir =20;
a = ++ a + ++ a ;//Sonuç 43
6. **int** bir =20;
bir = bir ++ + bir ++; //Sonuç 41
7. **int** bir =20;
bir = bir ++ + ++ bir ; //Sonuç 42
8. **int** a , b ;
bir =30;
b = a --; // Sonuç 39
9. **int** bir =20;

```

    a = -- a - 1; //Sonuç 18
10. int bir =20;
    a = a -- - 1; //Sonuç 19
11. int bir =20;
    a = -- a - -- a ; //Sonuç 1
12. int bir =20;
    a = a -- - a-- ; //Sonuç 1
13. int bir =20;
    a= a-- - --a; //Sonuç 2
14. int bir =20;
    a= a-- - ++a; //Sonuç 40

```

6. Java Yöntemleri - Yöntem Bildirme, Değer Aktarma ve Değer Döndürme

Yöntemler kodu yeniden kullanmamıza izin verir. Örneğin, iki rastgele sayıyı birçok kez ekliyorsanız, her seferinde yazmak yerine, bu sayıyı toplayacak bir yöntem yazabilir ve iki sayıyı toplayanınız gerektiğinde bu yöntemi çağırabilirsiniz.

Bir yöntemin tanımlanması:

Bir yöntemi tanımlamak için belirtmeniz gerekir

Değiştirici – Yöntemin erişim türünü tanımlar

Dönüş türü – Yöntemin döndürdüğü değerin veri türü

Yöntem Adı – Yöntemin adı

Parametre listesi – Giriş parametrelerinin virgülle ayrılmış listesi.

Yöntem gövdesi – Yürütmeniz gereken kod. Parantez içine alınmıştır.

Örnek:

1. Bir yöntemi şu şekilde tanımlayabilirsiniz:

```

public int add(int a, int b){
}

```

Burada public değiştiricidir. Int dönüş türüdür. Add yöntemin adıdır. int a ve int b parametrelerdir.

2. Yani yukarıdaki yöntemde bir tam sayı döndürüyoruz.

Yöntemden hiçbir şey döndürmezsek, bunu şu şekilde tanımlayabiliriz:

```

genel geçersiz eklenti(int a, int b){
}

```

Yukarıda hiçbir şeyi döndürmek istemediğimizden dönüş türü geçersizdir

3. Dönen dize

```
public String bir yöntem(String str1, String str2){
}
```

Burada da dizeleri geçiriyoruz.

4. Bir tamsayı dizisini döndürme

```
public int[] somemethod(int[] arr){
}
```

Yukarıdaki yöntemde, int[] olarak belirtilen bir tamsayı dizisini döndürüyoruz.

5. Bir dizi String döndürme

```
public String[] somemethod(String[] arr){
}
```

Yukarıdaki yöntemde, int[] olarak belirtilen bir dizi diziyi döndürüyoruz ve aynı zamanda bir dizi diziyi de iletiyoruz.

6. Bir nesnenin iletilmesi ve geri verilmesi

```
genel nesne değişikliği(obj o){
}
```

Yukarıdaki yöntemde, yöntemin sınıf adı obj ise, sınıf adı olarak dönüş tipini belirtmiştik, yani o sınıfa ait bir nesneyi döndürüyoruz. Benzer şekilde parametrelerde de bir nesneyi geçiriyoruz.

Bir yöntemi çağırmak:

Bir yöntemi çağırdığınızda, o yöntemin döndürdüğü değeri yakalamanız gerekir.

Örnek:

1. Tamsayıları girdi olarak alan ve bir tamsayı döndüren bir yöntemin çağırılması.

Yöntem:

```
public int add(int a, int b){
}
```

Yöntemi çağırmak:

```
int a = ekle(2, 3);
```

6A. Java Yöntemleri - Tam sayıların iletilmesi ve tam sayıların döndürülmesi

Tamsayı bağımsız değişkenlerini alan ve bir tamsayı döndüren bir Yöntemin çağırılması:

```
paket paket1;
```

```

halk sınıf AddingNumbers {

    halk int ekle ( int a , int B ) {
        int c = a + b ;
        geri dönmek C ;
    }
    halk statik void main(String[] args ) {

        int bir = 2;
        int b = 2;
        AddingNumbers an = new AddingNumbers();

        Sistem. out .println( an . add ( a , b ));

    }
}

```

1. Tamsayıları argüman olarak alan ve hiçbir şey döndürmeyen bir yöntemin çağırılması.

Yöntem:

```

genel geçersiz eklenti(int a, int b){
}

```

Yöntemi çağırmak:

ekle(2, 3)

Örnek:

paket paket1;

```

halk sınıf AddingNumbers {

    halk geçersiz ekleme ( int) a , int B ) {
        int c = a + b ;
        Sistem. out .println( c );
    }
    halk statik void main(String[] args ) {

        int bir = 2;
        int b = 2;
        AddingNumbers an = new AddingNumbers();

        bir .add( a , b );

    }
}

```

6B. Java Yöntemleri - Dizeleri Geçirme ve Dizeleri Geri Döndürme

1. Dizeleri geçirerek bir dize döndüren bir yöntemin çağırılması

Yöntem:


```
public String bir yöntem(String str1, String str2){
}
```

Yöntemi çağırmak:

```
String str3 = somemethod("Subbu", "Selenyum")
```

Örnek:

```
paket paket1;

halk class StringMethodExamples {

    public String birleştirme(String str1 , String str2 ) {
        String str3 = str1 + " " + str2 ;
        geri dönmek str3 ;
    }
    halk statik void main(String[] args ) {
        String str1 = "Subbu" ;
        String str2 = "Selenyum" ;

        StringMethodExamples sme = new StringMethodExamples();
        String str3 = sme .concatenate( str1 , str2 );
        Sistem. out .println( str3 );
    }
}
```

6C. Java Yöntemleri - Dizileri Geçirme ve Dizileri Döndürme

1. Bir diziyi ileterek diziyi döndüren bir yöntemin çağırılması

```
public int[] somemethod(int[] arr){
}
```

Yöntemi çağırmak:

```
int[] arr1 = {1, 2, 3};
```

```
int[] arr2 = bir yöntem(dizi1)
```

Örnek:

```
paket paket1;

halk sınıf İstisnaÖrneği {

    halk statik int [] mth( int [] dizi ) {
        int [] dizi1 = dizi ;
        için ( int ben =0; ben < varış uzunluk ; ben ++ ) {
            dizi1 [ i ] = dizi [ i ]*2;
        }

        geri dönmek dizi1 ;
    }
    halk statik void main(String[] args ) {
```

```

        int [] dizi1 = {3,4,5};
        int [] dizi2 = mth ( dizi1 );
        Sistem. out .println( arr2 [0]);
        Sistem. out .println( arr2 [1]);
        Sistem. out .println( arr2 [2]);
    }
}

```

2. Bir dizi String döndürme

```

public String[] concatenate(String[] arr){
}

```

Çağırma yöntemi:

```
String[] arr1= {"Subbu", "Selenyum"};
```

```
String[] arr2 = birleştirme(arr1);
```

```
paket paket1;
```

```
halk class StringMethodExamples {
```

```

    public String[] concatenate(String[] str ) {
        Dize[] str1 = str ;
        str1 [0] = str [0]+ " öğrenciler" ;
        str1 [1] = str [1]+ " öğrenciler" ;
        geri dönmek str1 ;
    }

```

```

halk statik void main(String[] args ) {
    String[] str = { "Subbu" , "Selenyum" };

```

```

    StringMethodExamples sme = new StringMethodExamples();
    String[] str1 = sme .concatenate( str );
    için ( int ben =0; ben < str1 . uzunluk ; ben ++ ) {
        Sistem. out .println( str1 [ i ]);
    }

```

```
}
```

6D. Java Yöntemleri - Bir nesnenin iletilmesi ve döndürülmesi

1. Bir nesnenin iletilmesi ve geri verilmesi

```

genel nesne değişikliği(obj o){
}

```

Arama Yöntemi:

```
nesne o1 = yeni nesne();
```

```
nesne o2 = değişim(o1);
```

Örnek:

```

paket paket1;

halk sınıf OPCA {
    halk int A ;

    genel OPCA setVariable(OPCA opc ) {
        OPCA opc1 = yeni OPCA();
        opc1 . a = işlem . a *10;
        geri dönmek opc1 ;
    }

    halk statik void main(String[] args ) {

        OPCA opc1 = yeni OPCA();
        opc1 . bir = 2;

        OPCA opc2 = yeni OPCA();
        opc2 = opc2 .setVariable( opc1 );
        Sistem. out .println ( opc2.a ) ;
    }
}

```

6E. Java Yöntemleri - Soruları

İlkel değişkenler yöntemlere nasıl aktarıldı - değere göre mi yoksa referansa göre mi?

Java'da ilkel değişkenler değerlere göre yöntemlere aktarılır. Geçirilen değer yöntemde değişirse orijinal değer değişmez.

Örnek:

```

paket soruları;

halk sınıf test sınıfı {
    halk statik void main(String[] args )
{
    int x = 5;
    değişiklik ( x );
    Sistem. out .println( x );
}

    halk statik geçersiz değişiklik ( int X )
{
    x = 10;
}
}

```

Sonuç: 5

Bir nesneyi Java'daki bir yönteme aktardığımızda, değere veya referansa göre mi geçiyor?

Java referansların bir kopyasını oluşturur ve bunu yönteme iletir, ancak yine de aynı bellek referansına işaret ederler. Yani, yöntemin içinden geçen referansa başka bir nesne ayarlanırsa, yöntemi çağıran nesne ve onun referansı etkilenmeden kalacaktır.

Nesnenin kendisini başka bir konuma veya nesneye atıfta bulunacak şekilde değiştirirsek değişiklikler geri yansıtılmaz

paket soruları;

```
//Referansların da iletildiğini gösteren bir Java programı
//değere göre.
sınıf Testi
{
    int X ;
    Test( int ben ) { x = ben ; }
    Test() { x = 0; }
}

sınıf test sınıfı
{
    halk statik void main(String[] args )
    {
        // t bir referanstır
        Test t = yeni Test(5);

        // Referans iletilir ve referansın bir kopyası
        // change()'da yaratıldı
        değişiklik ( t );

        // tx'in eski değeri yazdırılıyor
        Sistem. out .println ( t.x ) ;
    }
    halk statik geçersiz değişiklik (Test t )
    {
        // Referansı başka bir konuma yönlendirecek şekilde değiştirdik
        // artık referansta yapılan değişiklikler yansıtılmıyor
        // ana ekrana geri dön
        t = yeni Test();

        T . x = 10;
    }
}
```

7. Java Temelleri - Yöntemin Aşırı Yüklenmesi

Aynı adda ancak farklı argümanlara, farklı argüman türlerine veya farklı argüman sırasına sahip birden fazla yönteminiz varsa buna yöntem aşırı yükleme adı verilir.

Yöntem Aşırı Yüklemesinin üç yolu:

1. Farklı hayır. argümanların
Örn: ayrıntılar(int yaş, Dize adı)
ayrıntılar(int yaş, int sno, Dize adı)
2. Farklı argüman türleri
Örn: ayrıntılar(int a, int b)
ayrıntılar(Dize a, Dize b)
3. Farklı sıra
Örn: ayrıntılar(int yaş, Dize adı)
Ayrıntılar(Dize adı, int yaşı)

Örnek:

Yöntem Aşırı Yüklemesi – Farklı Argümanlar

```
paket FPPaket;

sınıf Demosu {
    halk statik geçersiz ayrıntılar ( int yaş , Dize adı ) {
        Sistem. out .println( "Yaş: " + yaş + " İsim: " + isim );
    }
    halk statik geçersiz ayrıntılar ( int yaş sno , Dize adı ) {
        Sistem. out .println( "Yaş " + yaş + " S.No " + sno + " İsim " +
isim );
    }
    halk statik void main(String[] args ) {
        ayrıntılar (20, "Venkat" );
        ayrıntılar (23, 123, "Ganesh" );
    }
}
```

Yöntem Aşırı Yüklemesi – Farklı Bağımsız Değişken Türleri

```
paket FPPaket;

sınıf Demosu {
    halk statik geçersiz ayrıntılar ( int a , int B ) {
        Sistem. out .println( "a, " + a + " b, " + b'dir );
    }
    halk statik geçersiz ayrıntılar (Dize a , Dize b ) {
        Sistem. out .println( "a, " + a + " b, " + b'dir );
    }
    halk statik void main(String[] args ) {
        ayrıntılar (20, 30);
        ayrıntılar ( "Venkat" , "Ganesh" );
    }
}
```

Yöntem Aşırı Yüklemesi – Farklı Sıra

```
paket FPPaket;

sınıf Demosu {
    halk statik geçersiz ayrıntılar ( int yaş , Dize adı ) {
        Sistem. out .println( "Yaş: " + yaş + " İsim: " + isim );
    }
    halk statik geçersiz ayrıntılar (Dize adı , int yaş ) {
        Sistem. out .println( "Yaş: " + yaş + " İsim: " + isim );
    }
    halk statik void main(String[] args ) {
        ayrıntılar (20, "Venkat" );
        ayrıntılar ( "Ganesh" , 22);
    }
}
```

7A. Java Temelleri - Yöntem Aşırı Yüklemesi - Soruları

Java'da yöntem aşırı yüklemesi nedir?

Aynı adda ancak farklı argümanlara sahip iki veya daha fazla yöntemle sahip bir sınıf, bu yöntemlerin aşırı yüklendiğini söyleriz. Java'da yöntem aşırı yüklemesi kullanılarak statik polimorfizm elde edilir. Yöntemlerin benzer görevleri ancak farklı girdiler veya değerlerle gerçekleştirmesini istediğimizde yöntem aşırı yüklemesi kullanılır. Aşırı yüklenmiş bir yöntem çağrıldığında Java ilk olarak yöntem adını, argüman sayısını ve argüman tipini kontrol eder; bu derleyiciye dayanarak bu yöntemi yürütür. Derleyici, derleme zamanında hangi yöntemin çağrılacağına karar verir. Java'da aşırı yükleme kullanılarak statik polimorfizm veya statik bağlama elde edilebilir.

Not: Dönüş türü, yöntem imzasının bir parçası değildir. Farklı dönüş türlerine sahip yöntemlerimiz olabilir, ancak dönüş türü tek başına Java'da bir yöntemi çağırarak için yeterli değildir.

Bir main() yöntemi aşırı yüklenebilir mi?

Evet. Sınıfta farklı yöntem imzasına ve uygulamaya sahip istediğiniz sayıda `main()` yönteminiz olabilir .

Java neden operatörün aşırı yüklenmesini desteklemiyor?

Operatörün aşırı yüklenmesi, kodun okunmasını ve bakımını çok zorlaştırır. Kodun basitliğini korumak için Java, operatörün aşırı yüklenmesini desteklemez.

Yöntemin aşırı yüklenmesine hangi kısıtlamalar getirildi?

İki yöntem aynı ada ve bağımsız değişken listesine sahip olmayabilir ancak farklı dönüş türlerine sahip olabilir.

Bir yöntem, farklı dönüş türüne ancak aynı argüman türüne dayalı olarak aşırı yüklenebilir mi?

Hayır, çünkü yöntemler dönüş türleri kullanılmadan çağrılabilir, bu durumda derleyici için belirsizlik söz konusudur.

8. Java Oluşturucuları - Oluşturucu Kullanımları

Yapıcı, yeni oluşturulan bir nesnenin değişkenlerini başlatmak için kullanılan bir kod bloğudur. Bir yapıcı, Java'daki bir örnek yöntemle benzer, ancak bir dönüş türüne sahip olmadığı için bir yöntem değildir.

Yapıcı sınıfla aynı isimle bildirilmelidir.

Örnek:

```
paket YapıcıDemo;

halk sınıf YapıcıDemo1 {
    int yaş ;
    Dize adı ;
    YapıcıDemo1(){
        Bu . yaş = 10;
        Bu . isim = "Subbu" ;
    }
    YapıcıDemo1( int yaş , Dize adı ){
        Bu . yaş = yaş ;
        Bu . isim = isim ;
    }
}
```

```

    }
    halk statik void main(String[] args ) {
        ConstructorDemo1 cd1 = new ConstructorDemo1();
        Sistem. out .println( "Yaş: " + cd1 . yaş );
        Sistem. out .println( "Ad: " + cd1 . name );

        ConstructorDemo1 cd2 = new ConstructorDemo1(15, "Venkat" );
        Sistem. out .println( "Yaş: " + cd2 . yaş );
        Sistem. out .println( "İsim " + cd2.isim );
    }
}

```

8A. Java Oluşturucuları - Soruları

Bir kurucu tanımlayın?

Yapıcı, bir nesnenin durumunu başlatmak için kullanılan bir yöntemdir ve nesne yaratıldığı sırada çağrılır. Yapıcı için kurallar şunlardır:

Yapıcı Adı sınıf adıyla aynı olmalıdır.

Bir yapıcının dönüş tipi olmamalıdır.

Derleyici ne zaman bir sınıf için varsayılan kurucu sağlar?

Başka hiçbir kurucu sağlanmadıysa, derleyici bir sınıf için varsayılan bir kurucu sağlar.

Eğer sınıfta parametrelili bir kurucum varsa, derleyici varsayılan bir kurucu oluşturacak mı?

Hayır, sınıfta parametrelili kurucu varsa derleyici varsayılan kurucuyu yaratmaz. Örneğin, yapıcısı olmayan bir sınıfım varsa, derleyici varsayılan kurucuyu yaratacaktır. Örneğin:

```

genel sınıf Araba {
}

```

Yukarıdaki Araba sınıfında hiçbir yapıcı yoktur, dolayısıyla derleyici varsayılan bir kurucu oluşturur.

```

genel sınıf Araba {
    Araba(Dize adı) {
    }
}

```

Bu örnekte derleyici herhangi bir varsayılan kurucu oluşturmayacak çünkü Car sınıfında zaten bir kurucu var.

Java'daki sınıf adıyla aynı bir yöntem adına sahip olabilir miyiz?

Evet. sınıf adıyla aynı yöntem adını alabiliriz, herhangi bir derleme hatası vermez ancak yöntem adının sınıf adıyla aynı olduğuna dair bir uyarı mesajı gösterir.

Java'daki yapıcıları geçersiz kılabilir miyiz?

Java'da yalnızca yöntemler geçersiz kılınabilir. Yapıcılar Java'da miras alınamaz. Bu nedenle Java'da yapıcıları geçersiz kılmanın bir anlamı yoktur.

Bir kurucuyu diğer kurucudan nasıl çağırırım?

- Aynı sınıf içinde: Aynı sınıftaki yapıcılar için `this()` anahtar sözcüğü kullanılarak yapılabilir.
- Temel sınıftan: temel sınıftan yapıcıyı çağırarak için `super()` anahtar sözcüğünü kullanarak.

Buna yapıcı zincirleme de denir.

Örnek:

```
paket soruları;

//Yapıcı Zincirlemeyi gösteren Java programı
//aynı sınıf içinde This() anahtar sözcüğünü kullanarak
sınıf test sınıfı
{
    // varsayılan kurucu 1
    // varsayılan kurucu başka bir kurucuyu çağırarak
    // aynı sınıftan bu anahtar kelimeyi kullanıyoruz
    test sınıfı()
    {
        // yapıcı 2'yi çağırır
        bu (5);
        Sistem. out .println( "Varsayılan kurucu" );
    }

    // parametrelili yapıcı 2
    test sınıfı( int x )
    {
        // yapıcı 3'ü çağırır
        bu (5, 15);
        Sistem. out .println( x );
    }

    // parametrelili yapıcı 3
    test sınıfı( int x , dahili y )
    {
        Sistem. out .println( x * y );
    }

    halk statik void main(Dize bağımsız değişkenleri [])
    {
        // ilk önce varsayılan kurucuyu çağırır
        yeni test sınıfı ();
    }
}
```

Ebeveyn sınıfı yapıcısına erişmek istiyorsak süper yöntemini kullanın.

```
paket soruları;

sınıf temel sınıf{
    Dize adi ;
    temel sınıf(Dize adi ){
        Bu . isim = isim ;
        Sistem. out .println( isim );
    }
}
sınıf test sınıfı temel sınıfı genişletir
{
    testsınıfı(){
```



```

        this ( "yapıcı çağırıyor" );
        Sistem. out .println( "Argüman oluşturucu yok" );
    }
    testsınıfı(Dize dizisi ){
        süper ( str );
    }
    halk statik void main(String[] args ) {
        testsınıfı t = yeni testsınıfı();
    }
}

```

Bu anahtar kelimeyi kullanarak başka bir kurucudan bir kurucu çağırıyorsak, anahtar kelimenin ilk önce olması gerekir. Aksi takdirde hata verir.

paket soruları;

```

sınıf test sınıfı
{
    testsınıfı(){
        Sistem. out .println( "Argüman oluşturucu yok" );
        this ( "yapıcı çağırıyor" );
    }
    testsınıfı(Dize dizisi ){
        Sistem. out .println( str );
    }
    halk statik void main(String[] args ) {
        testsınıfı t = yeni testsınıfı();
    }
}

```

Yukarıdaki, "Oluşturucu çağırısının bir yapıcıdaki ilk ifade olması gerekir" şeklinde bir hata veriyor.

9. Java Temelleri - Özellikler Sınıfı

Özellikler sınıfı nesnesi "Sistem", yapılandırma için bir dizi sistem özelliğini korur. Kendi sistem özelliğinizi de tanımlayabilirsiniz.

Java'nın kendisi Properties nesnesini kullanarak kendi yapılandırmasını korur.

Bir sistem özelliğini alma:

```
System.getProperty("java.class.path");
```

Tüm sistem özelliklerini alma:

```
System.getProperties();
```

Kendi sistem özelliğinizi ayarlama:

```
System.setProperty("TestKey","Merhaba");
```

Örnek:

```
paket FPPaket;
```

```

halk sınıf ÖzelliklerTest {

    halk statik void main(String[] args ) {
        Sistem. out .println(System.getProperties ( ));
        Sistem. out .println(System.getProperty ( " java.class.path" ));
        Sistem. setProperty ( "TestKey" , "Merhaba" );
        Sistem. out .println(System.getProperty ( " TestKey" ));
    }
}

```

10. Erişim Değiştiricileri varsayılan, genel, korumalı ve Özel

erişim değiştiricileri bir sınıfın, yöntemin, değişkenin veya kurucunun görünürlüğünü kontrol etmek için kullanılır. Java'da 4 farklı erişim değiştirici bulunmaktadır. Bunlar; varsayılan, genel, özel ve korumalıdır.

varsayılan: varsayılan üyeler veya erişim değiştiricisi olmayan üyeler paket içinde görünür. Ve yalnızca aynı pakette bulunan alt sınıflara miras alınırlar. Bu, miras alınmadıkları ve paketin dışında görünmedikleri anlamına gelir.

public: public üyeler her yerde görülebilir ve herhangi bir alt sınıfa miras alınırlar.

özel: Bir sınıfın özel üyelerine, ister alan, ister yöntem, ister kurucu olsun, tanımlandıkları sınıfın dışından erişilemez. özel üyeler de alt sınıfa miras alınmaz.

korumalı: korunan üyeler, genel üyelerin yarısına ve varsayılan üyelerin yarısına sahiptir; yani korunan üyeler, varsayılan üyeler gibi paket içinde görünür ve genel üyeler gibi herhangi bir alt sınıfa miras alınabilirler.

Sınıf – Genel ve varsayılan sınıf için erişim değiştiricileri. Özel ve Korumalı sınıflar için kullanılamaz.

Yöntemler – Yöntem için erişim değiştiricileri varsayılan, genel, özel ve korumalıdır.

Değişkenler – Değişkenler için erişim değiştiriciler varsayılan, genel, özel ve korumalıdır.

İki paket oluşturun: FPPackage ve SPPackage

FPPackage altında FirstProgram.java ve SecondProgramInFPP.java olmak üzere iki sınıf oluşturun

FirstProgram.java:

```

paket FPPaket;

halk sınıf İlkProgram {

    halk statik void main(String[] args ) {

        Sistem. out .println( "Merhaba Dünya!" );

    }

    Ed'i koru int toplam( int a , int B ) {
        int c = a + b ;
        geri dönmek C ;
    }
}

```

```

    }
}

SecondProgramInFPP.java

paket FPPaket;

halk sınıf SecondProgramInFPP {

    halk statik void main(String[] args ) {
        Sistem. out .println( "İlk Program aynı pakette çağrılıyor" );
        İlkProgram fp = yeni İlkProgram();
        int bir = 5;
        int b = 7;
        int C ;
        c = fp .sum( a , b );
        Sistem. out .println( "Toplam " + c );
    }

}

```

SPPackage altında SecondProgram.java sınıfını oluşturun

```

paketi ;

içe aktarın ;

halk class SecondProgram FirstProgram'ı genişletir {
    halk statik void main(String[] args ) {
        Sistem. out .println( "FPPackage'da FirstProgramı Çağırarak" );
        İlkProgram fp = yeni İlkProgram();
        int c = fp . toplam (5, 5);
        Sistem. out .println( "Toplam " + c );
    }
}

```

Şimdi erişim türlerini değiştirin ve kontrol edin.

10 A. Erişim Değiştiriciler - Soruları

Sınıf için hangi erişim değiştiriciler kullanılabilir?

Public ve default sınıfı için yalnızca iki erişim değiştirici kullanabiliriz.

public: Public değiştiriciye sahip bir sınıf görülebilir

- 1) Aynı sınıfta
- 2) Aynı paket alt sınıfında
- 3) Aynı pakette alt sınıf olmayan
- 4) Farklı paket alt sınıfında
- 5) Alt sınıf olmayan farklı paketlerde.

default: Varsayılan değiştiriciye sahip bir sınıfa erişilebilir

- 1) Aynı sınıfta
- 2) Aynı paket alt sınıfında
- 3) Aynı pakette alt sınıf olmayan
- 4) Farklı paket alt sınıfında
- 5) Alt sınıf olmayan farklı paketlerde.

Yöntemler için hangi erişim değiştiricilerin kullanılabileceğini açıklayın?

Yöntemler için genel, özel, korumalı ve varsayılan tüm erişim değiştiricileri kullanabiliriz.

public: Bir metod public olarak bildirildiğinde ona erişilebilir.

- 1) Aynı sınıfta
- 2) Aynı paket alt sınıfında
- 3) Aynı pakette alt sınıf olmayan
- 4) Farklı paket alt sınıfında
- 5) Alt sınıf olmayan farklı paketlerde.

default: Bir yöntem varsayılan olarak bildirildiğinde, o yönteme erişebiliriz.

- 1) Aynı sınıfta
- 2) Aynı paket alt sınıfında
- 3) Aynı pakette alt sınıf olmayan

Varsayılan erişim yöntemine erişemiyoruz

- 1) Farklı paket alt sınıfı
- 2) Alt sınıf olmayan farklı paketler.

korumalı:

Bir yöntemin korumalı olduğu bildirildiğinde ona erişilebilir.

- 1) Aynı sınıfta olanlar
 - 2) Aynı paket alt sınıfında
 - 3) Aynı pakette alt sınıf olmayanlar ile
 - 4) Farklı paket alt sınıflarında
- Farklı paketlerdeki alt sınıf olmayanlara erişilemez.

özel: Bir yöntem özel olarak bildirildiğinde, ona yalnızca o sınıftan erişilebilir.

İçinde erişilemiyor

- 1) Aynı paket alt sınıfı
- 2) Alt sınıf olmayan aynı paket
- 3) Farklı paket alt sınıfı
- 4) Alt sınıf olmayan farklı paketler.

Üst sınıf için hangi erişim değiştiricilere izin verilir?

Üst düzey sınıf için yalnızca iki erişim değiştiriciye izin verilir.

genel ve varsayılan.

Bir sınıf public olarak bildirilirse her yerde görünür. Bir sınıf varsayılan olarak bildirilirse yalnızca aynı pakette görünür.

Eğer sınıfa erişim değiştirici olarak özel ve korumalı vermeye çalışırsak aşağıdaki derleme hatasını alırız.

Yalnızca public, abstract ve final sınıfı için Yasadışı Değiştiriciye izin verilir.

paket soruları;

```
sınıf testi{ //Hata: Genel tür dosyasında tanımlanmalıdır
    halk statik geçersiz ana() {
        Sistem. out .println( "merhaba" );
    }
}
```

```

    }
}
özel sınıf test_sınıfı //Hata
{
    halk statik void main(String[] args ) {
        Sistem. out .println( "merhaba" );
        Ölçek. ana ();
    }
}

```

Değişkenler için hangi erişim değiştiricilerin kullanılabileceğini açıklayın?

Değişkenler için genel, özel, korumalı ve varsayılan tüm erişim değiştiricileri kullanabiliriz.

public: Bir değişken public olarak bildirildiğinde ona erişilebilir.

- 1) Aynı sınıfta
- 2) Aynı paket alt sınıfında
- 3) Aynı pakette alt sınıf olmayanlar
- 4) Farklı paket alt sınıfında
- 5) Alt sınıf olmayan farklı paketlerde.

default: Bir değişken varsayılan olarak bildirildiğinde, o yönetime erişebiliriz.

- 1) Aynı sınıfta
- 2) Aynı paket alt sınıfında
- 3) Aynı pakette alt sınıf olmayan

Varsayılan erişim değişkenlerine erişemiyoruz

- 4) Farklı paket alt sınıfı
- 5) Alt sınıf olmayan farklı paketler.

korumalı: Bir değişken korumalı olarak bildirildiğinde ona erişilebilir

- 1) Aynı sınıfta olanlar
- 2) Aynı paket alt sınıfında
- 3) Aynı pakette alt sınıf olmayan
- 4) Farklı paket alt sınıflarında

Farklı paketlerdeki alt sınıf olmayanlara erişilemez

özel: Bir değişken özel olarak bildirildiğinde, ona yalnızca o sınıftan erişilebilir.

İçinde erişilemiyor

- 1) Aynı paket alt sınıfı
- 2) Alt sınıf olmayan aynı paket
- 3) Farklı paket alt sınıfı
- 4) Alt sınıf olmayan farklı paketler.

Özel olarak bildirilen bir sınıfa paketinin dışından erişilebilir mi?

Mümkün değil.

Bir sınıf korumalı olarak ilan edilebilir mi?

Korumalı erişim değiştiricisi sınıfa ve arayüzlere uygulanamaz. Yöntemler, alanlar `korumalı` olarak bildirilebilir , ancak bir arayüzdeki yöntemler ve alanlar `korumalı` olarak bildirilemez .

Korumalı bir yöntemin erişim kapsamı nedir?

`Korumalı` bir yönteme aynı paketteki sınıflar veya herhangi bir paketteki sınıfın alt sınıfları tarafından erişilebilir.

Bir değişken özel olarak bildirilirse değişkene nereden erişilebilir?

Özel bir değişkene yalnızca bildirildiği sınıf içinden erişilebilir.

Özel, korumalı ve kamusal derken ne anlıyorsunuz?

Bunlar erişilebilirlik değiştiricileridir. `Özel` en kısıtlayıcıdır, `kamu ise` en az kısıtlayıcıdır. Aynı paket bağlamında `korumalı ve varsayılan tür` (ayrıca `korumalı paket olarak da bilinir`) arasında gerçek bir fark yoktur , ancak korumalı anahtar kelime, farklı bir paketteki türetilmiş bir sınıfın görünürlüğüne izin verir.

Dış sınıfın üyesi olan bir iç sınıfla hangi değiştiriciler kullanılabilir?

(Yerel olmayan) bir iç sınıf, genel, korumalı, özel, statik, nihai veya soyut olarak bildirilebilir.

Bir sınıf herhangi bir erişim değiştiricisi olmadan bildirilirse, sınıfa nereden erişilebilir?

Herhangi bir erişim değiştiricisi olmadan bildirilen bir sınıfın paket erişimine sahip olduğu söylenir. Bu, sınıfa yalnızca aynı pakette tanımlanan diğer sınıflar ve arayüzler tarafından erişilebileceği anlamına gelir.

Bir yöntemin korumalı olduğu bildirilirse, yönteme nereden erişilebilir?

Korumalı bir yönteme yalnızca aynı paketin sınıfları veya arayüzleri veya bildirildiği sınıfın alt sınıfları tarafından erişilebilir.

Herkese açık ve halka açık olmayan bir sınıf arasındaki fark nedir?

Bir ortak sınıfa paketinin dışından da erişilebilir. Herkese açık olmayan bir sınıfa, paketinin dışından erişilemez.

11. Java Erişimsiz Değiştiriciler - Statik değişkenler, yöntemler, bloklar ve sınıflar

Java, görünürlük dışındaki işlevleri sağlamak için başka değiştiriciler sağlar. Bu değiştiricilere **Erişim Dışı Değiştiriciler** adı verilir . Java'da birçok erişim dışı değiştirici vardır. Her değiştiricinin kendi işlevi vardır. En çok kullanılan erişim dışı değiştiricilerden bazıları aşağıda listelenmiştir.

static : Statik olarak bildirilen üyeler, bir sınıfın tüm örnekleri için ortaktır. Statik üyeler, sınıf belleğinde saklanan sınıf düzeyindeki üyelerdir.

final : Bu değiştirici, bir değişkenin, yöntemin veya sınıfın daha fazla değiştirilmesini kısıtlamak için kullanılır. Final olarak bildirilen bir değişkenin değeri, değer aldıktan sonra değiştirilemez. Bir son yöntem, alt sınıfta geçersiz kılınamaz ve bir son sınıfa alt sınıf oluşturamazsınız. Java'daki final anahtar kelimesi hakkında daha fazla bilgi için bu yazıya bakın .

abstract : Bu değiştirici bir sınıfla veya bir yöntemle kullanılabilir. Bu değiştiriciyi değişkene ve yapıcıya uygulayamazsınız. Özet olarak bildirilen bir yöntemin alt sınıfta değiştirilmesi gerekir. Soyut olarak bildirilen bir sınıfı başlatamazsınız.

Statik:

Statik anahtar kelime sınıf, değişken, yöntem ve blokla birlikte kullanılabilir. Statik üyeler belirli bir örnek yerine sınıfa aittir; bu, bir üyeyi statik yaparsanız, ona aynı sınıf/çocuk sınıfta nesne olmadan ve farklı bir sınıfta sınıf adıyla erişebileceğiniz anlamına gelir.

Statik Değişkenler:

```
paket Statik Paket;
```

```
halk sınıf Statik1 {

    int sayım = 0;
    halk geçersiz sayım() {
        say ++;
    }
    halk int dönüşsayısı() {
        geri dönmek saymak ;
    }
    halk statik void main(String[] args ) {
        Sistem. out .println( "Merhaba" );
        Statik1 s1 = yeni Statik1();
        Statik1 s2 = yeni Statik1();
        s1 .count();
        int c = s1 .returncount();
        int d = s2 .returncount();
        Sistem. out .println( "Sayma değeri " + c );
        Sistem. out .println( "Sayma değeri " + d );
    }
}
```

Değişkenin önündeki statik anahtar kelimeyi kaldırın ve sonuçları gözlemleyin.

Statik anahtar kelimeyi final olarak değiştirin ve sonuçları gözlemleyin.

Statik Yöntemler:

Herhangi bir nesne olmadan statik bir yöntem çağırabiliriz çünkü bir üyeyi statik hale getirdiğimizde sınıf düzeyinde olur. Statik anahtar kelimeyi kaldırır ve onu statik olmayan hale getirirsek, onu çağırarak için sınıftan bir nesne oluşturmamız gerekir.

```
paket FPPaket;
```

```
sınıf StaticMethod
{
    // Bu statik bir yöntemdir
    // myMethod()' u geçersiz kıl
    statik void myMethod()
    {
```

```

Sistem. out .println( "myMethod" );
}

halk statik void main(String[] args )
{
    /* Bunu çağırdığımızı görebilirsiniz
    * herhangi bir nesne oluşturmada yöntem.
    */
    //StaticMethod s = new StaticMethod();
    // s.myMethod ();
    myMethod();
}
}

```

Yukarıdaki örnekte “myMethod()” metoduna erişim sağlayacak bir nesne oluşturduk. Bu metodu statik olarak bildirirsek bir nesne yaratmamıza gerek kalmaz.

Statik Blok:

Statik blok, statik değişkenleri başlatmak için kullanılır. Bu blok, sınıf belleğe yüklendiğinde yürütülür. Bir sınıf, programa yazıldıkları sırayla yürütülecek birden fazla Statik bloğa sahip olabilir.

paket FPPaket;

```

halk sınıf StaticBlock {

    statik int sayı ;
    statik Dize metni ;
    statik {
        Sistem. out .println( "Blok 1'de değişkenler başlatılıyor" );
        sayı = 10;
        metin = "merhaba" ;
    }
    statik {
        Sistem. out .println( "Blok 2'de değişkenler başlatılıyor" );
        sayı = 11;
        metin = "merhaba2" ;
    }
    halk statik void main(String[] args ) {
        Sistem. out .println( "Sayının değeri " + sayı );
        Sistem. out .println( "Metnin değeri " + metin );
    }
}

```

Statik Sınıf:

Bir sınıf ancak iç içe geçmiş bir sınıfsa statik hale getirilebilir.

İç içe statik sınıfın Dış sınıfın referansına ihtiyacı yoktur

Statik bir sınıf, Outer sınıfının statik olmayan üyelerine erişemez.

paket FPPaket;

```

sınıf StatikSınıf{
    özel statik String str = "Statik Sınıf Örneği" ;

    //Statik sınıf
    statik sınıf MyNestedClass{
        //statik olmayan yöntem
    }
}

```



```

    halk geçersiz disp() {

        /* Dış sınıfın str değişkenini yaparsanız
        * statik değilse derleme hatası alırsınız
        * çünkü: iç içe geçmiş bir statik sınıf, olmayanlara erişemez
        * dış sınıfın statik üyeleri.
        */
        Sistem. out .println( str );
    }

}

halk statik void main(Dize bağımsız değişkenleri [])
{
    /* İç içe geçmiş sınıfın örneğini oluşturmak için dış sınıfa
    ihtiyacımız yoktu
    * sınıf örneği ancak normal iç içe geçmiş bir sınıf için ihtiyacınız
    olacak
    * öncelikle dış sınıfın bir örneğini oluşturmak için
    */
    StaticClass.MyNestedClass obj = new StaticClass.MyNestedClass();
    obj .disp();
}
}

```

11A. Java Erişimsiz Değiştiriciler - Statik Değişkenler, Yöntemler, Bloklar ve Sınıflar - Soruları - Bölüm 1

Java'da statik anahtar kelime nedir ?

Statik, Erişim Dışı Bir Değiştiricidir. Statik bir şeye sahip olduğumuzda, bu onun nesneye değil sınıfa ait olduğu anlamına gelir. Statik değişkene, yönteme, iç içe geçmiş sınıfa ve başlatma bloklarına (statik blok) uygulanabilir.

Statik değişken nedir?

Statik değişkenler sınıfa aittir ve statik değişkenler bir sınıfın tüm örnekleri tarafından paylaşılır.

Statik bir değişken , sınıf yükleme sırasında belleğin yalnızca bir kez tahsis edilmesini sağlar.

sınıf için örnek oluşturmaya gerek kalmadan "<<SınıfAdı>>.<<VariableName>>" çağrılarak doğrudan erişilebilir .

Statik bir değişkenin değeri tüm örnekler için ortak olacaktır

genel sınıf StaticVariableExample

```

{
    statik int a =10;

    public static void main(String args[]){

        StaticVariableExample s1 = new StaticVariableExample();
        StaticVariableExample s2 = new StaticVariableExample();
        System.out.println("s1.a değeri :"+s1.a);
    }
}

```

```

System.out.println("s2.a değeri :"+s2.a);
//s1'i tek başına bir değerle değiştir
s1.a=20;
System.out.println("s1.a değeri :"+s1.a);
System.out.println("s2.a değeri :"+s2.a);
}
}

```

Çıktı şu şekilde olacaktır

: s1.a değeri :10s2.a değeri :10s1.a değeri :20s2.a değeri :20

Yerel değişkenler statik olarak atanamaz; sınıf yüklemesi sırasında bellek atanmadığı için derleme zamanı hatası "**ifadenin geçersiz başlangıcı**" olur.

Java'da statik üyelere nasıl erişiriz?

Örnek değişkenlere ve örnek yöntemlere referans değişkeni kullanılarak erişilebilir. Ancak statik değişkenlere veya statik yöntemlere erişmek için Java'da Sınıf adını kullanırız.

Statik yöntem nedir?

Statik bir yöntem nesneden ziyade sınıfa aittir . "<<SınıfAdı>>.<<YöntemAdı>>" sınıf adı kullanılarak doğrudan çağrılabilir.

statik değişkenlere doğrudan erişebilir ve statik olmayan değişkenlere erişemez ve yalnızca statik bir yöntemi doğrudan çağırabilir ve ondan statik olmayan bir yöntemi çağırılmaz .

Yalnızca statik olan main() yöntemi JVM tarafından otomatik olarak çağrılacaktır. Statik yöntemin tamamı otomatik olarak çağrılmayacaktır.

Statik yöntemler Java'daki örnek değişkenlere erişebilir mi?

Hayır. Statik yöntemlerde örnek değişkenlere erişilemez. Örnek değişkene statik yöntemle erişmeye çalıştığımızda derleme hatası alıyoruz. Hata aşağıdaki gibidir:

"Statik olmayan alan adına statik bir referans yapılamaz"

Statik yöntemler örnek değişkenlere doğrudan erişemese de, örnek işleyiciyi kullanarak bunlara erişebilirler.

Statik değişkenler belleğe ne zaman yüklenir?

İlgili Sınıf yüklendiğinde çalışma zamanında yüklenirler.

Main() yöntemi olmadan statik bir blok var olabilir mi?

Evet . Ana yöntem olmadan sınıfta tek başına statik bloğunuz olabilir .

Java'da statik yöntemleri aşırı yükleyebilir miyiz ?

Evet , Java'da statik bir yöntemi aşırı yükleyebilirsiniz.

Java'daki statik yöntemleri geçersiz kılabilir miyiz?

Hayır , Java'da herhangi bir Çalışma Zamanı olmayacağından statik bir yöntemi geçersiz kılamazsınız. Polimorfizm oluyor.

Neden main() yöntemi statik olarak bildiriliyor?

`main()` yöntemi, sınıfın başlatılmasından önce bile JVM tarafından çağrılır, dolayısıyla `static` olarak bildirilir .

Eğer `main()` yöntemimiz statik olarak bildirilmemişse, JVM'nin önce nesneyi yaratması ve çağrı yapması gerekir, bu da fazladan bellek tahsisi sorununa neden olur.

Statik blok nedir?

Statik blok , bir sınıf JVM'ye ilk kez yüklendiğinde çalıştırılacak olan Java sınıfı içindeki bir kod bloğudur. Değişkenleri başlatmak için çoğunlukla statik blok kullanılacaktır.

Statik blok yükleme sırasında yalnızca bir kez çağrılacaktır ve **herhangi bir dönüş türüne** veya herhangi bir anahtar kelimeye (**this** veya **super**) sahip olamaz.

sınıf testi

```
{
    int değer;
    statik{
değer = 100;
}
}
```

Kodumuzda birden fazla statik blok bulunabilir mi?

Evet kodumuzda birden fazla statik blok bulunabilir. Yazıldığı sırayla yürütülecektir.

Statik sınıf nedir?

Java'da yalnızca iç içe geçmiş sınıfların `static` olarak bildirilmesine izin verilir , üst düzey bir sınıf statik olarak bildirilemez.

Statik sınıflar bir sınıfın içinde yuvalanmış olsalar da, dış sınıfın referansına ihtiyaç duymazlar, yalnızca dış sınıf gibi davranırlar.

Yapıcılar Java'da statik olabilir mi?

Genel olarak statik bir yöntem, "Yöntemin belirli bir nesneye değil sınıfa ait olduğu" anlamına gelir, ancak bir yapıcı her zaman bir nesneye göre çağrılır, dolayısıyla bir yapıcının **statik olmasının bir anlamı yoktur** .

Neden soyut yöntem Java'da statik olamaz?

Soyut bir sınıfta somut bir yönteminiz olduğunda bu yöntemin statik olabileceğini varsayalım. Diyelim ki aşağıdaki gibi bir sınıfımız var

```
genel sınıf ÖzetTest
{
    statik geçersiz disp ()
{
    System.out.println("statik yöntemin gösterimi");
```

```
}
}
```

Daha sonra disp()' a "AbstractTest.disp()" aracılığıyla erişilebilir . Ancak aynı nedenden dolayı, bir static bildirdiğinizde uygulanamaz. soyut olma yöntemi . Statik yöntem doğrudan çağrılabilirdiğinden, onu soyut yapmak, işe yaramayan tanımsız bir yöntemin çağrılmasını mümkün kılacaktır, dolayısıyla buna izin verilmez.

Java'daki Arayüzde statik yöntemler bulunabilir mi ?

Hayır, Arayüzde statik yöntemler bulunamaz çünkü tüm yöntemler dolaylı olarak soyuttur . Bu nedenle bir arayüzün statik bir yöntemi olamaz.

Soyut sınıfta statik değişken bulunabilir mi?

Evet, soyut bir sınıfın içinde statik değişkenler bulunabilir.

11B. Java Erişimsiz Değiştiriciler - Statik Değişkenler, Yöntemler, Bloklar ve Sınıflar - Soruları - Bölüm B

Soyut sınıfta statik değişken bulunabilir mi?

Evet, soyut bir sınıfın içinde statik değişkenler bulunabilir.

Bir yöntemi ne zaman statik olarak tanımlayacaksınız?

Sınıfın nesnesi oluşturulmadan önce bile bir yöntem erişilmesi gerekiyorsa, yöntemi **static** olarak bildirmeliyiz .

Statik bir yöntem veya statik bir kod bloğuna uygulanan kısıtlama nedir?

Statik bir yöntem, bir örnek oluşturmadan örnek değişkenlerine başvurmamalıdır ve örneğe başvurmak için "this" operatörünü kullanamaz.

Statik olmayan yöntem statik bağlamdan başvurulamaz mı?

genel sınıf Testi

```
{
/** String[] args ile Statik olmayan ana yöntem**/
public static void main(String[] args)
{
Hoş geldin();
}
geçersiz hoşgeldin()
{
System.out.println("JavaInterviewPoint'e hoş geldiniz");
```

```
}
}
```

Çağırmaya çalıştığımız hoş geldin() yöntemi, örnek düzeyinde bir yöntemdir , onu çağırarak bir örneğimiz yok . Statik yöntemler sınıfa aittir , statik olmayan yöntemler ise sınıfın örneklerine aittir ve bu nedenle "statik olmayan yöntem statik bir bağlamdan başvurulamaz" hatası verilir .

Bir sınıf statik olarak bildirilebilir mi?

Üst seviye sınıfı statik olarak bildiremeyiz, ancak yalnızca iç sınıf statik olarak bildirilebilir.

```
halk sınıf Ölçek
{
    statik sınıf İç Sınıf
    {
        halk statik void InnerMethod()
        { System.out.println( "Statik İç Sınıf!" ); }
    }
    halk statik void main(Dize args[])
    {
        Test.InnerClass.InnerMethod();
    }
}
//çıkış: Statik İç Sınıf!
```

Main() yürütülmeden önce bile "Merhaba" yazdırmak istiyorum. Bunu nasıl başaracaksınız?

İfadeyi statik bir kod bloğunun içine yazdırın. Statik bloklar, sınıf belleğe yüklendiğinde ve hatta bir nesne oluşturulmadan önce yürütülür. Bu nedenle `main()` yönteminden önce yürütülecektir . Ve yalnızca bir kez yürütülecektir.

paket soruları;

```
sınıf test sınıfı
{
    statik {
        Sistem. out .println( "merhaba" );
    }
    halk statik void main(String[] args ) {

    }
}
```

Bu, sonucu "merhaba" olarak yazdıracaktır.

Statik değişkenin önemi nedir?

Statik değişkenler, sınıftaki tüm nesnelerin aynı değişkene başvurduğu sınıf düzeyindeki değişkenlerdir. Bir nesnenin değeri değişirse değişiklik tüm nesnelere yansır.

Bir yöntemin içinde statik bir değişken bildirebilir miyiz?

Statik değişkenler sınıf düzeyindeki değişkenlerdir ve bir yöntemin içinde bildirilemezler. Bildirilirse sınıf derlenmez.

Statik ve statik olmayan bir iç sınıf arasındaki fark nedir?

Statik olmayan bir iç sınıf, sınıfın dış sınıfının örnekleriyle ilişkili nesne örneklerine sahip olabilir. Statik bir iç sınıfın herhangi bir nesne örneği yoktur.

Bir sınıfın yöntemi içinde tanımlanan statik bir değişkene ne olur?

Yapamam. Derleme hatası alırsınız.

Kaç tane statik başlatıcınız olabilir?

İstediğiniz kadar, ancak statik başlatıcılar ve sınıf değişkeni başlatıcıları metinsel sırayla yürütülür ve bu sınıf değişkenleri kapsamda olsa bile, kullanımdan sonra bildirimleri metinsel olarak görünen sınıfta bildirilen sınıf değişkenlerine atıfta bulunmayabilir.

Java'da neden main() yöntemi genel, statik ve geçersizdir?

public : “public”, sınıf dışında kullanılacak bir erişim belirtecidir. Ana yöntemin public olarak bildirilmesi, sınıfın dışında da kullanılabilmesi anlamına gelir.

static : Bir yöntemi çağırmak için nesneye ihtiyacımız var. Bazen bir nesnenin yardımı olmadan bir yöntemi çağırmak gerekebilir. Daha sonra bu metodu static olarak ilan ediyoruz. JVM, anahtar sözcüğün statik olduğunu bildirerek nesne oluşturmadan main() yöntemini çağırır.

void: void dönüş türü, bir yöntem herhangi bir değer döndürmediğinde kullanılır. main() yöntemi herhangi bir değer döndürmez, dolayısıyla main() yöntemi geçersiz olarak bildirilir.

Sınıf değişkenlerinin veya statik değişkenlerin kapsamını veya yaşam süresini açıklayın?

Statik değişkenler sınıfın örneklerine ait değildir. Sınıfı başlatmadan önce bile statik alanlara erişebiliriz. Statik değişken uygulamanın ömrü boyunca hafızada kalır.

Java'da statik içe aktarmayı açıklayın?

Java 5.0'dan itibaren statik değişkenleri kaynak dosyaya aktarabiliriz. Statik üyenin kaynak dosyaya aktarılmasına statik içe aktarma denir. Statik içe aktarmanın avantajı, statik değişkenlere sınıf veya arayüz adı olmadan erişebilmemizdir.

Sözdizimi:

statik paketadı.sınıfadı.statikdeğişkenadı içe aktar;

Örn: statik com.abc.Employee.eno'yu içe aktarın;

Bir sınıftan tüm statik değişkenleri kaynak dosyamıza aktarmak için *.

statik com.abc.Employee'yi içe aktarın.*

12. Java Erişimsiz Değiştiriciler - Soyut Sınıflar ve Yöntemler

Soyut Sınıf

Soyut ” anahtar sözcüğü kullanılarak bildirilen sınıfa soyut sınıf denir. Soyut yöntemlerin (gövdesi olmayan yöntemler) yanı sıra somut yöntemleri (gövdesi olan normal yöntemler) de olabilir. Normal bir sınıfın (soyut olmayan sınıf) soyut yöntemleri olamaz.

Soyut sınıf bildirimi

Soyut bir sınıf, yöntemlerin ana hatlarını çizer ancak tüm yöntemleri mutlaka uygulamaz.

Neden soyut bir sınıfa ihtiyacımız var?

Diyelim ki bir sınıf kuşumuz var. Bir kuşun rengi kuştan kuşa değişir, bu nedenle color() yöntemini soyut olarak bildiririz ve bunu ebeveyn sınıfında uygulamamıza gerek kalmaz. Ancak kuşlar uçacak ve bu ebeveyn sınıfında normal bir yöntem olarak uygulanabilir.

Ebeveyn Sınıfı:

```
paket FPPaket;

halk soyut sınıf Kuş {
    halk soyut geçersiz renk();
    halk statik Dize sinek() {
        geri dönmek "Uçarım" ;
    }
}
```

Çocuk Sınıfı:

```
paket FPPaket;

halk Papağan sınıfı Bird'ü genişletiyor {

    halk statik void main(String[] args ) {
        // TODO Otomatik oluşturulan yöntem saplaması
        Dize sinek dizisi = sinek ();
        Papağan p = yeni Papağan();
        p .renk();
        Sistem. out .println( flystring );
    }
    halk geçersiz renk() {
        Sistem. out .println( "Benim rengim yeşil" );
    }
}
```

Not 1: Yukarıdaki örnekte gördüğümüz gibi, ebeveyn sınıfında tüm yöntemleri uygulamanın zor olduğu veya çoğu zaman gereksiz olduğu durumlar vardır. Bu durumlarda ana sınıfı soyut olarak tanımlayabiliriz, bu da onu kendi başına tamamlanamayan özel bir sınıf haline getirir.

Soyut sınıftan türetilen bir sınıf, ana sınıfta soyut olarak bildirilen tüm yöntemleri uygulamalıdır.

Not 2: Soyut sınıf başlatılamaz, bu da onun nesnesini oluşturamayacağınız anlamına gelir. Bu sınıfı kullanmak için, bu sınıfı genişleten ve soyut yöntemlerin uygulanmasını sağlayan başka bir sınıf oluşturmanız gerekir, ardından o alt sınıfın nesnesini, ana sınıfın soyut olmayan yöntemlerini ve uygulanan yöntemleri (bunlar) çağırmak için kullanabilirsiniz. bunlar ebeveynde soyuttu ancak çocuk sınıfında uygulandı).

Not 3: Eğer bir çocuk soyut ana sınıfın tüm soyut yöntemlerini uygulamiyorsa, o zaman alt sınıfın da soyut olarak bildirilmesi gerekir.

Neden soyut bir sınıfın nesnesini yaratamıyoruz?

Bu sınıflar eksik olduğundan, gövdesi olmayan soyut yöntemlere sahiptirler, yani Java bu sınıfın nesnesini yaratmanıza izin verirse, o zaman birisi bu nesneyi kullanarak soyut yöntemi çağırırsa o zaman ne olur? Çağrılacak yöntemin gerçek bir uygulaması olmayacaktı.

Ayrıca bir nesnenin somut olması nedeniyle. Soyut bir sınıf bir şablon gibidir, bu yüzden onu kullanabilmeniz için onu genişletmeniz ve üzerine inşa etmeniz gerekir.

Soyut sınıf ve Beton sınıfı

Beton sınıfı denir .

Anahtar noktaları:

Soyut bir sınıfın başka bir sınıf tarafından genişletilmediği sürece hiçbir faydası yoktur.

Bir sınıfta soyut bir yöntem bildirirseniz, sınıfın özetini de bildirmeniz gerekir. somut bir sınıfta soyut bir yöneme sahip olamazsınız. Bunun tersi de her zaman doğru değildir: Eğer bir sınıfın herhangi bir soyut yöntemi yoksa o sınıf da soyut olarak işaretlenebilir.

Soyut olmayan bir yöneme de (somut) sahip olabilir.

12A. Java Erişimsiz Değiştiriciler - Soyut Sınıflar ve Yöntemler - Soruları

Röportajda Soyutlama Nasıl Tanımlanır?

Soyutlama, uygulama ayrıntılarını gizleme ve kullanıcıya yalnızca işlevselliği gösterme işlemidir.

Soyut olarak bildirilen ve uygulaması olmayan bir yöntem, soyut yöntem olarak bilinir.

Java'da soyutlamaya ulaşmanın iki yolu vardır

1- Soyut sınıfa göre (%0 ila %100), 2- Arayüze göre (%100)

Java'da soyutlama ve soyut sınıf nedir?

Soyutlama:

Soyutlama, uygulama ayrıntılarını gizleme ve kullanıcıya yalnızca işlevselliği gösterme işlemidir.

Başka bir şekilde, kullanıcıya yalnızca önemli şeyleri gösterir ve örneğin sms göndermek gibi dahili ayrıntıları gizler, siz sadece metni yazıp mesajı gönderirsiniz. Mesaj teslimiyle ilgili dahili işlemleri bilmiyorsunuz.

Soyutlama, nesnenin nasıl yaptığı yerine ne yaptığına odaklanmanızı sağlar.

Soyutlamayı başarmanın yolları

Java'da soyutlamaya ulaşmanın iki yolu vardır

Soyut sınıf (%0 - 100)

Arayüz (%100)

Java'daki soyut sınıf:

Soyut olarak bildirilen bir sınıfa soyut sınıf denir. Yaygınlaştırılması ve yönteminin uygulanması gerekiyor. Örneklenemez.

Örnek soyut sınıf

soyut sınıf A{}

Soyut sınıfların önemli özellikleri şunlardır:

1) Soyut sınıflar başlatılamaz.

2) Soyut bir sınıf soyut yöntemleri, somut yöntemleri veya her ikisini birden içerir.

3) Soyut sınıfı genişleten herhangi bir sınıf, soyut sınıfın tüm yöntemlerini geçersiz kılmalıdır.

4) Soyut bir sınıf 0 veya daha fazla soyut yöntem içerebilir.

Soyut yöntem:

Soyut olarak bildirilen ve uygulaması olmayan bir yöntem, soyut yöntem olarak bilinir. Uygulaması alt sınıfa ertelenmiştir.

Örnek soyut yöntem

```
abstract void printStatus();//gövde yok ve özet yok
```

Soyut sınıfın Java'da yapıcıları olabilir mi?

Evet, soyut sınıf Java'da yapıcıyı bildirebilir ve tanımlayabilir. Soyut sınıfın örneğini oluşturamayacağınız için yapıcı yalnızca yapıcı zincirleme sırasında, yani somut uygulama sınıfının örneğini oluşturduğunuzda çağrılabilir.

Soyut sınıf Java'da arayüz uygulayabilir mi? tüm yöntemlerin uygulanmasını gerektiriyor mu?

Evet, soyut sınıf, apps anahtar sözcüğünü kullanarak arayüzü uygulayabilir. Soyut oldukları için tüm yöntemleri uygulamaya gerek duymazlar.

Type'ı bildirmek için bir arayüzle birlikte soyut bir temel sınıf sağlamak iyi bir uygulamadır.

Soyut sınıf Java'da final olabilir mi?

Hayır, soyut sınıf Java'da final olamaz. Bunları sonlandırmak, soyut sınıfın genişletilmesini engelleyecektir; bu, soyut sınıfı kullanmanın tek yoludur.

Bunlar da birbirine zıttır, abstract anahtar sözcüğü bir sınıfın genişletilmesini sağlar, onu kullanmak için, final anahtar sözcüğü ise bir sınıfın genişletilmesini engeller.

Gerçek dünyada da soyut, eksikliği ifade ederken final, bütünlüğü göstermek için kullanılır. Sonuç olarak, Java'da sınıfınızı hem soyut hem de final yapamazsınız, aynı zamanda bu bir derleme zamanı hatasıdır.

Soyut sınıfın örneğini oluşturabilir misiniz? Soyut sınıfın bir nesnesini yaratabilir misiniz?

Hayır, Java'da soyut sınıfın örneğini oluşturamazsınız, bunlar eksiktir.

Ancak soyut sınıfınız herhangi bir soyut yöntem içermiyorsa bunun örneğini oluşturamazsınız.

Bir sınıf özeti yaparak derleyiciye bunun eksik olduğunu ve başlatılmaması gerektiğini söylediniz. Bir kod soyut sınıfı başlatmaya çalıştığında Java derleyicisi hata verecektir.

Soyut sınıfın soyut yöntemlere sahip olması gerekli midir?

Hayır, soyut bir sınıfın herhangi bir soyut yöntemlere sahip olması zorunlu değildir. Java'da bir sınıfın özetini, bildiriminde yalnızca abstract anahtar kelimesini kullanarak yapabiliriz.

Java'da soyut yöntem nedir?

Soyut bir yöntem, gövdesi olmayan bir yöntemdir. Yöntem bildiriminde soyut anahtar kelimeyi kullanır. Java Arayüzünde bildirilen tüm yöntemler varsayılan olarak soyuttur. İşte Java'daki soyut yöntemin bir örneği

```
public void abstract printVersion();
```

Şimdi, bu yöntemi uygulamak için bu soyut sınıfı genişletmemiz ve yöntemini geçersiz kılmamız gerekiyor.

Soyut sınıf Java'da ana yöntemi içerebilir mi?

Evet, soyut sınıf ana yöntemi içerebilir, bu yalnızca başka bir statik yöntemdir ve herhangi bir örnek oluşturmayana kadar Soyut sınıfını ana yöntemle çalıştırabilirsiniz.

Soyut bir değişkenin kullanımı nedir?

Değişkenler soyut olarak bildirilemez. yalnızca sınıflar ve yöntemler **soyut** olarak bildirilebilir .

Soyut bir sınıf herhangi bir soyut yöntem olmadan tanımlanabilir mi?

Evet mümkün. Bu temel olarak sınıfın örnek oluşturulmasını önlemek içindir.

Bir yöntemin veya sınıfın soyut olması ne anlama gelir?

Soyut bir sınıf başlatılamaz. Soyut yöntemler yalnızca soyut sınıflara dahil edilebilir. Bununla birlikte, soyut bir sınıfın herhangi bir soyut yöntemle sahip olması zorunlu değildir, ancak çoğu böyledir. Soyut bir sınıfın her bir alt sınıfı, üst sınıflarının soyut yöntemlerini geçersiz kılmalı veya soyut olarak bildirilmelidir.

13. Java Erişimsiz Değiştiriciler - Son Anahtar Kelime

Final

Final, erişim dışı bir değiştiricidir ve değişkenlere, yöntemlere ve sınıflara uygulanabilir.

Son Değişken – Değişken değiştirilemez veya yeni değerler atanamaz.

Son Yöntemler – Bunlar için yöntem geçersiz kılma işlemi yapılamaz.

Son Sınıflar – Bu sınıflar devralınamaz.

Son Değişkenler: Son değişkenler değiştirilemez ve bunları değiştirmeye çalıştığınızda “Son alan atanamıyor” derleme hatası alırsınız.

Örnek:

```
paket FPPaket;

halk sınıf FinalKeywordEx {

    final String str = "Subbu'nun Selenium Dersleri" ;
    halk statik void main(String[] args ) {

        FinalKeywordEx fke = new FinalKeywordEx();
        Sistem. out .println( "Son değişken a: " + fke . str );

        fke . str = "Java ve Selenium eğitimleri" ; //Hata:
FinalKeywordEx.str son alanı atanamıyor

    }

}
```

Son Yöntemler: Son yöntemler geçersiz kılınamaz. “Son yöntem üst öğeden geçersiz kılınamıyor” hatası veriyor.

Örnek:

```
paket FPPaket;

sınıf Ebeveyn Sınıfı {

    halk son geçersiz yöntemA() {
        Sistem. out .println( "Bu son yöntemdir" );
    }
}

halk FinalKeywordEx sınıfı ParentClass{' ı genişletir

    halk son geçersiz methodA() { //Hata: ParentClass'tan son yöntem geçersiz
kılınamıyor
        Sistem. out .println( "Son yöntem geçersiz kılınamaz" );
    }
    halk statik void main(String[] args ) {

    }
}
```

Final Dersi: Final dersleri devralınamaz. Bunları devralmaya çalışırsanız, "Çocuk sınıfı türü, son sınıf üst sınıfını alt sınıflandıramaz" hatasını alırsınız.

Örnek:

```
paket FPPaket;

son sınıf Ebeveyn Sınıfı {

}

halk FinalKeywordEx sınıfı genişletiliyor ParentClass { //Hata: FinalKeywordEx
türü, ParentClass son sınıfının alt sınıfını oluşturamaz

    halk statik void main(String[] args ) {

    }

}
```

13A. Java Erişimsiz Değiştiriciler - Son Anahtar Kelime - Soruları

Bir main() yöntemi final olarak bildirilebilir mi?

Evet. Miras alan herhangi bir sınıf, kendi varsayılan **main()** yöntemine sahip olamayacaktır . Çünkü son yöntem geçersiz kılınamaz.

```
paket FPPaket;

sınıf Ebeveyn Sınıfı {
    halk statik son void main(String[] args ) {

    }
}
```

```

halk FinalKeywordEx sınıfı ParentClass{' 1 genişletir

    halk statik geçersiz main(String[] args ) { //Hata: Üst sınıftan son yöntem
geçersiz kılınamıyor

    }

}

```

Bir değişkeni final olarak bildirmenin amacı nedir?

Son değişkenin değeri değiştirilemez . **final** değişkenleri kullanılmadan önce başlatılmalıdır.

Genel bir örnek bir ev inşa etmektir. Bir ev inşa ettiğinizde mevcut toplam metrekare sabittir. Yani bu bir son değişkendir.

Bir yöntemi nihai olarak bildirmenin etkisi nedir?

Final olarak bildirilen bir yöntem geçersiz kılınamaz. Bir alt sınıf, farklı bir uygulamayla aynı yöntem imzasına sahip olamaz.

Sınıfımın başka bir sınıf tarafından miras alınmasını istemiyorum. Ne yapmalıyım?

final olarak ilan etmelisiniz . Ancak **soyut** bir sınıfsa sınıfınızı **final** olarak tanımlayamazsınız . **Final** olarak bildirilen bir sınıf başka bir sınıf tarafından genişletilemez.

Java API'de tanımlanan son sınıflara birkaç örnek verebilir misiniz?

Java.lang.String, **Java.lang.Math** son sınıflardır .

finalin nihayet ve finalize()'dan farkı nedir?

final , bir sınıfa, yönteme veya değişkene uygulanabilen bir değiştiricidir. **final** sınıfı devralınamaz, **final** yöntemi geçersiz kılınamaz ve **son** değişken değiştirilemez.

nihayet, try bloğu kod bölümü tarafından bir istisna oluşturulup oluşturulmadığına bakılmaksızın yürütülen bir istisna işleme kodu bölümüdür.

finalize(), kaynak serbest bırakma etkinliği için son bir şans vermek üzere çöp toplama nesnesinden hemen önce JVM tarafından yürütülecek olan Object sınıfının bir yöntemidir.

Bir sınıfın veya üyenin nihai olması ne anlama gelir?

Son sınıf miras alınamaz. Bir alt sınıfta son yöntem geçersiz kılınamaz. Son alan başlatıldıktan sonra değiştirilemez ve bildirildiği yerde bir başlatıcı ifadesi içermelidir.

Sabitler nelerdir ve Java'da sabitler nasıl oluşturulur?

Sabitler programın çalışması sırasında değerleri değiştirilemeyen sabit değerlerdir. Final anahtar kelimesini kullanarak Java'da sabitler yaratırız.

Örn: son int sayısı =10;

final String str="Subbus Selenyum Dersleri"

14. Java Arayüzleri

Kısmi soyutlama için soyut sınıf kullanılır. Arayüz tam soyutlama için kullanılır.

Soyutlama, yalnızca ilgili verileri gösterdiğiniz ve bir nesnenin gereksiz ayrıntılarını kullanıcıdan gizlediğiniz bir işlemdir.

Arayüz bir sınıfa benziyor ancak bir sınıf değil. Bir arayüz, tıpkı sınıf gibi yöntemlere ve değişkenlere sahip olabilir, ancak arayüzde bildirilen yöntemler varsayılan olarak soyuttur (yalnızca yöntem imzası, gövde yok). Ayrıca bir arayüzde bildirilen değişkenler varsayılan olarak genel, statik ve finaldir.

Arayüzlerdeki yöntemlerin gövdesi olmadığından, onlara erişebilmeniz için sınıf tarafından uygulanmaları gerekir. Arayüzü uygulayan sınıf, o arayüzün tüm yöntemlerini uygulamalıdır. Ayrıca Java programlama dili birden fazla sınıfı genişletmenize izin vermez, ancak sınıfınızda birden fazla arayüz uygulayabilirsiniz.

Sözdizimi:

```
arayüz DemoArayüz{
    genel geçersiz testmetodu1();
    genel geçersiz test yöntemi2();
}
```

Örnek:

Test Arayüzü Arayüzü:

```
paket FPPaket;

halk arayüz TestArayüz {
    halk geçersiz test1();
    halk geçersiz test2();
}
```

ClassForInterface Sınıfı:

```
paket FPPaket;

halk classForInterface sınıfı TestInterface'i uygular {
    halk geçersiz test1() {
        Sistem. out .println( "test1 uygulanıyor" );
    }
    halk geçersiz test2() {
        Sistem. out .println( "test2 uygulanıyor" );
    }
    halk statik void main(String[] args ) {
        TestInterface cl = new ClassForInterface();
        cl .test1();
        cl .test2();
    }
}
```

Bir arayüz başka bir arayüzü uygulayamaz. Başka bir arayüzü genişletmesi gerekiyor.

Örnek:

Arayüz Testi Arayüz:

```
paket FPPaket;
```

```
halk arayüz TestArayüz {
    halk geçersiz test1();
    halk geçersiz test2();
}
```

Arayüz TestiArayüz1:

```
paket FPPaket;
```

```
halk arayüz TestInterface1 genişletiliyor Test Arayüzü {
    halk geçersiz test3();
    halk geçersiz test4();
}
```

ClassForInterface Sınıfı:

```
paket FPPaket;
```

```
halk classForInterface sınıfı TestInterface1'i uygular {
    halk geçersiz test1() {
        Sistem. out .println( "test1 uygulanıyor" );
    }
    halk geçersiz test2() {
        Sistem. out .println( "test2 uygulanıyor" );
    }
    halk geçersiz test3() {
        Sistem. out .println( "test3 uygulanıyor" );
    }
    halk geçersiz test4() {
        Sistem. out .println( "test4 uygulanıyor" );
    }
    halk statik void main(String[] args ) {
        TestInterface c1 = new ClassForInterface();
        c1 .test1();
        c1 .test2();
        c1 .test3();
        c1 .test4();
    }
}
```

Sınıf yalnızca TestInterface1'i uyguluyor olsa da, TestInterface1, TestInterface'i genişlettiğinden, TestInterface'in tüm yöntemlerini de uygulamak zorundadır.

14A. Java Arayüzleri – Önemli Noktalar

Anahtar noktaları:

- 1) Java'da bir arayüz oluşturamıyoruz. Bu, bir arayüzün nesnesini yaratamayacağımız anlamına gelir
- 2) Arayüz, yöntemlerinin hiçbirinin gövdesi olmadığından tam soyutlama sağlar. Öte yandan soyut sınıf, hem soyut hem de somut (gövdeli yöntemler) yöntemlere sahip olabileceğinden kısmi soyutlama sağlar.
- 3) apps anahtar sözcüğü sınıflar tarafından bir arayüzü uygulamak için kullanılır.

4) Herhangi bir arayüz metodunun class içerisinde uygulamasını sağlarken public olarak belirtilmesi gerekmektedir.

5) Herhangi bir arayüzü uygulayan sınıf, o arayüzün tüm yöntemlerini uygulamalıdır, aksi takdirde sınıf soyut olarak bildirilmelidir.

6) Arayüz özel, korumalı veya geçici olarak ilan edilemez.

7) Tüm arayüz yöntemleri varsayılan olarak soyut ve geneldir.

8) Arayüzde bildirilen değişkenler varsayılan olarak genel, statik ve finaldir.

arayüz Deneyin

```
{
int a=10;
genel int a=10;
public static final int a=10;
son int a=10;
statik int a=0;
}
```

Yukarıdaki ifadelerin tümü aynıdır.

9) Arayüz değişkenleri bildirim sırasında başlatılmalıdır, aksi takdirde derleyici hata verecektir.

arayüz Deneyin

```
{
int x;//Derleme zamanı hatası
}
```

Yukarıdaki kod, x değişkeninin değeri bildirim sırasında başlatılmadığından derleme zamanı hatası verecektir.

10) Herhangi bir uygulama sınıfının içinde, arayüzde bildirilen değişkenleri değiştiremezsiniz çünkü bunlar varsayılan olarak genel, statik ve finaldir. Burada x değişkenine sahip "Try" arayüzünü uyguluyoruz. X değişkeninin değerini ayarlamaya çalıştığımızda, x değişkeni varsayılan olarak public static final olduğundan ve final değişkenleri yeniden başlatılamadığından derleme hatasıyla karşılaştık.

sınıf Örnek uygular Dene

```
{
public static void main(String args[])
{
x=20; //derleme zamanı hatası
}
}
```

11) Bir arayüz herhangi bir arayüzü genişletebilir ancak uygulayamaz. Sınıf arayüzü uygular ve arayüz arayüzü genişletir.

12) Bir sınıf herhangi bir sayıda arayüzü uygulayabilir.

13) İki arayüzde iki veya daha fazla aynı yöntem varsa ve bir sınıf her iki arayüzü de uyguluyorsa, yöntemin bir kez uygulanması yeterlidir.

arayüz bir

```
{
```

```

genel geçersiz aaa();
}
arayüz B
{
genel geçersiz aaa();
}
sınıf Merkezi A,B'yi uygular
{
genel boşluk aaa()
{
//Burada herhangi bir kod var
}
public static void main(String args[])
{
//ifadeler
}
}

```

14) Bir sınıf, aynı ada sahip ancak farklı dönüş tipine sahip yöntemlere sahip iki arayüzü uygulayamaz.

```

arayüz bir
{
genel geçersiz aaa();
}
arayüz B
{
genel int aaa();
}

sınıf Merkezi A,B'yi uygular
{

genel geçersiz aaa() // hata
{
}
public int aaa() // hata
{
}
public static void main(String args[])
{

}
}

```

15) Değişken adları çakışmaları arayüz adına göre çözülebilir.

```

arayüz bir
{
intx=10;
}
arayüz B
{

```



```

intx=100;
}
Merhaba sınıfı A,B'yi uygular
{
public static void Main(String args[])
{
/* x'e yapılan referans belirsiz her iki değişken de x'tir
* bu yüzden sorunu çözmek için arayüz adını kullanıyoruz
* değişken
*/
System.out.println(x);
System.out.println(Ax);
System.out.println(Bx);
}
}

```

14B. Java Arayüzleri - Soruları

Java'da arayüz tanımlansın mı?

Arayüz soyut yöntemler ve sabitlerin koleksiyonudur. Bir arayüz aynı zamanda saf veya yüzde 100 soyut sınıf olarak da tanımlanır. Soyut erişim değiştiricisini tanımlasak da tanımlamasak da, arayüzler örtülü olarak soyuttur. Bir sınıf uygulayan arayüz, arayüzde tanımlanan tüm soyut yöntemleri geçersiz kılar. Uygulama anahtar sözcüğü arayüzü uygulamak için kullanılır.

Arayüzün amacı nedir?

Arayüz bir sözleşmedir. Arayüz iki nesne arasındaki iletişim gibi davranır. Bir arayüz tanımlarken, sınıfımızın ne yapması gerektiğini, nasıl yaptığını değil, bir sözleşmeyi tanımlıyoruz. Bir arayüz, bir yöntemin ne yaptığını tanımlamaz. Arayüzün gücü, ilgisiz farklı sınıfların arayüzü uygulayabildiğinde ortaya çıkar. Arayüzler çalışma zamanında dinamik yöntem çözümlemesini destekleyecek şekilde tasarlanmıştır.

Java'daki arayüzlerin özelliklerini açıklayın?

- 1) Arayüzlerde tanımlanan tüm yöntemler, soyut değiştirici bildirilmese bile örtülü olarak soyuttur.
- 2) Arayüzdeki tüm yöntemler, public olarak bildirilse de ilan edilmese de, publictir.
- 3) Arayüz içinde bildirilen değişkenler varsayılan olarak genel, statik ve finaldir.
- 4) Arayüzler başlatılamaz.
- 5) Arayüz içinde statik yöntemler bildiremeyiz.
- 6) Arayüzü uygulamak için 'implements' anahtar sözcüğü kullanılır.
- 7) Sınıfın aksine, arayüz herhangi bir sayıda arayüzü genişletebilir.
- 8) Arayüz içinde bir sınıf tanımlayabiliriz ve sınıf, arayüze iç sınıf gibi davranır.
- 9) Bir arayüz bir sınıfı genişletebilir ve bir arayüzü uygulayabilir
- 10) Java'da çoklu kalıtım arayüzler aracılığıyla sağlanır.

C Sınıfı, m1 ve m2 bildirimlerini içeren Arayüz l'i uygular. C Sınıfı, m2 yönteminin uygulanmasını sağlamıştır. C Sınıfı bir nesne oluşturabilir miyim?

Hayır mümkün değil. **C Sınıfı**, **Arayüz** l'deki tüm yöntemlerin uygulanmasını sağlamalıdır . **C Sınıfı m1 yöntemi** için uygulama sağlamadığından , **abstract** olarak bildirilmesi gerekir . Soyut sınıflar başlatılamaz.

Arayüz içindeki bir yöntem final olarak bildirilebilir mi?

Hayır mümkün değil. Bunu yapmak derleme hatasına neden olur. Bir **arayüzde** yöntem bildirimi için geçerli olan yegâne değiştiriciler **public** ve **abstract**'tir .

Bir Arayüz başka bir Arayüz uygulayabilir mi?

Arayüzler uygulama sağlamaz, dolayısıyla bir arayüz başka bir arayüzü uygulayamaz. Arayüz başka bir arayüzü genişletir.

Bir Arayüz başka bir Arayüzü genişletebilir mi?

Evet, bir Arayüz başka bir Arayüzü devralabilir, bu nedenle bir Arayüz birden fazla Arayüzü genişletebilir.

Bir Sınıf birden fazla Sınıfı genişletebilir mi?

Mümkün değil. Bir Sınıf yalnızca bir sınıfı genişletebilir ancak herhangi bir sayıda Arayüzü uygulayabilir.

Neden bir Arayüz birden fazla Arayüzü genişletebiliyor ama bir Sınıf neden birden fazla Sınıfı genişletemiyor?

Temel olarak Java çoklu kalıtıma izin vermez, bu nedenle bir Sınıf yalnızca bir Sınıfı genişletmekle sınırlıdır. Ancak Arayüz saf bir soyutlama modelidir ve sınıflar gibi kalıtım hiyerarşisine sahip değildir (tüm sınıfların temel sınıfının Object olduğunu unutmayın). Böylece bir Arayüzün birden fazla Arayüzü genişletmesine izin verilir.

Bir Arayüz nihai olabilir mi?

Mümkün değil. Bunu yapmak derleme hatasıyla sonuçlanacaktır.

Arayüzlerdeki değişkenler için özel ve korumalı değiştiriciler tanımlayabilir miyiz?

HAYIR.

Bir Arayüzdeki yöntemler için hangi değiştiricilere izin verilir?

Arayüzlerdeki yöntemler için yalnızca **genel** ve **soyut değiştiricilere izin verilir**.

Bir nesne referansı ne zaman bir arayüz referansına dönüştürülebilir?

Nesne başvuru alan arayüzü uyguladığında, bir nesne referansı bir arayüz referansına dönüştürülür.

Arayüz içerisinde statik metodlar tanımlayabilir miyiz?

Arayüz içinde statik yöntemler bildiremeyiz. Arayüzlerde yalnızca örnek yöntemlere izin verilir. Arayüz yöntemleri için yalnızca genel ve soyut değiştiricilere izin verilir. Arayüz içerisinde statik metodları bildirmeye çalıştığımızda "Illegal modifier for the arayüz metodu Classname.methodName(); yalnızca genel ve özete izin verilir".

Soyut sınıf ve arayüz arasındaki fark?

Arayüz	Soyut Sınıf
---------------	--------------------

Arayüz yalnızca soyut yöntemleri içerir	Soyut sınıf soyut yöntemleri, somut yöntemleri veya her ikisini birden içerir
Arayüzdeki yöntemlere ilişkin Erişim Belirleyicileri herkese açık olmalıdır	soyut sınıftaki yöntemler için herhangi bir erişim belirtecine sahip olabiliriz .
final olmalıdır	belirtecine sahip olması dışında
Java'da Çoklu Kalıtım arayüz kullanılarak uygulanır	Soyut sınıfı kullanarak çoklu kalıtıma ulaşamayız .
apps anahtar sözcüğünü kullanırsınız	Soyut bir sınıfı genişletmek için extends anahtar sözcüğünü kullanırsınız.

15. Java Mirası

Bir sınıfın başka bir sınıfın özelliklerini (veri üyeleri) ve işlevlerini (yöntemlerini) edindiği sürece miras denir. Kalıtımın amacı, bir sınıfın yalnızca benzersiz özellikleri yazması ve geri kalan ortak özellik ve işlevlerin başka bir sınıftan genişletilebilmesi için kodun yeniden kullanılabilirliğini sağlamaktır.

Çocuk Sınıfı:

Başka bir sınıfın özelliklerini genişleten sınıfa çocuk sınıf, alt sınıf veya türetilmiş sınıf denir.

Ebeveyn Sınıfı:

Özellikleri ve işlevleri başka bir sınıf tarafından kullanılan (miras alınan) sınıf, ebeveyn sınıfı, süper sınıf veya Temel sınıf olarak bilinir.

Örnek:

ParentClass.java

paket FPPaket;

```

halk sınıf Ebeveyn Sınıfı {
    String name = "Subbu" ;
    String role = "Yönetici" ;
    String şirket = "IBM" ;
    halk statik void displayDetails() {
        EbeveynSınıfı bilgisayar = yeni EbeveynSınıfı();
        Sistem. out .println( "Ad: " + pc . adı );
        Sistem. out .println( "Rol " + pc . rolü );
        Sistem. out .println( "Şirket: " + pc . şirket );
    }
}

```

ChildClass.java

paket FPPaket;

```

halk class ChildClass, ParentClass'ı genişletir {

    halk statik void main(String[] args ) {

```

```

        ÇocukSınıfı c1 = new ÇocukSınıfı();
        Sistem. out .println( "Ad: " + c1 . name );
        Sistem. out .println( "Rol " + c1 . rol );
        Sistem. out .println( "Şirket: " + c1 . şirket );
    }
}

```

15A. Yapıcılar ve Kalıtım

Yapıcılar ve Kalıtım:

Bir alt sınıfın yapıcısı çağrıldığında, varsayılan olarak üst sınıfın yapıcısını çağırır.

Ebeveyn Sınıfı:

```

paket FPPaket;

halk sınıf Ebeveyn Sınıfı {
    String name = "Subbu" ;
    String role = "Yönetici" ;
    String şirketi = "IBM" ;
    EbeveynSınıfı(){
        Sistem. out .println( "Ebeveyn Sınıfı Oluşturucusu" );
    }
    halk statik void displayDetails() {
        EbeveynSınıfı bilgisayar = yeni EbeveynSınıfı();
        Sistem. out .println( "Ad: " + pc . adı );
        Sistem. out .println( "Rol " + pc . rolü );
        Sistem. out .println( "Şirket: " + pc . şirket );
    }
}

```

Çocuk Sınıfı:

```

paket FPPaket;

halk class ChildClass, ParentClass'ı genişletir {
    ÇocukSınıfı(){
        Sistem. out .println( "Çocuk Sınıfı Oluşturucusu" );
    }
    halk statik void main(String[] args ) {
        ÇocukSınıfı c1 = new ÇocukSınıfı();
        Sistem. out .println( "Ad: " + c1 . name );
        Sistem. out .println( "Rol " + c1 . rol );
        Sistem. out .println( "Şirket: " + c1 . şirket );
    }
}

```

15B. Kalıtım ve Yöntemin Geçersiz Kılması

Kalıtım ve Yöntemin Geçersiz Kılması:

Ebeveyn sınıfının aynı yöntemini alt sınıfta bildirirsek buna yöntem geçersiz kılma denir.

Alt yöntem sınıfını alt sınıf nesnesini kullanarak çağırabiliriz.

Super anahtar sözcüğünü kullanarak ebeveyn sınıfı yöntemini çağırabiliriz.

Örnek:

Ebeveyn Sınıfı:

```
paket FPPaket;

halk sınıf Ebeveyn Sınıfı {

    EbeveynSınıfı(){
        Sistem. out .println( "Ebeveyn Sınıfı Oluşturucusu" );
    }
    halk void displayDetails() {
        Sistem. out .println( "Ben ebeveyn sınıfıyım" );
    }
}
```

Çocuk Sınıfı:

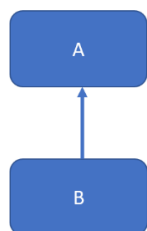
```
paket FPPaket;

halk class ChildClass, ParentClass'ı genişletir {
    ÇocukSınıfı(){
        Sistem. out .println( "Çocuk Sınıfı Oluşturucusu" );
    }
    halk void displayDetails() {
        Sistem. out .println( "Ben alt sınıfa aitim" );
        süper .displayDetails();
    }
    halk statik void main(String[] args ) {
        ÇocukSınıfı c1 = new ÇocukSınıfı();
        c1 .displayDetails();
    }
}
```

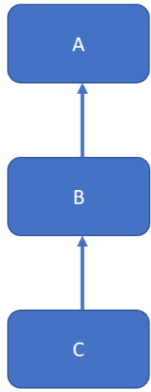
15C. Farklı Miras Türleri

Miras Türleri:

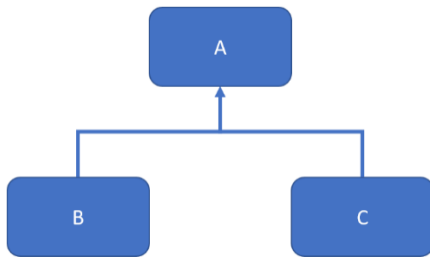
Tek Kalıtım: Bir sınıfın yalnızca başka bir sınıfa yayılmasına tekli miras denir.



Çok Düzeyli Kalıtım: Çok düzeyli kalıtımda bir temel sınıfımız var ve bu sınıfı bir alt sınıftan genişletiyoruz ve bu sınıf başka bir alt sınıftan genişletilecek.

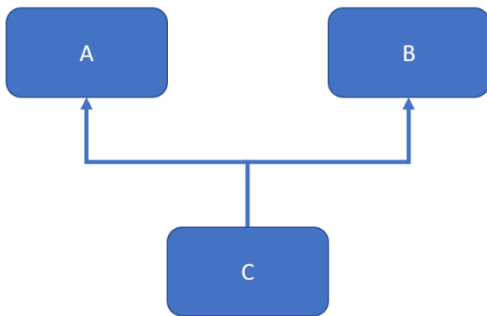


Hiyerarşik Kalıtım: Hiyerarşik mirasta, tek bir sınıf birden fazla sınıf tarafından genişletilecektir.

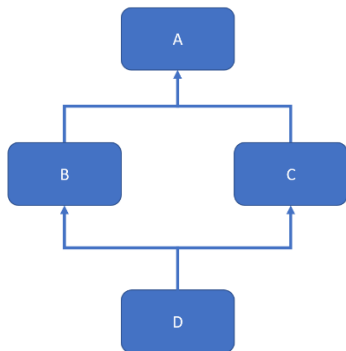


Çoklu Kalıtım: Çoklu kalıtımda, tek bir sınıf birden fazla sınıfı genişletir.

Java, sınıflar için çoklu kalıtımı desteklemez. Yalnızca arayüzler için desteklenir.



Hibrit Kalıtım: Tek ve çoklu kalıtımın birleşimidir.



15D. Miras - Soruları

Miras Nedir?

Kalıtım, bir sınıfın başka bir sınıfta tanımlanan yapıyı ve davranışı paylaştığı bir kavramdır. Kalıtım tek sınıfa uygulanıyorsa Tekli Kalıtım, birden fazla sınıfa bağlıysa çoklu Kalıtım denir.

Java'da mirasın önemini açıklayın?

Yeniden Kullanılabilirlik: Kalıtımın en büyük avantajı kodun yeniden kullanılmasıdır. Kalıtım kullanarak kodun kopyalanmasını önleyebiliriz. Tüm ortak durum ve davranışları bu sınıfa yerleştirebiliriz, bu sınıfı genişleterek şunları yapabiliriz:

Genişletilebilirlik: Mevcut koda dokunmadan uygulamamıza yeni işlevler ekleyebiliriz. Örneğin Ms word'ü ele alırsak, msword'ün word 2003,2007 gibi birçok versiyonuyla karşılaştık. Her yeni kod yazmadıklarında mevcut kodu ve bazı özellikleri yeniden kullanırlar.

Java çoklu kalıtımı destekliyor mu?

Java çoklu kalıtımı desteklemez.

Bir sınıf, üst sınıfının yapıcılarını miras alır mı?

Bir sınıf, yapıcılarını süper sınıflarının herhangi birinden miras almaz.

Java'da alan gizlemeyi açıklayın?

Eğer üst sınıf ve alt sınıf aynı alanlara sahipse, alt sınıf üst sınıf alanlarını geçersiz kılamaz. Bu durumda alt sınıf alanları süper sınıf alanlarını gizler. Eğer alt sınıfta süper sınıf değişkenleri kullanmak istiyorsak, süper sınıf değişkenlerine erişmek için süper anahtar sözcüğünü kullanırız.

Java tarafından desteklenen farklı miras türleri nelerdir?

Java şunları destekler:

Tek Miras

Çok Düzeyli Kalıtım

Hiyerarşik Miras

Java Çoklu Kalıtımı ve Hibrit Kalıtımı desteklemez.

Kalıtım ve Kapsülleme arasındaki fark nedir?

Kalıtım, ebeveyn-çocuk ilişkisi yaratan nesne yönelimli bir kavramdır. Ebeveyn sınıfı için yazılan kodu yeniden kullanmanın yollarından biridir ancak aynı zamanda Polimorfizmin temelini oluşturur. Öte yandan, Kapsülleme, bir sınıfın iç ayrıntılarını gizlemek için kullanılan nesne yönelimli bir kavramdır; örneğin HashMap, elemanların nasıl saklanacağını ve karma değerlerin nasıl hesaplanacağını kapsüller.

Kalıtım ve Soyutlama arasındaki fark nedir?

Soyutlama, ayrıntıları soyutlayarak nesneleri gizlemek için kullanılan nesne yönelimli bir kavramdır. Tasarım sisteminde yardımcı olur. Öte yandan Kalıtım kodun yeniden kullanılmasına izin verir. Daha önce kodladığınız işlevselliği Inheritance kullanarak yeniden kullanabilirsiniz.

Java'da özel bir yöntemi geçersiz kılabilir miyiz?

Hayır, Java'da özel bir yöntemi geçersiz kılamazsınız çünkü özel yöntem, geçersiz kılma için gerekli olan Java'daki alt sınıf tarafından miras alınmaz. Aslında, özel bir yöntem sınıf dışındaki hiç kimse tarafından görülemez ve daha da önemlisi, özel yöntem yapıları bir çağrı, gerçek nesne kullanılarak çalışma zamanı yerine derleme zamanında Type bilgileri kullanılarak çözümlenir.

Bir sınıf Java'da birden fazla arayüz uygulayabilir mi?

Evet, bir sınıf Java'da birden fazla arayüz uygulayabilir; örneğin bir sınıf aynı anda hem Karşılaştırılabilir hem de Serileştirilebilir olabilir. Bu nedenle, Etkili Java'da açıklandığı gibi Türü tanımlamak için en iyi kullanım arayüzü olmalıdır. Bu özellik, bir sınıfın programda polimorfik bir rol oynamasına olanak tanır.

Java'da bir sınıf birden fazla sınıfı genişletebilir mi?

Hayır, bir sınıf Java'da yalnızca bir sınıfı daha genişletebilir. Her sınıf da varsayılan olarak Java'daki `Java.lang.Object` sınıfını genişletir.

Java'da bir arayüz birden fazla arayüzü genişletebilir mi?

Evet, sınıflardan farklı olarak Java'da bir arayüz birden fazla arayüzü genişletebilir.

Alt sınıfın bir nesnesini süper sınıf tipindeki bir referans değişkeninde tutuyorsanız, bir alt sınıfın yöntemini nasıl çağırırsınız?

Önce üst sınıfın referans değişkenine göre tutulan nesneyi alt sınıfa aktararak alt sınıfın bir yöntemini çağırabilirsiniz. Nesneyi alt sınıf referans türünde tuttuğunuzda, alt sınıftan yöntemleri çağırabilirsiniz. Daha fazla ayrıntı için tür dönüştürmenin Java'da nasıl çalıştığını görün.

16. Java Polimorfizmi

Polimorfizm, bir yöntemin üzerinde işlem yaptığı nesneye bağlı olarak farklı şeyler yapabilme yeteneğidir. Başka bir deyişle polimorfizm, bir arayüzü tanımlamanıza ve birden fazla uygulamaya sahip olmanıza olanak tanır.

Polimorfizm Türleri:

1. Statik Polimorfizm / Derleme zamanı polimorfizmi
2. Dinamik Polimorfizm / Çalıştırma zamanı polimorfizmi

Yöntemin aşırı yüklenmesi statik polimorfizmdir.

Yöntem geçersiz kılma dinamik polimorfizmdir.

Örnek:

Man.java

```
paket FPPaket;

halk sınıf Adam {
    halk geçersiz eylem() {
        Sistem.out.println( "Tara, Yürü ve Koş" );
    }
}
```


Çocuk.java

```

paket FPPaket;

halk sınıf Çocuk Man'i genişletir {
    halk geçersiz eylem() {
        Sistem. out .println( "Tarama" );
    }
    halk statik void main(String[] args ) {
        Adam m = yeni Çocuk();
        m .eylem();
    }
}

```

Youngster.java

```

paket FPPaket;

halk sınıf Genç Adamı genişletiyor {
    halk geçersiz eylem() {
        Sistem. out .println( "Yürü ve Koş" );
    }
    halk statik void main(String[] args ) {
        Adam m = yeni Genç();
        m .eylem();
    }
}

```

Oldman.java

```

paket FPPaket;

halk Oldman sınıfı Man'i genişletiyor {
    halk geçersiz eylem() {
        Sistem. out .println( "Yürüyüş" );
    }
    halk statik void main(String[] args ) {
        Adam m = yeni Oldman();
        m .eylem();
    }
}

```

16A. Java Polimorfizmi - Soruları

Aşırı yükleme ve geçersiz kılma kullanılarak hangi nesne yönelimli konseptle ulaşılır?

Polimorfizm.

Polimorfizm nedir ve türleri nelerdir?

Tek bir görev farklı şekillerde yapılabilir.

Yöntemin aşırı yüklenmesi (derleme zamanı polimorfizmi), yöntemi geçersiz kılma (çalışma zamanı polimorfizmi)

Yöntem geçersiz kılma nedir?

Çocuk sınıfı için bir yöntemin özel uygulaması.

Yöntem aşırı yüklemesi nedir?

Bir sınıfın aynı adda ancak farklı parametrelere sahip birden fazla yöntemi varsa, buna Yöntem Aşırı Yükleme denir.

Yöntem aşırı yükleme ve geçersiz kılma arasındaki fark?

Yöntem aşırı yüklemesi	Yöntem geçersiz kılma
Metodun aşırı yüklenmesi durumunda metodun imzası değişir	Yöntemin geçersiz kılınması durumunda aynı kalır.
Yöntemi bir sınıfta aşırı yükleyebilir	Yalnızca alt sınıfta yapılabilir.
Java'da statik, son veya özel yöntemi aşırı yükleyebilir	Java'da statik, son ve özel yöntem geçersiz kılınmıyor
Java'daki aşırı yüklenmiş yöntem statik bağlama ile bağlanır	Geçersiz kılınan yöntemler dinamik bağlamaya tabidir.

Çalışma zamanı polimorfizmi ve derleme zamanı polimorfizmi nedir?

Derleme zamanı polimorfizmi:

Java'daki yöntemin aşırı yüklenmesinden başka bir şey değildir. Basit bir ifadeyle, bir sınıfın aynı adda ancak farklı sayıda argümana veya farklı argüman türlerine veya her ikisine birden sahip birden fazla yöntemi olabileceğini söyleyebiliriz.

Çalışma zamanı polimorfizmi:

Çalışma zamanı polimorfizmi veya dinamik yöntem gönderimi, geçersiz kılınan bir yönteme yapılan çağrının derleme zamanı yerine çalışma zamanında çözümlendiği bir süreçtir. Bu süreçte geçersiz kılınan bir yöntem, bir süper sınıfın referans değişkeni aracılığıyla çağrılır. Çağrılacak yöntemin belirlenmesi, referans değişkeni tarafından atıfta bulunulan nesneye bağlıdır.

Veri üyesi aracılığıyla polimorfizme ulaşabilir miyiz?

Hayır, Polimorfizm her zaman bir nesnenin davranışı yoluyla elde edilir, yalnızca bir nesnenin özellikleri polimorfizm durumunda herhangi bir rol oynamaz.

main() yöntemini aşırı yükleyebilir miyiz?

Evet, main metodunu aşırı yükleyerek bir sınıfta birçok main() metoduna sahip olabiliriz.

```
paket FPPaket;
```

```
halk sınıf AnaAşırı Yükleme {
```

```
    halk statik void main(String[] args ) {
```

```
        Sistem. out .println( "Normal ana yöntem" );
```

```
        ana (3);
```

```
        ana ( "Subbu" , "Selenyum" );
```

```
    }
```

```
    halk statik geçersiz ana ( int) argümanlar ) {
```

```
        Sistem. out .println( "Tek argümanlı normal ana yöntem" );
```

```

    }

    halk statik void main(Dize args1 , String args2 ) {

        Sistem. out .println( "İki argümanlı normal ana yöntem" );

    }

}

```

Yukarıdaki programı çalıştırdığınızda, kaç tane argüman ilettiğinize bakılmaksızın her zaman ilk ana yöntemi yazdırır, çünkü bu bir sınıfın giriş noktasıdır.

Bu nedenle programdan diğer ana yöntemleri çağırmanız gerekir.

17. Java Süper Anahtar Kelimesi

Super anahtar sözcüğü doğrudan ebeveyn sınıfın nesnelerini ifade eder.

Süper anahtar kelime şu amaçlarla kullanılır:

- 1) Hem ebeveyn hem de çocuk sınıfın aynı adı taşıyan üyeleri olduğunda ana sınıfın veri üyelerine erişmek için
- 2) Ana sınıfın argümansız ve parametrelili yapısını açıkça çağırmak için
- 3) Çocuk sınıf bunu geçersiz kıldığında ana sınıfın yöntemine erişmek için yöntem.

Ebeveyn Sınıfı Değişkenlerine Erişmek İçin Süper Anahtar Kelime:

17A. Java Temelleri - Süper Anahtar Kelimeyle Ebeveyn Sınıfı Değişkenlerine Erişim

Super anahtar sözcüğünü kullanarak Parent sınıfı değişkenlerine erişme:

ParentClass.java:

```

paket FPPaket;

halk sınıf Ebeveyn Sınıfı {

    int yaş = 10;
    String name = "Subbu" ;

}

```

ChildClass.java

```

paket FPPaket;

halk class ChildClass, ParentClass'ı genişletir {
    int yaş = 15;
    Dize adı = "Venkat" ;
    halk void displayParentDetails() {

```

```

        Sistem. out .println( süper . yaş );
        Sistem. out .println( süper . isim );
    }
    halk statik void main(String[] args ) {
        ÇocukSınıfı cl = new ÇocukSınıfı();
        cl .displayParentDetails();
        Sistem. out .println( cl . yaş );
        Sistem. out .println ( cl.isim ) ;
    }
}

```

17B. Java Temelleri - Ebeveyn Sınıfı No-Arg Yapıcısına Erişim

Ebeveyn Sınıfı Argüman Oluşturucusuna Erişim:

Ebeveyn Sınıfı:

```

paket FPPaket;

halk sınıf Ebeveyn Sınıfı {

    EbeveynSınıfı(){
        Sistem. out .println( "Ana sınıfın bağımsız değişken oluşturucusu"
    );
    }
    Ebeveyn Sınıfı( int değer ){
        Sistem. out .println( "Argümanlı ana sınıf kurucusu " + değer );
    }
}

```

Çocuk Sınıfı:

```

paket FPPaket;

halk class ChildClass, ParentClass'ı genişletir {
    ÇocukSınıfı(){
        süper (100);
        Sistem. out .println( "Ben alt sınıf kurucusuyum" );
    }
    halk statik void main(String[] args ) {
        ÇocukSınıfı cl = new ÇocukSınıfı();
    }
}

```

Yöntem Geçersiz Kıldığında Süper Anahtar Kelime:

Örnek:

ParentClass.java:

```

paket FPPaket;

halk sınıf Ebeveyn Sınıfı {
    halk geçersiz yöntem geçersiz kılma() {
        Sistem. out .println( "Bu yöntem geçersiz kılınacak" );
    }
}

```

```
    }
}
```

ChildClass.java:

```
paket FPPaket;

halk class ChildClass, ParentClass'ı genişletir {
    halk geçersiz yöntem geçersiz kılma() {
        Sistem. out .println( "Bu yöntem geçersiz kılındı" );
    }
    halk geçersiz ebeveyn yöntemi() {
        süper .methodoverriding();
    }
    halk statik void main(String[] args ) {
        ÇocukSınıfı cl = new ÇocukSınıfı();
        cl .metodooverriding();
        cl .parentmethod();
    }
}
```

17C. Ebeveyn Sınıfı Bağımsız Değişken Oluşturucusuna Erişim

Ebeveyn Sınıfı Argüman Oluşturucusuna Erişim:

Ebeveyn Sınıfı:

```
paket FPPaket;

halk sınıf Ebeveyn Sınıfı {

    EbeveynSınıfı(){
        Sistem. out .println( "Ana sınıfın bağımsız değişken oluşturucusu"
    );
    }
    Ebeveyn Sınıfı( int değer ){
        Sistem. out .println( "Argümanlı ana sınıf kurucusu " + değer );
    }
}
```

Çocuk Sınıfı:

```
paket FPPaket;

halk class ChildClass, ParentClass'ı genişletir {
    ÇocukSınıfı(){
        süper (100);
        Sistem. out .println( "Ben alt sınıf kurucusuyum" );
    }
    halk statik void main(String[] args ) {
        ÇocukSınıfı cl = new ÇocukSınıfı();
    }
}
```

17D. Yöntem Geçersiz Kıldığında Süper Anahtar Kelime

Yöntem Geçersiz Kıldığında Süper Anahtar Kelime:

Örnek:

ParentClass.java:

```
paket FPPaket;

halk sınıf Ebeveyn Sınıfı {
    halk geçersiz yöntem geçersiz kılma() {
        Sistem. out .println( "Bu yöntem geçersiz kılınacak" );
    }
}
```

ChildClass.java:

```
paket FPPaket;

halk class ChildClass, ParentClass'ı genişletir {
    halk geçersiz yöntem geçersiz kılma() {
        Sistem. out .println( "Bu yöntem geçersiz kılındı" );
    }
    halk geçersiz ebeveyn yöntemi() {
        süper .methodoverriding();
    }
    halk statik void main(String[] args ) {
        ÇocukSınıfı c1 = new ÇocukSınıfı();
        c1 .metodoverriding();
        c1 .parentmethod();
    }
}
```

17E. Geçersiz Kılmamış Bir Ana Yönteme Erişme

Geçersiz kılınmayan ana yönteme nasıl erişilir?

Bir ana yöntem geçersiz kılınmadığında, ona yalnızca alt sınıf nesnesiyle erişebilirsiniz.

Örnek:

ParentClass.java:

```
paket FPPaket;

halk sınıf Ebeveyn Sınıfı {
    halk geçersiz yöntem geçersiz kılma() {
        Sistem. out .println( "Bu yöntem geçersiz kılınacak" );
    }
    halk void methodnotoverriding() {
        Sistem. out .println( "Geçersiz kılınmadım" );
    }
}
```

ChildClass.java:

```

paket FPPaket;

halk class ChildClass, ParentClass'ı genişletir {
    halk geçersiz yöntem geçersiz kılma() {
        Sistem. out .println( "Bu yöntem geçersiz kılındı" );
    }
    halk geçersiz ebeveyn yöntemi() {
        süper .methodoverriding();
    }
    halk statik void main(String[] args ) {
        ÇocukSınıfı c1 = new ÇocukSınıfı();
        c1 .metodoverriding();
        c1 .parentmethod();
        c1 .methodnotnotoverriding();
    }
}

```

17F. Java Süper Anahtar Kelime - Soruları

Süper anahtar kelime nedir?

Super anahtar sözcüğü, üst sınıf yöntemlerinden birini geçersiz kılan geçersiz kılınan yöntemi çağırmak için kullanılır. Bu anahtar kelime, geçersiz kılınan yöntemlere ve aynı zamanda üst sınıfın gizli üyelerine erişime izin verir.

Ayrıca bir kurucudan gelen çağrıyı üst sınıftaki bir kurucuya iletir.

Neden super veya this anahtar kelimesini ana yöntemde kullanamıyoruz?

Statik anahtar kelime esas olarak Java'da verimli bellek yönetimi için kullanılır. Statik anahtar kelime değişkenlere, yöntemlere, bloklara ve iç içe geçmiş sınıflara uygulanabilir.

Statik anahtar kelime sınıfın örneğinden ziyade sınıfa aittir. Bir yöntem statik olarak bildirildiğinde, ona yalnızca Sınıf adıyla erişilebilir. Statik yöntem yalnızca statik veri üyelerine ve yöntemlerine erişebilir. Statik olmayan üyelere, yani örnek değişkenlere erişemez.

Süper anahtar kelime: Aşağıdaki kullanımlara sahiptir,

- Bu, ana sınıf nesnesine atıfta bulunmak için kullanılan bir referans değişkeni olan bir anahtar kelimedir.
- Ebeveyn sınıf yöntemlerini çağırmak için kullanılır.
- super() ebeveyn sınıf yapıcısını çağırmak için kullanılır.
- super anahtar sözcüğü, üst sınıfın örnek değişkenlerine erişmek için kullanılır.

Yukarıdaki nedenlerden dolayı, super anahtar sözcüğü statik yöntem altında kullanılamaz, yalnızca statik olmayan bağlamda kullanılabilir.

Bu anahtar kelime: Aşağıdaki kullanımlara sahiptir,

- Geçerli sınıf örneği değişkenine atıfta bulunmak için kullanılabilir.
- Geçerli sınıf nesnesine (yapıcısı veya yöntemi çağrılan nesne) atıfta bulunmak için kullanılır. Bunu kullanarak geçerli nesnenin herhangi bir üyesine bir örnek yöntem veya yapıcı içinden başvurabilirsiniz.

Artık statik yöntemler nesnelerle değil sınıfla ilgili olduğundan, bu statik yöntemlerin içinde kullanılamaz.

Java'daki süper anahtar kelimeyle ilgili bazı noktalar yazar mısınız?

Bunlar java'da super anahtar sözcüğünün bazı kullanımlarıdır.

Süper anahtar sözcüğü, üst sınıf örnek değişkenlerine atıfta bulunmak için kullanılabilir.

Süper anahtar kelime, ebeveyn sınıfı yöntemini çağırmak için kullanılabilir.

Süper anahtar kelime, ebeveyn sınıfı yapıcısını çağırmak için kullanılabilir.

Super anahtar kelimesini kullanarak alt sınıftaki ebeveyn sınıfı değişkenlerine erişebilir miyiz?

Evet, Java super anahtar sözcüğünü kullanarak alt sınıftaki ebeveyn sınıfı değişkenine erişebiliriz.

Super anahtar kelimesini kullanarak alt sınıfta ebeveyn sınıf yöntemini çağırabilir miyiz?

Evet, super anahtar kelimesini kullanarak alt sınıfta ebeveyn sınıfı yöntemine erişebiliriz.

Super anahtar sözcüğünü kullanarak alt sınıf yapıcısında ebeveyn sınıf yapıcısını çağırabilir miyiz?

Evet, alt sınıf yapıcısında ebeveyn sınıf yapıcısını süper anahtar sözcüğünü kullanarak çağırabiliriz, ancak süper anahtar sözcüğü alt sınıf yapıcısında ilk ifade olmalıdır.

Yapıcıda hem "this" hem de "super" kelimesini kullanabilir miyiz?

Hayır, Java'da mümkün değildir, Java yapıcısında this ve super anahtar kelimelerini birlikte kullanamayız çünkü this ve super herhangi bir Java yapıcısında ilk ifade olmalıdır.

this() ve super() arasındaki fark nedir?

Java this() yapıcısı ile super() yapıcısı arasında bazı farklar vardır.

Java'da bu()

Geçerli sınıf yapıcısına erişmek için kullanılır.

Aynı sınıftan başka bir kurucunun içinde kullanılabilir.

Veri üyelerimiz ve yerelimiz aynı isimde olduğunda belirsizlik hatasını ortadan kaldırmak için kullanılabilir.

Örneğin: Belirsizlik hatasını kaldırın

Java süper()

Alt sınıftan ebeveyn sınıf yapıcısına erişmek için kullanılır.

Yalnızca alt sınıf yapıcısının içinde kullanılabilir.

Programlardan herhangi bir belirsizlik hatasını ortadan kaldırmaz.

Üst sınıf yöntemini çağırmak için bir alt sınıfın statik yönteminde süper anahtar sözcüğünü kullanabilir miyiz?

Hayır, super anahtar sözcüğünü statik yöntemlerde kullanamayız çünkü bu anahtar kelime doğrudan ana nesneye aittir ve static de sınıf düzeyine aittir.

18. This Keyword – Sınıf Değişkenlerine Erişim

Bu Java'da bir anahtar kelimedir. “This” anahtar kelimesi şununla kullanılabilir:

1. Geçerli nesnenin herhangi bir üyesine bir örnek yöntem veya yapıcı içinden başvurmak için kullanılabilir
2. “this” anahtar sözcüğü aynı Sınıftan başka bir Yöntemi çağırmak için Yöntemlerin içinde de kullanılabilir.
3. “this” anahtar sözcüğü kullanılabilir. Buna Açık Yapıcı Çağırma denir.

Sınıf değişkenlerine erişim:

Örnek:

```
paket FPPaket;
```

```
halk class ThisKeyword {
    statik int A ;
    statik int B ;
    halk geçersiz atama ( int c , dahili D ) {
        bir = c ;
        b = d ;
    }
    halk statik void main(String[] args ) {
        ThisKeyword tk = new ThisKeyword();
        teşekkürler . (5,6)' yı ata ;
        Sistem. out .println( "Bir değer " + a );
        Sistem. out .println( "B değeri " + b );
    }
}
```

Sonuç:

Bir değer 5'tir
B değeri 6

Yukarıdaki örnekte a ve b örnek değişkenlerine değer atamak için “atama(int c, int d)” yöntemini oluşturduk.

Yöntemi atama(int a, int b) olarak değiştirirsek program, değerleri yerel değişkenlere mi yoksa örnek değişkenlere mi ataması gerektiği konusunda kafa karıştırır.

Örnek:

```
paket FPPaket;
```

```
halk class ThisKeyword {
    statik int A ;
    statik int B ;
    halk geçersiz atama ( int ) a , int B ) {
        bir = bir ;
        b = b ;
    }
    halk statik void main(String[] args ) {
        ThisKeyword tk = new ThisKeyword();
        tk .atama(5,6);
        Sistem. out .println( "Bir değer " + a );
        Sistem. out .println( "B değeri " + b );
    }
}
```

```
}
```

Sonuç:

Bir değer 0'dır
B değeri 0

Bu nedenle, değerleri örnek değişkenlere atamak için bu anahtar kelimeyi kullanınız.

Örnek:

```
paket FPPaket;

halk class ThisKeyword {
    statik int A ;
    statik int B ;
    halk geçersiz atama ( int a , int B ) {
        Bu . bir = bir ;
        Bu . b = b ;
    }
    halk statik void main(String[] args ) {
        ThisKeyword tk = new ThisKeyword();
        tk.atama(5,6);
        Sistem. out .println( "Bir değer " + a );
        Sistem. out .println( "B değeri " + b );
    }
}
```

Sonuç:

Bir değer 5'tir
B değeri 6

18A. Bu Anahtar Kelime - Örnek Değişkenlerine Erişim

Örnek Değişkenlerine Erişim:

```
paket FPPaket;

halk class ThisExample {

    int bir = 10;

    halk statik void main(String[] args ) {

        ThisExample te = new ThisExample();
        te .method1();
    }

    halk geçersiz yöntem1() {
        int bir = 30;
        Sistem. out .println( " bir değer " + bu .a );
    }
}
```

18B. Bu Anahtar Kelime - Aynı Sınıftaki Başka Bir Yöntemden Bir Yönteme Erişmek

Aynı sınıftaki başka bir yöntemden bir yönteme erişmek:

```
paket FPPaket;

halk class ThisExample {

    int bir = 10;

    halk geçersiz yöntem1() {
        Sistem. out .println( "Yöntem 1'deyim" );
        bu .method2();
    }

    halk geçersiz yöntem2() {
        Sistem. out .println( "Ben yöntem 2'yim" );
    }

    halk statik void main(String[] args ) {

        ThisExample te = new ThisExample();
        te .method1();
    }
}
```

18C. Bu Anahtar Kelime - Aynı Sınıftaki Başka Bir Yapıcıdan Bir Yapıcıya Erişmek

Bir kurucuya aynı sınıftaki başka bir kurucudan erişme:

```
paket FPPaket;

halk class ThisExample {

    int bir = 10;

    Bu Örnek(){

        this ( "Subbus Selenyum Dersleri" );
        Sistem. out .println( "Normal Oluşturucu" );
    }

    ThisExample(Dize dizisi ){
        Sistem. out .println( "Parametrelili kurucu ve parametreleyici: " +
str );
    }

    halk statik void main(String[] args ) {

        ThisExample te = new ThisExample();
    }
}
```

```
}
```

18D. Java Bu Anahtar Kelime - Soruları

this() ve super() yapıcılarla nasıl kullanılır?

`this()` aynı sınıftan bir kurucuyu çağırmak için kullanılır. `super()`, bir üst sınıf yapıcığı çağırmak için kullanılır.

this() ve super() arasındaki farklar nelerdir?

`this()`, aynı sınıftaki bir kurucudan diğer bir kurucuya erişmek için kullanılırken, `super()` üst sınıf kurucuya erişmek için kullanılır. `this()` veya `super()` mevcutsa, yapıcındaki ilk ifade olmalıdır.

This anahtar sözcüğünü statik yöntemde kullanabilir miyiz?

Hayır. Statik yöntem sınıfa ait olduğundan This anahtar sözcüğünü statik yöntemde kullanamayız.

Statik yöneme erişmek için This anahtar sözcüğünü kullanabilir miyiz?

Evet. Çağırdığımız yöntem statik değilse anahtar kelimeyi kullanabiliriz.

Bu yöntemi main() yönteminde kullanabilir miyiz?

Hayır. Main() yöntemi statik olduğundan, this anahtar sözcüğünü main() yönteminde kullanamayız.

Bu anahtar kelimeye ilişkin yöntemi yapıcından çağırabilir miyiz?

Evet. Bu anahtar sözcüğü kullanarak yapıcından statik olmayan yöntemi çağırabiliriz.

Java'da "buna" bir referans atamak mümkün mü?

Hayır. "Buna" bir değer atayamayız çünkü o her zaman mevcut nesneyi gösterir ve Java'da her zaman son referanstır.

Eğer "this"i değiştirmeye veya referans atamaya çalışırsak derleme zamanı hatası verecektir.

Bu anahtar kelimenin kullanım alanları nelerdir?

hem örnek hem de yerel değişken adları aynıysa, örnek değişkene erişmek için bu kullanılmalıdır.

Bu anahtar kelimeyi yapıcının aşırı yüklenmesinde kullanabiliriz, bir kurucuyu diğerinden çağırmak için `buna` ihtiyacımız `var()`; ve `bu()`; çağrı yapıcının ilk ifadesi olmalıdır.

Bu, diğer statik olmayan yöntemleri `with in` yöntemlerden çağırmak için kullanılabilir.

Bunu yöntemin parametresi olarak geçirebilir miyiz?

Evet, bunu bir yöntemde parametre olarak geçirebiliriz

Bunu statik üyeleri yönlendirmek için kullanabilir miyiz?

Evet, bunu kullanarak bir sınıfın statik değişkenine erişmek mümkündür ancak bu önerilmez ve en iyi uygulamalara göre bu, statik olmayan referansta kullanılmalıdır.

Bunu statik bloklarda kullanmak mümkün mü?

Hayır, bu anahtar kelimenin statik blokta kullanılması mümkün değildir.

Bir yöntemden “bunu” döndürebilir miyiz?

Evet. Bunu geçerli sınıf nesnesi olarak döndürebiliriz.

```
paket FPPaket;

halk class ThisExample {

    int A ;

    halk geçersiz yöntem1( int A ) {
        Bu . bir = bir ;
    }

    halk int getA() {
        geri dönmek A ;
    }

    public ThisExample show() {
        geri dönmek Bu ;
    }

    halk statik void main(String[] args ) {

        ThisExample te = new ThisExample();
        te .method1(10);

        ThisExample te1 = new ThisExample();
        te1 = te .show();
        Sistem. out .println( te1 .getA());
    }

}
```

19. Java If Else Koşulları - Bölüm 1

Java If – Aksi takdirde:

- a) if deyimi
- b) iç içe if deyimi c) if-else deyimi d) if-else-if deyimi

eğer Açıklama:

Yapı:

```
Eğer(koşul){
    ifade;
}
```

Örnek:

```
paket FPPaket;

halk sınıf ifelse {
```

```

    halk statik void main(String[] args ) {
        int say11 = 5;
        int say12 = 3;

        if ( say11 > say12 ) {
            Sistem. out .println( "Doğru" );
        }
    }
}

```

İç içe If ifadesi:

Yapı:

```

Eğer(koşul){
    Eğer(koşul){
        ifade;
    }
}

```

Örnek:

```

paket FPPaket;

halk sınıf ifelse {

    halk statik void main(String[] args ) {
        int say11 = 5;
        int say12 = 3;
        int say13 = 2;
        if ( say11 > say12 ) {
            if ( say12 > say13 ) {
                Sistem. out .println( "Doğru" );
            }
        }
    }
}

```

if-else ifadesi:

```

if(koşul) {
    ifade;
}

başka {
    ifade;
}

```

Örnek:

```

paket FPPaket;

halk sınıf ifelse {

```

```

    halk statik void main(String[] args ) {
        int say11 = 5;
        int say12 = 6;

        if ( say11 > say12 ) {
            Sistem. out .println( "Doğru" );
        }
        başka {
            Sistem. out .println( "Yanlış" );
        }
    }
}

```

if-else-if ifadesi:

```

if(koşul) {
    ifade;
}

başka eğer {
    ifade;
}

```

Örnek:

paket FPPaket;

```

halk sınıf ifelse {

    halk statik void main(String[] args ) {
        int say11 = 5;
        int say12 = 6;

        if ( say11 == say12 ) {
            Sistem. out .println( "say11, say12'ye eşittir" );
        }
        başka if ( say11 > say12 ){
            Sistem. out .println( "say11, say12'den büyük" );
        }
        başka {
            Sistem. out .println( "say11, say12'den küçüktür" );
        }
    }
}

```

19A. Java If Else - Farklı Karşılaştırma Operatörlerini Kullanma

Farklı Karşılaştırma Operatörlerini Kullanma:

Farklı karşılaştırma operatörleri

==, !=, >, <, >=, <=

Yukarıda == operatörünü zaten görmüştük.

!= ile örnek:

```

halk sınıf Java Örnekleri {

    halk statik void main(Dize [] args ){
        int bir = 5;
        int b = 8;

        eğer ( a != b ) {
            Sistem. out .println( "a ve b eşit değil" );
        }
        başka {
            Sistem. out .println( "her iki değer de eşit" );
        }

    }

}

```

Sonuç: a ve b eşit değil

> ile örnek:

```

halk sınıf Java Örnekleri {

    halk statik void main(Dize [] args ){
        int bir = 5;
        int b = 8;

        eğer ( a > b ) {
            Sistem. out .println( "a, b'den büyüktür" );
        }
        başka {
            Sistem. out .println( "a, b'den küçüktür" );
        }

    }

}

```

Sonuç: a, b'den küçüktür

< ile örnek:

```

halk sınıf Java Örnekleri {

    halk statik void main(Dize [] args ){
        int bir = 5;
        int b = 8;

        eğer ( a < b ) {
            Sistem. out .println( "a, b'den küçüktür" );
        }
        başka {
            Sistem. out .println( "a, b'den büyüktür" );
        }

    }

}

```



```
}
```

Sonuç: a, b'den küçüktür

>= ile örnek:

```
halk sınıf Java Örnekleri {

    halk statik void main(Dize [] args ){
        int bir = 9;
        int b = 8;

        eğer ( a >= b ) {
            Sistem. out .println( "a, b'den büyük veya ona eşittir" );
        }
        başka {
            Sistem. out .println( "a, b'den küçüktür" );
        }
    }

}
```

Sonuç: a, b'den büyük veya ona eşittir

<= ile örnek:

```
halk sınıf Java Örnekleri {

    halk statik void main(Dize [] args ){
        int bir = 5;
        int b = 8;

        eğer ( bir <= b ) {
            Sistem. out .println( "a, b'den küçük veya eşittir" );
        }
        başka {
            Sistem. out .println( "a, b'den büyüktür" );
        }
    }

}
```

Sonuç: a, b'den küçük veya ona eşittir

19B. Java If Else - Farklı Mantıksal Operatörler Kullanma

Farklı Mantıksal Operatörlerin Kullanımı:

Sahip olduğumuz farklı mantıksal operatörler şunlardır:

&& (Mantıksal VE), || (Mantıksal VEYA) ve ! (Olumsuzlama)

&& (Mantıksal VE) kullanma:

&& (Mantıksal AND) iki veya daha fazla koşulu doğrulamamız gerekiyorsa ve tüm koşulların doğru olmasını istiyorsak kullanılır. Tüm koşullar doğruysa yalnızca koşulun altındaki kod bloğu yürütülür.

İki Koşullu Örnek 1:

```

halk sınıf Java Örnekleri {

    halk statik void main(Dize [] args ){
        int bir = 5;
        int b = 8;
        int c = 10;

        if ( a < b && a < c ) {
            Sistem. out .println( "a, b ve c'den küçüktür" );
        }
        başka {
            Sistem. out .println( "a, b veya c'den büyük olabilir" );
        }

    }

}

```

Üç koşullu Örnek 2:

```

halk sınıf Java Örnekleri {

    halk statik void main(Dize [] args ){
        int bir = 5;
        int b = 8;
        int c = 10;
        int d = 20;

        if ( a < b && a < c && a < d ) {
            Sistem. out .println( "a, b ve c ve d'den küçüktür" );
        }
        başka {
            Sistem. out .println( "a, b'den veya c'den veya d'den büyük
olabilir" );
        }

    }

}

```

|| kullanma (Mantıksal VEYA):

|| (Mantıksal VEYA) iki veya daha fazla koşulu doğrulamamız gerekiyorsa ve koşullardan en az birinin doğru olmasını istiyorsak kullanılır. Koşullardan en az biri doğruysa yalnızca koşulun altındaki kod bloğu yürütülür.

İki koşullu Örnek 1:

```

halk sınıf Java Örnekleri {

    halk statik void main(Dize [] args ){
        int bir = 5;
        int b = 8;
        int c = 10;

        eğer ( a < b || a < c ) {
            Sistem. out .println( "a, b veya c'nin birinden küçüktür" );
        }

    }

}

```

```

        başka {
            Sistem. out .println( "a, b veya c'den büyüktür" );
        }
    }
}

```

Üç koşullu Örnek 2:

```

halk sınıf Java Örnekleri {

    halk statik void main(Dize [] args ){
        int bir = 21;
        int b = 8;
        int c = 10;
        int d = 20;

        if ( a < b || a < c || a < d ) {
            Sistem. out .println( "a, b ve c ve d'den küçüktür" );
        }
        başka {
            Sistem. out .println( "a, b'den ve c ve d'den büyüktür" );
        }

    }

}

```

Kullanarak! (Olumsuz):

```

halk sınıf Java Örnekleri {

    halk statik void main(Dize [] args ){
        boolean test = doğru ;

        eğer (! test ) {
            Sistem. out .println( "test yanlış" );
        }
        başka {
            Sistem. out .println( "test doğru" );
        }

    }

}

```

19C. Java Switch Case İfadeleri

Anahtar – Durum Açıklaması:

Switch case, birden fazla seçeneğimiz olduğunda ve her biri için ayrı bir işlem yapmamız gerektiğinde kullanılır.

Yapı:

```

Anahtar(ifade)
{
    Durum sabiti:
        İfadeler;
        kırmak;
    Durum sabiti:
        İfadeler;
        kırmak;
    Varsayılan:
        İfadeler;
        kırmak;
}

```

Örnek:

```

paket FPPaket;

halk sınıf java örnekleri {

    halk statik void main(String[] args ) {
        int sayı1 = 2;
        anahtar ( sayı1 ) {
            dava 1:
                Sistem. out .println( "Ben 1'im" );
                kırmak ;
            durum 2:
                Sistem. out .println( "Ben 2'yim" );
                kırmak ;
            durum 3:
                Sistem. out .println( "Ben 3 yaşımdayım" );
                kırmak ;
            varsayılan :
                Sistem. out .println( "Ben 1, 2 veya 3 değilim" );
        }
    }
}

```

Sonuç: 2 yaşımdayım

Daha önce Java, switch case ifadelerinde yalnızca tam sayıları destekliyordu. Ancak artık dizeler de destekleniyor.

```

halk sınıf Java Örnekleri {

    halk statik void main( String [] args ){

        Sicim bir = "iki" ;

        anahtar ( a ) {
            dava "bir" :
                Sistem. out .println( "Ben 1'im" );
                kırmak ;
            dava "iki" :
                Sistem. out .println( "Ben 2'yim" );
                kırmak ;
        }
    }
}

```

```

dava "üç" :
    Sistem. out .println( "Ben 3 yaşımdayım" );
    kırmak ;
dava "dört" :
    Sistem. out .println( "Ben 4 yaşımdayım" );
    kırmak ;
dava "beş" :
    Sistem. out .println( "Ben 5 yaşımdayım" );
    kırmak ;
dava "altı" :
    Sistem. out .println( "Ben 6 yaşımdayım" );
    kırmak ;
varsayılan :
    Sistem. out .println( "Ben 1 veya 2 veya 3 veya 4 veya
5 veya 6 değilim" );
    }
}
}

```

Sonuç: 3 yaşımdayım

19D. Java Switch Case İfadeleri - Aynı Kodu Yürüten Çeşitli Değerler Anahtar – Durum Açıklaması:

Switch Case, birden fazla değer aynı kodu çalıştırdığında da kullanılabilir.

```

halk sınıf Java Örnekleri {
    halk statik void main(Dize [] args ){
        int ay = 4;
        geçiş ( ay ) {
            dava 1:
            durum 3:
            durum 5:
            durum 7:
            durum 8:
            durum 10:
            durum 12:
                Sistem. out .println( "Gün sayısı " +31);
                kırmak ;
            durum 2:
                Sistem. out .println( "Gün sayısı " +28);
                kırmak ;
            varsayılan :
                Sistem. out .println( "Gün sayısı " +30);
        }
    }
}

```

19E. Java Switch Case İfadeleri - Break İfadesi Olmadan

Anahtar – Durum Açıklaması:

Switch – Case'deki Break ifadesi:

Her case ifadesinden sonra break ifadesini sağlamanız gerekir, aksi takdirde diğer tüm case ifadeleri de yürütülecektir.

Örneğin aşağıdaki programda tüm break ifadelerini çıkardım.

```

halk sınıf Java Örnekleri {

    halk statik void main(Dize [] args ){

        String a = "Üç" ;

        anahtar ( a ) {
            dava "bir" :
                Sistem. out .println( "Ben 1'im" );

            dava "iki" :
                Sistem. out .println( "Ben 2'yim" );

            dava "üç" :
                Sistem. out .println( "Ben 3 yaşımdayım" );

            dava "dört" :
                Sistem. out .println( "Ben 4 yaşımdayım" );

            dava "beş" :
                Sistem. out .println( "Ben 5 yaşımdayım" );

            dava "altı" :
                Sistem. out .println( "Ben 6 yaşımdayım" );

            varsayılan :
                Sistem. out .println( "Ben 1 veya 2 veya 3 veya 4 veya
5 veya 6 değilim" );
        }

    }

}

```

“Üç” durumundaki koşul karşılandığında, yalnızca “üç” durumunu değil, bundan sonraki her şeyi yürütecektir.

Sonuç:

```

ben 3 yaşımdayım
ben 4 yaşımdayım
ben 5 yaşımdayım

```

ben 6 yaşımdayım
Ben 1 ya da 2 ya da 3 ya da 4 ya da 5 ya da 6 değilim

19F. Java If Else ve Switch Case - Soruları

Bir if ifadesi ile bir switch ifadesi arasındaki fark nedir?

Değer aralıklarını veya birden fazla koşulu kontrol etmemiz gerektiğinde *if-then-else* ifadesi tercih edilir.

switch *ifadesi* daha uygundur. İlişkisel veya mantıksal operatörleri kullanmanız gerekiyorsa, anahtar durumunu kullanamazsınız.

If-then ve *if-then-else* ifadelerini açıklayınız . Koşul olarak ne tür ifadeler kullanılabilir?

Her iki ifade de programımıza, yalnızca belirli bir koşulun *true* olarak değerlendirilmesi durumunda içlerindeki kodu çalıştırmasını söyler . Ancak *if-then-else* ifadesi, *if* yan tümcesinin *false* olarak değerlendirilmesi durumunda ikincil bir yürütme yolu sağlar .

Switch ifadesini açıklayın . *Switch* cümlesinde hangi nesne türleri kullanılabilir ?

Anahtar, bir değişkenin değerine bağlı olarak birden fazla yürütme yolunun seçilmesine olanak tanır.

case veya *default* ile etiketlenir ; *switch* deyimi her *case* ifadesini bir eşleşme açısından değerlendirir ve bir *break* ifadesi bulunana kadar eşleşen etiketi izleyen tüm ifadeleri yürütür . Bir eşleşme bulamazsa bunun yerine varsayılan blok yürütülür

Bir *switch*'in *case* cümlecğine *break* ifadesini koymayı unutsak ne olur ?

break ifadesi bulana kadar , bu etiketler ifadenin değeriyle eşleşmese bile tüm durum etiketlerinin yürütülmesine devam edeceği anlamına gelir .

Switch deyimini *String*'lerle kullanabilir miyiz?

Java 7'den önce *Switch*'te yalnızca *int* değerlerini ve *enum* sabitlerini kullanabiliyorduk. Java 7'den başlayarak *Switch* ifadesinde dizeleri kullanabiliriz. Java 7'den önceki *switch* ifadesinde dizeler kullanırsak, "yalnızca *int* ve *enum* sabitlerine izin verilir" derleme zamanı hatası alırız.

20. Selenyum - Döngüler ve Her Biri İçin Döngüler İçin Java

Döngüler İçin ve Her Döngü İçin Java:

Programlama dillerinde döngü, bir dizi ifadenin tekrar tekrar yürütülmesini kolaylaştıran bir özelliktir.

Sözdizimi:

```
for(başlatma; koşul; artırma/azaltma)
```

```
{
```

```
    ifadeler;
```

```
}
```

Örnek:

```

paket FPPaket;

halk sınıf ifelse {

    halk statik void main(String[] args ) {

        için ( int ben =0; ben <10; ben ++ ) {
            Sistem. out .println( i );
        }
        Sistem. out .println( "Tamamlandı" );
    }
}

```

For Döngüleri içindeki Döngüler İçin:

Ayrıca diğer for döngülerinin içinde for döngüleri de oluşturabiliriz.

```

paket paket1;

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        için ( int ben =0; ben <10; ben ++ ) {
            Sistem. out .println( "Döngünün dışındayım" );
            için ( int j =0; j <5; j ++ ) {
                Sistem. out .println( " -- Döngünün içindeyim" );
            }
        }
    }
}

```

Her Döngü İçin:

Her döngü için esas olarak Diziler ve Koleksiyonlar için kullanılır.

Örneğin bir dizimiz varsa dizinin elemanlarına aşağıdaki yolu kullanarak erişebiliriz.

```

paket paket1;

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        String[] a = { "pazar" , "pazartesi" , "salı" , "çarşamba" ,
"perşembe" , "cuma" , "cumartesi" };

        için ( int ben =0; ben < a . uzunluk ; ben ++ ) {

            Sistem. out .println( a [ i ] );

        }
    }
}

```

İçinde dizi uzunluğu olan normal bir for döngüsü kullanabilir ve dizi boyunca döngü yapabiliriz.

Aynı şey for her döngü kullanılarak da başarılabilir.

```
paket paket1;

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        String[] a = { "pazar" , "pazartesi" , "salı" , "çarşamba" ,
        "perşembe" , "cuma" , "cumartesi" };

        for (Dize öğesi : a ) {

            Sistem. out .println( öğe );

        }

    }

}
```

Yukarıdaki programda kullandığımız

for (Dize öğesi: a)

Burada dizinin her değeri döngüye girecek öğede saklanacaktır.

Her döngü için sözdizimi şöyledir

For (dizi/koleksiyon değişkeninin veri türü: dizi/koleksiyon)

Her döngü için dizideki öğelerin toplamını bulan başka bir program:

```
paket paket1;

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        int [] a = {1,2,3,4,5,6,7,8,9,10};

        int toplam = 0;

        için ( int öğe : a ) {

            toplam = toplam + öğe ;

        }

        Sistem. out .println( "Dizi değerlerinin toplamı: " + toplam );

    }

}
```

20A. Döngüler İçin Java - Kesme ve Etiketli Kesme

Java For Loop – Break ve Etiketli Break İfadeleri:

Ara İfadeleri:

Mevcut döngüden çıkmak istiyorsanız break ifadesini kullanabilirsiniz. Normalde break ifadeleri koşullarla birlikte kullanılır.

Örnek:

```
paket paket1;

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        için ( int ben =0; ben <10; ben ++ ) {

            Sistem. out .println( "i değeri " + i );

            için ( int j =50; j <53; j ++ ) {
                eğer ( ben == 5) {
                    kırmak ;
                }
                Sistem. out .println( " -- j değeri " + j );
            }

        }

    }

}
```

Etiketli Ara İfadeleri:

Etiketlenmemiş bir *break* ifadesi en içteki *switch for* , *while* veya *do-while* ifadesini sonlandırırken, etiketli bir *break* dıştaki bir ifadenin yürütülmesini sonlandırır .

```
paket paket1;

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

dış : for ( int ben =0; ben <10; ben ++ ) {

        Sistem. out .println( "i değeri " + i );

        için ( int j =50; j <53; j ++ ) {
            eğer ( ben == 5) {
                kırmak dış ;
            }
            Sistem. out .println( " -- j değeri " + j );
        }

    }

}
```

```
}
```

20B. Java For Loops - Devam Et ve Etiketli Devam Et

Java For Loop - Continue ve Etiketli Continue İfadeleri:

Açıklamalara Devam Et:

Continue ifadesi, en içteki for, while ve do-while döngülerindeki geçerli yinelemenin sonuna atlar.

Eski:

```
paket paket1;

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        için ( int ben =0; ben <10; ben ++ ) {

            Sistem. out .println( "i değeri " + i );

            için ( int j =50; j <53; j ++ ) {
                eğer ( ben == 5) {
                    devam etmek ;
                }
                Sistem. out .println( " -- j değeri " + j );
            }

        }

    }

}
```

Etiketli Devam İfadeleri:

Etiketlenmemiş bir *devam ifadesi*, en içteki for , while veya do-while döngüsündeki geçerli yinelemenin sonuna atlarken, etiketli bir *devam ifadesi*, verilen etiketle işaretlenmiş bir dış döngüye atlar.

```
paket paket1;

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        dış : for ( int ben =0; ben <10; ben ++ ) {

            Sistem. out .println( "i değeri " + i );

            için ( int j =50; j <53; j ++ ) {
                eğer ( j == 51) {
                    devam edin ;
                }
                Sistem. out .println( " -- j değeri " + j );
            }

        }

    }

}
```

```

    }
}
}

```

20C. Java While Döngüleri ve İç While Döngüleri

While Döngüleri ve İç While Döngüleri:

Yapı:

```

while(koşul){
    İfadeler;
}

```

Örnek:

```

paket FPPaket;

halk sınıf ifelse {

    halk statik void main(String[] args ) {
        int ben = 0;
        while ( i <10) {
            Sistem. out .println( i );
            ben ++;
        }
        Sistem. out .println( "Tamamlandı" );
    }
}

```

İç While Döngüleri:

```

paket paket1;

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        int j = 0;

        while ( j <3) {

            Sistem. out .println( " j değeri " + j );

            int k = 0;

            while ( k <3) {

                Sistem. out .println( " -- k değeri " + k );

                k ++;

            }

        }

    }
}

```

```

        j ++;
    }

}

```

20D. Java While Döngüleri - Break ve Etiketli Break

While Döngüleri – Break ve Etiketli Break İfadeleri:

Ara Bildirimi:

Break ifadeleri yürütme sırasında mevcut döngüyü kırmak için kullanılır.

```

paket paket1;

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        int j = 0;

        while ( j <3) {

            Sistem. out .println( " j değeri " + j );

            int k = 0;

            while ( k <3) {

                Sistem. out .println( " -- k değeri " + k );

                eğer ( k ==1)
                    kırmak ;

                k ++;

            }

            j ++;

        }

    }

}

```

Etiketli Ara İfadeleri:

Etiketli break ifadeleri dış döngüyü kırmak için kullanılır.

```

paket paket1;

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

```

```

    int j = 0;
    dış: while ( j <3) {
        Sistem. out .println( " j değeri " + j );

        int k = 0;

        while ( k <3) {
            Sistem. out .println( " -- k değeri " + k );

            eğer ( k ==1)
                kırılma ;

            k ++;
        }
        j ++;
    }
}

```

20E. Java While Döngüleri - Devam Et ve Etiketli Devam Et

While Döngüsü – Continue ve Etiketli Continue İfadeleri:

Açıklamaya Devam Et:

```

paket paket1;

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        int j = 0;

        while ( j <3) {

            Sistem. out .println( " j değeri " + j );

            int k = 0;

            while ( k <3) {

                eğer ( k >=1) {
                    k ++;
                    devam etmek ;
                }
                Sistem. out .println( " -- k değeri " + k );
                k ++;
            }
        }
    }
}

```

```

        j ++;
    }
}
}

```

Etiketli Devam İfadesi:

```

paket paket1;

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        int j = 0;

        dış: while ( j <3) {

            Sistem. out .println( " j değeri " + j );

            int k = 0;

            while ( k <3) {

                Sistem. out .println( " -- k değeri " + k );

                eğer ( k >=0) {
                    j ++;
                    devam edin ;
                }
                //
                //k++;
            }

            //j++;
        }

    }

}

```

20F. Java Do While Döngüleri ve İç Do While Döngüleri

Do – While Döngüleri ve İç Do – While Döngüleri:

Yapı:

```

Yapmak {
    İfadeler;

```

```

} while(Koşul)

```

Örnek:

```

paket FPPaket;

```

```

halk sınıf ifelse {

    halk statik void main(String[] args ) {
        int ben = 12;
        Yapmak {
            Sistem. out .println( i );
            ben ++;
        } while ( i <10);
        Sistem. out .println( "Tamamlandı" );
    }
}

```

İç do – while döngüleri:

```
paket paket1;
```

```

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        int j = 0;

        Yapmak {

            Sistem. out .println( " j değeri " + j );
            int k = 0;
            Yapmak {
                Sistem. out .println( "-- k değeri " + k );

                k ++;
            } while ( k <3);
            j ++;
        } while ( j <3);

    }

}

```

20G. Java Do While Döngüleri - Break ve Etiketli Break

Do – While Döngüleri – Break ve Etiketli Break İfadeleri:

Ara Bildirimi:

```
paket paket1;
```

```

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        int j = 0;

        Yapmak {

            Sistem. out .println( " j değeri " + j );

```



```

        int k = 0;
        Yapmak {
            Sistem. out .println( " -- k değeri " + k );
            k ++;
            eğer ( k ==1)
                kırmak ;
        } while ( k <3);
        j ++;
    } while ( j <3);
}
}

```

Etiketli Mola İfadesi:

```

paket paket1;

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        int j = 0;
        dış : yap {

            Sistem. out .println( " j değeri " + j );
            int k = 0;
            Yapmak {
                Sistem. out .println( " -- k değeri " + k );
                k ++;
                eğer ( k ==1)
                    kırılma ;
            } while ( k <3);
            j ++;
        } while ( j <3) ;

    }

}

```

20 saat. Java Do While Döngüleri - Devam Et ve Etiketli Devam Et

Do – While Döngüleri – Continue ve Etiketli Continue İfadeleri:

Açıklamaya Devam Et:

```

paket paket1;

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

```

```

    int j = 0;

    Yapmak {

        Sistem. out .println( " j değeri " + j );
        int k = 0;
        Yapmak {
            eğer ( k >=1) {
                k ++;
                devam etmek ;
            }
            Sistem. out .println( " -- k değeri " + k );
            k ++;
        } while ( k <3);
        j ++;
    } while ( j <3);

}
}

```

Etiketli Devam İfadesi:

```

paket paket1;

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        int j = 0;

        dış: yap {

            Sistem. out .println( " j değeri " + j );
            int k = 0;
            Yapmak {
                eğer ( k >=1) {
                    k ++;
                    j ++;
                    devam edin ;
                }
                Sistem. out .println( " -- k değeri " + k );
                k ++;
            } while ( k <3);
            j ++;
        } while ( j <3);

    }

}
}

```

201. Java Döngüleri - While Döngüsü ve Do While Döngüsü Arasındaki Fark

While döngüsü ile do-while döngüsü arasındaki fark nedir?

While döngüsü, koşul karşılanmadığı takdirde döngü içindeki kodu çalıştırmaz. Ancak do-while döngüsü bunu en az bir kez çalıştırır.

Örnek:

döngü sırasında:

```
paket paket1;
```

```
halk sınıf Java Örnekleri {
```

```
    halk statik void main(String[] args ) {
```

```
        int j = 5;
```

```
        while ( j < 3) {
```

```
            Sistem. out .println( "j'nin değeri " + j );
```

```
        }
```

```
    }
```

```
}
```

Yukarıdaki while döngüsünde j'nin değeri 3'ten büyüktür. Dolayısıyla hiçbir şey yazdırmaz.

do-while döngüsü:

```
paket paket1;
```

```
halk sınıf Java Örnekleri {
```

```
    halk statik void main(String[] args ) {
```

```
        int j = 5;
```

```
        Yapmak {
```

```
            Sistem. out .println( "j'nin değeri " + j );
```

```
        } while ( j < 3);
```

```
    }
```

```
}
```

Yukarıdaki do-while döngüsünde j'nin değeri 3'ten büyük olsa bile altta belirtilen koşul nedeniyle yine de bir kez çalıştırılır.

20J. Java Döngüleri - Soruları

Java hangi tür döngüleri destekler?

Java üç farklı döngü türü sunar: for, while ve do-while.

For döngüsü, bir dizi değer üzerinde yinleme yapmanın bir yolunu sağlar. Bir görevin kaç kez tekrarlanacağını önceden bildiğimizde bu çok kullanışlı olur.

Bir *while* döngüsü, *belirli bir koşul doğruyken* bir ifade bloğunu çalıştırabilir .

Do -*while*, *Boole* ifadesinin değerlendirmesinin döngünün en altında olduğu *while* ifadesinin bir varyasyonudur . Bu, kodun en az bir kez çalıştırılacağını garanti eder.

Geliştirilmiş döngü nedir?

Yinelenabilir arayüzü uygulayan herhangi bir nesnenin tüm öğelerini yinlemek üzere tasarlanmış *for* ifadesinin başka bir sözdizimidir .

Sözdizimi:

```
for(veri türü değişkeni : dizi/koleksiyon){
    //buraya kod yaz
}
```

Break ifadesi ile devam ifadesi arasındaki fark nedir?

Bir break ifadesi, geçerli olduğu ifadenin (switch, for, do veya while) sonlandırılmasıyla sonuçlanır. Devam ifadesi, geçerli döngü yinlemesini sonlandırmak ve kontrolü döngü ifadesine döndürmek için kullanılır.

Bir döngüden beklendiği gibi nasıl çıkabilirsiniz?

Break deyimini kullanma.

Etiketli ve etiketsiz break ifadesi arasındaki fark nedir?

Etiketlenmemiş bir *break* ifadesi en içteki *switch for* , *while* veya *do-while* ifadesini sonlandırırken, etiketli bir *break* dıştaki bir ifadenin yürütülmesini sonlandırır .

Etiketsiz ve etiketli devam ifadesi arasındaki fark nedir?

Etiketlenmemiş bir *devam ifadesi*, en içteki *for* , *while* veya *do-while* döngüsündeki geçerli yinlemenin sonuna atlarken, etiketli bir *devam ifadesi* , verilen etiketle işaretlenmiş bir dış döngüye atlar.

Bir for ifadesi süresiz olarak döngü yapabilir mi?

Evet, for ifadesi süresiz olarak döngüye girebilir. Örneğin, aşağıdakileri göz önünde bulundurun:

```
for(;;);
```

```
paket paket1;
```

```

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        için (;;)
        {
            Sistem. out .println( "merhaba" );
            Sistem. out .println( "nasılsın?" );
        }

    }

}

```

Yukarıdakiler sürekli olarak yinelenecek ve ifadeleri yazdıracaktır.

21. Java String'leri - String nedir ve nasıl çalışırlar?

Teller:

Bir dize bir karakter dizisidir (Bu bir karakter dizisidir).

Dizeler temel olarak üç şekilde oluşturulabilir.

1. Karakter Dizisini Kullanma
Eski:
karakter ch = {'H','e','l','l','o'};
String str = new String(ch);
2. Dize Değişmezini Kullanma
String str1 = "Merhaba";
3. Yeni anahtar kelime kullanma
String str2 = new String("Merhaba");

Örnek:

```
paket paket1;
```

```

halk sınıf Java Örnekleri {

    halk statik void main(String[] args ) {

        //İlk yol
        char [] ch = { 'H' , 'e' , 'l' , 'l' , 'o' , 'l' };
        String str1 = new String( ch );

        //İkinci yol
        String str2 = "Merhaba2" ;

        //Üçüncü yol
        String str3 = new String( "Merhaba3" );

        Sistem. out .println( str1 );

        Sistem. out .println( str2 );

        Sistem. out .println( str3 );

    }

}

```

Dize değişmezi ile bir dize ve new anahtar sözcüğüyle dizeler oluşturduğumuzda fark nedir?

Dize değişmezi ile bir dize oluşturduğumuzda, ilk olarak bellekte aynı değere sahip bir dizinin mevcut olup olmadığı araştırılır. Mevcut bir dize varsa, yalnızca o belleğe işaret eder.

New anahtar kelimesiyle bir dize oluşturduğumuzda bu, bir dize nesnesi oluşturduğumuz anlamına gelir. String değerleri aynı olsa bile stringler için her zaman yeni bir hafıza alanı oluşturur ve string değerlerini orada saklar.

Dizeler değişmez

Dizeler değişmezdir, yani değiştirilemezler. Ancak yeni dizelere atanabilirler.

Bu, bir dize oluşturduğunuzda, nesne çöp toplanana kadar bellekte orada kalacağı anlamına gelir. Bellekteki dizinin değerini değiştiremezsiniz. Ancak onu yeni bir değere atamaya çalıştığınızda, o yeni değer yeni bir hafıza alanında yaratılacaktır. Ancak eski dize herhangi bir referans olmaksızın orada kalır.

21A. Java Dizeleri - Dize Yöntemleri

Farklı Dize Yöntemleri:

Bir dizinin uzunluğu – `uzunluk()`

Karakterin veya bir alt dizinin dizini alma – `indexOf(Character or substring)`

Karakteri belirli bir pozisyona getirmek – `charAt(index)`

Bir dizinin bir alt dize içerip içermediğini kontrol etme – `include()`

Bir dizinin bir şeyle bitip bitmediğini kontrol etme – `endsWith()`

Bir dizedeki alt dizeleri değiştirme – `Değiştir, DeğiştirTüm, DeğiştirFirst`

Bir dizeyi küçük harfe dönüştürme – `toLowerCase()`

Bir dizeyi Büyük Harfe dönüştürme – `toUpperCase()`

paket FPPaket;

halk sınıf StringsDemo {

halk statik void main(String[] args) {

 String strtest1 = "merhaba merhaba" ;

 //Dizinin uzunluğu

 Sistem. **out** .println("Dizinin uzunluğu " + strtest1 .length());

 //Bir karakterin veya alt dizinin dizini

 Sistem. **out** .println("1'nin indeksi " + strtest1 .indexOf("1"));

 Sistem. **out** .println("Merhaba dizini: " + strtest1 .indexOf("merhaba"));

 //Belirli bir indeksteki karakter

 Sistem. **out** .println("3. dizindeki karakter " + strtest1 .charAt(3));

 //Bir dizinin alt dizge içerip içermediğini kontrol ediyoruz

 Sistem. **out** .println("Dize merhaba içeriyor mu?" + strtest1 .contains("merhaba"));

```

        //Bir stringin bir şeyle bitip bitmediğini kontrol ediyoruz
        Sistem. out .println(( "Dize i ile bitiyor mu? " + strtest1
        .endsWith( "i" )));
        Sistem. out .println(( "Dize hi ile bitiyor mu? " + strtest1
        .endsWith( "hi" )));

        //Değiştir, Tümünü Değiştir, İlkini Değiştir
        String strtest2 = "merhaba merhaba merhaba merhaba" ;
        Sistem. out .println( "Merhaba, merhaba ile değiştiriliyor: " +
        strtest2 .replace( "merhaba" , "merhaba" ));
        Sistem. out .println( "Merhaba, merhaba ile değiştiriliyor: " +
        strtest2 .replaceFirst( "merhaba" , "merhaba" ));
        Sistem. out .println( "Merhaba, merhaba ile değiştiriliyor: " +
        strtest2 .replaceAll( "merhaba" , "merhaba" ));

        // küçük harfe ve büyük harfe
        String strtest3 = "Merhaba Merhaba" ;
        Sistem. out .println( "Tüm Büyük Harfler: " + strtest3
        .toUpperCase());
        Sistem. out .println( "Tüm Küçük Harfler: " + strtest3
        .toLowerCase());
    }
}

```

Bir String'i Tamsayıya Dönüştürme:

Bir dizeyi tam sayıya dönüştürmek için bunun bir sayı dizisi olması gerekir.

```
Integer.parseInt(string);
```

Örnek:

```

paket FPPaket;

halk sınıf StringsDemo {

    halk statik void main(String[] args ) {

        String strtest1 = "53" ;
        // int k = strtest1 * 100;
        int i = Tamsayı. parseInt ( strtest1 );
        int j = i *100;
        Sistem. out .println( "j değeri " + j );

    }
}

```

NumberFormatException:

Sayı olmayan bir dizeyi tam sayıya dönüştürmeye çalışırsanız bu istisnaya karşılaşırsınız.

Java'daki alt dizeler:

Java'da substring() yöntemini kullanarak bir dizinin alt dizesini alabilirsiniz.

Bu yöntem için sadece başlangıç indeksini iletirseniz sağladığınız indeksten başlayarak stringin sonuna kadar bir alt string elde edersiniz.

Bu yöntem için başlangıç indeksini ve bitiş indeksini iletirseniz, başlangıç indeksinden bitiş indeksine kadar, bitiş indeksini içermeyen bir alt dize elde edersiniz.

```

halk sınıf StringsEx {

    halk statik void main(String[] args ) {

        String str = new String( "Selenium Dersleri" );
        Sistem. out .println( str .substring(9));
        Sistem. out .println( str .substring(9,14));

    }

}

```

21B. Java Dizeleri - İki Dizeyi Karşılaştırma ve Birleştirme

İki String'in karşılaştırılması:

Dizeler equals yöntemiyle karşılaştırılır.

```
Str1.equals(Str2);
```

== diğerlerini karşılaştırmak için kullanılır.

Örneğin tam sayı değerleri == yöntemiyle karşılaştırılır.

Dizeleri karşılaştırmak için == kullanırsanız, referans noktalarını karşılaştırır ancak gerçek değerleri karşılaştırmaz.

```
paket FPPaket;
```

```

halk sınıf StringsDemo {

    halk statik void main(String[] args ) {
        //Test 1
        String strtest1 = "merhaba" ;
        String strtest2 = "merhaba" ;
        if ( strtest1 == strtest2 ) {
            Sistem. out .println( "Test1: Her ikisi de eşittir" );
        }
        başka {
            Sistem. out .println( "Test1: Her ikisi de eşit değil" );
        }

        //Test 2
        String strtest3 = new String( "merhaba" );
        String strtest4 = new String( "merhaba" );
        if ( strtest3 == strtest4 ) {
            Sistem. out .println( "Test2: Her ikisi de eşittir" );
        }
        başka {
            Sistem. out .println( "Test2: İki de eşit değil" );
        }

        //Test 3
    }
}

```



```

String strtest5 = new String( "merhaba" );
String strtest6 = new String( "merhaba" );
if ( strtest5 .equals( strtest6 )) {
    Sistem. out .println( "Test3: Her ikisi de eşittir" );
}
başka {
    Sistem. out .println( "Test3: Her ikisi de eşit değil" );
}
}
}

```

İki diziye birleştirme:

İki stringi birleştirmek için + operatörünü kullanırız.

paket FPPaket;

```

halk sınıf StringsDemo {

    halk statik void main(String[] args ) {

        String strtest1 = "merhaba" ;
        String strtest2 = "merhaba" ;

        String strtest3 = strtest1 + strtest2 ;
        Sistem. out .println( strtest3 );

        String strtest4 = strtest1 .concat( strtest2 );
        Sistem. out .println( strtest4 );

    }
}

```

21C. Java Strings - Bir String'in diğer String'in Alt Dizisi olup olmadığını kontrol etme

Bir String'in Diğer String'in Alt Dizisi olup olmadığının kontrol edilmesi:

Bir stringin diğer stringin alt stringi olup olmadığını kontrol etmek için aşağıdaki iki metodu kullanabiliriz.

1. İçerir()
2. İndeksi()

Örnek:

paket alt dizisi;

```

halk sınıf SubstringÖrnek {
    static String str1 = "subbu" ;
    static String str2 = "subbu selenyum eğitimleri" ;

    halk statik void main(String[] args ) {
        if ( str2 .indexOf( str1 ) != -1) {
            Sistem. out .println( "1. Yöntem: str2, str1'in alt dizisidir"
        );
    }
    başka {

```

```

        Sistem. out .println( "1. Yöntem: str2, str1'in alt dizisi
DEĞİLDİR" );
    }

    if ( str2 .contains( str1 )) {
        Sistem. out .println( "2. Yöntem: str2, str1'in alt dizisidir"
);
    }
    başka {
        Sistem. out .println( "2. Yöntem: str2, str1'in alt dizisi
DEĞİLDİR" );
    }
}
}

```

Bunun sonucunda:

1. Yöntem: str2, str1'in alt dizisidir
2. Yöntem: str2, str1'in alt dizisidir

21D. Java Dizeleri - Soruları

String tipindeki bir değişken hangi değere otomatik olarak başlatılır?

Bir String türünün varsayılan değeri null'dur.

Java'da Karakter Sabiti ile Dize Sabiti arasındaki fark?

Karakter sabiti tek tırnak içine alınır. Dize sabitleri çift tırnak içine alınır. Karakter sabitleri tek rakam veya karakterdir. Dize Sabitleri karakter koleksiyonudur. Örn : '2', 'A' Örn : "Merhaba Dünya"

Java'da Dizeler oluşturma'nın farklı yolları nelerdir?

Dizeler temel olarak üç şekilde oluşturulabilir.

1. Karakter Dizisini Kullanma
Eski:
karakter[] ch = {'H','e','l','l','o'};
String str = new String(ch);
2. Dize Değişmezini Kullanma
String str1 = "Merhaba";
3. Yeni anahtar kelime kullanma
String str2 = new String("Merhaba");

Verilen karakteri String'den kaldıracak bir yöntem yazın mı?

Bir String'in tüm oluşumlarını başka bir String ile değiştirmek için replacementAll yöntemini kullanabiliriz. Dikkat edilmesi gereken önemli nokta, String'i argüman olarak kabul etmesidir, bu nedenle String'i oluşturmak için Character sınıfını kullanacağız ve onu tüm karakterleri boş String ile değiştirmek için kullanacağız.

```
str.replaceAll(Character.toString(c), "");
```

```

halk sınıf StringsEx {

    halk statik void main(String[] args ) {

        String str = new String( "Selenyum Dersleri" );
    }
}

```

```

//yukarıdakilerden e'yi çıkarıyoruz

karakter ch = 'e' ;

String str1 = str .replaceAll(Character.toString ( ch ) , "" );

Sistem. out .println( str1 );

}

}

```

Bir dizeyi büyük harfe veya küçük harfe nasıl dönüştürebiliriz?

toLowerCase() ve toUpperCase() yöntemlerini kullanma.

Büyük/küçük harf dikkate alınmadan dizeleri nasıl karşılaştırabiliriz?

equalsIgnoreCase(String str) yöntemini kullanma.

Örnek:

```

halk sınıf StringsEx {

    halk statik void main(String[] args ) {

        String str1 = new String( "SELENYUM ÖĞRETİCİLERİ" );

        String str2 = new String( "Selenyum Dersleri" );

        if ( str1 .equalsIgnoreCase( str2 )) {
            Sistem. out .println( "Her iki dize de eşittir" );
        }
        başka {
            Sistem. out .println( "Dizeler eşit değil" );
        }

    }

}

```

Java'da bir dize nasıl bölünür?

split(string regex) yöntemini kullanma

Örnek:

```

halk sınıf StringsEx {

    halk statik void main(String[] args ) {

        String str1 = new String( "Subbu'dan Selenyum Dersleri" );

        String[] arr1 = str1 .split( " " );

        Sistem. out .println( "dizinin uzunluğu " + dizi1 . uzunluk );

        için ( int ben =0; ben < arr1 . uzunluk ; ben ++ ) {
            Sistem. out .println( arr1 [ i ] );
        }

    }

}

```

```
}
```

Sonuç:

```
dizinin uzunluğu 4
Selenium
Öğreticiler
ile
subbu
```

Aşağıdakilerle kaç nesne oluşturulacak?

String str = "Selenium Dersleri"

String str1 = "Selenium Dersleri"

Yalnızca tek bir nesne.

equals() yöntemi ile == operatörü arasındaki fark nedir?

equals() yöntemi dizelerin içeriğiyle eşleşirken == operatörü dizelerin nesnesi veya referansı ile eşleşir.

String sınıfı final mi?

Evet.

Bir dizinin boş olup olmadığı nasıl kontrol edilir?

Bir dizinin boş olup olmadığını bulmak için dizinin uzunluğunu kontrol edebiliriz. Dize uzunluğunu kontrol etmek için isEmpty() yöntemini de kullanabiliriz.

```
halk sınıf StringsEx {

    halk statik void main(String[] args ) {

        String str1 = new String( "" );

        if ( str1 .isEmpty()) {
            Sistem. out .println( "usign isEmpty() - Dize boş" );
        }

        if ( str1 .length() == 0) {
            Sistem. out .println( "uzunluk() kullanılıyor - Dize boş" );
        }
    }

}
```

Bir dizeyi int'ye nasıl dönüştürebilirim?

Bir dizeyi yalnızca dizinin numaralı bir dize olması durumunda int'ye dönüştürebiliriz. Aksi halde "NumberFormatException" istisnası atılır

Örnek:

```
halk sınıf StringsEx {

    halk statik void main(String[] args ) {

        String str1 = new String( "67" );
```

```

        int hayır = Tamsayı. parseInt ( str1 );

        Sistem. out .println( hayır );
    }
}

```

Bir dizeyi ikiye nasıl dönüştürebilirim?

Double.parseDouble() yöntemini kullanarak bir dizeyi double'a dönüştürüyoruz.

```

halk sınıf StringsEx {

    halk statik void main(String[] args ) {

        String str1 = new String( "67.89" );

        çift hayır = Çift. parseDouble ( str1 );

        Sistem. out .println( hayır );

    }
}

```

22. Java Dizileri

Java Dizileri:

Dizi, benzer türdeki öğelerin bir koleksiyonudur.

Java dizisi, benzer veri tipindeki öğeleri içeren bir nesnedir. Benzer öğeleri sakladığımız bir veri yapısıdır. Bir Java dizisinde yalnızca sabit bir öğe kümesini saklayabiliriz.

Avantajları:

Kod Optimizasyonu

Rasgele erişim

Dezavantajları:

Bir dizide yalnızca sabit boyuttaki öğeler saklanabilir.

Dizi Türleri:

Tek Boyutlu

Çok boyutlu

Tek Boyutlu Dizinin Bildirilmesi:

veri türü[] arr;

veri türü []arr;

veri türü arr[];

Bir dizi nasıl başlatılır:

ArrayRefVar = yeni veri türü[boyut];

Örnek:

```
paket FPPaket;
```

```
halk sınıf DizilerDemo {
```

```
    halk statik void main(String[] args ) {
        int test dizisi [] = yeni int [5];
        test dizisi [0] = 1;
        test dizisi [1] = 2;
        test dizisi [2] = 3;
        test dizisi [3] = 4;
        test dizisi [4] = 5;

        Sistem. out .println( test dizisi [0]);
        Sistem. out .println( test dizisi [1]);
        Sistem. out .println( test dizisi [2]);
        Sistem. out .println( test dizisi [3]);
        Sistem. out .println( test dizisi [4]);
    }
}
```

Dizi uzunluğu .length ile bulunabilir.

Örn: testarray.length

```
paket FPPaket;
```

```
halk sınıf DizilerDemo {
```

```
    halk statik void main(String[] args ) {
        int test dizisi [] = yeni int [5];
        test dizisi [0] = 1;
        test dizisi [1] = 2;
        test dizisi [2] = 3;
        test dizisi [3] = 4;
        test dizisi [4] = 5;

        Sistem. out .println( "Dizi uzunluğu: " + test dizisi . uzunluk );

        için ( int ben =0; i < test dizisi . uzunluk ; ben ++ ) {
            Sistem. out .println( test dizisi [ i ] );
        }
    }
}
```

Bir diziye aşağıdaki gibi bildirebilir, başlatabilir ve başlatabiliriz.

```
int a[] = {1,2,3,4,5}
```

Örnek:

```
paket FPPaket;
```

```
halk sınıf DizilerDemo {
```

```
    halk statik void main(String[] args ) {
        int test dizisi [] = {1,2,3,4,5};

        Sistem. out .println( "Dizi uzunluğu: " + test dizisi . uzunluk );

        için ( int ben =0; i < test dizisi . uzunluk ; ben ++ ) {
            Sistem. out .println( test dizisi [ i ] );
        }
    }
}
```

```

    }
}

```

22A. Java Dizileri - Bir Diziyi Bir Yönteme Aktarmak

Java Dizileri:

Bir Diziyi Bir Yönteme Aktarmak:

Maksimumu Bulma Bir Dizideki Değer:

```

paket FPPaket;

halk sınıf DizilerDemo {

    halk statik int findMax( int varış []) {
        int maksimum = dizi [0];
        için ( int ben = 0; ben < varış uzunluk ; ben ++ ) {
            if ( max < dizi [ i ] ) {
                maksimum = dizi [ ben ];
            }
        }
        geri dönmek maksimum ;
    }

    halk statik void main(String[] args ) {

        int test dizisi [] = {5,33,65,43,67,34};
        int maxvalue = findMax ( testdizisi );
        Sistem. out .println( "Maksimum Değer: " + maksimum değer );
    }
}

```

22B. Java Dizileri - Bir Yöntemden Dizi Döndürme

Java Dizileri:

Bir yöntemden dizi döndürme:

```

paket FPPaket;

halk sınıf DizilerDemo {

    halk statik int [] returnArray() {
        int [] dizi = {1,2,3,4,5};
        geri dönmek varış ;
    }

    halk statik void main(String[] args ) {

        int arr [] = returnArray ();
        için ( int ben =0; ben < varış uzunluk ; ben ++ ) {
            Sistem. out .println( arr [ i ] );
        }
    }
}

```

22C. Java Dizileri - ArrayIndexOutOfBoundsException İstisnası

Java Dizileri:

ArrayIndexOutOfBoundsException:

Dizinin indeksinde olmayan bir öğeye erişmeye çalıştığınızda, ArrayIndexOutOfBoundsException hatasıyla karşılaşsınız.

```
paket FPPaket;

halk sınıf DizilerDemo {

    halk statik void main(String[] args ) {
        int [] dizi = {1,2,3,4,5};
        için ( int ben =0; ben <= arr.uzunluk ; ben ++ ) {
            Sistem. out .println( arr [ i ] );
        }
    }
}
```

Bu, "ArrayIndexOutOfBoundsException" değerini verir.

22D. Java Dizileri – ArrayStoreException

Java Dizileri:

ArrayStoreException nedir?

ArrayStoreException, dizi türünden farklı türde bir öğe sakladığınızda gelir.

Örnek:

```
paket paket1;

halk sınıf DizilerÖrnek {

    halk statik void main(String[] args ) {

        Nesne[] arrobj = yeni Dize[3];

        arrobj [0] = new Tamsayı(3);

    }

}
```

Sonuç: "main" iş parçacığında istisna [Java.lang.ArrayStoreException](#) :
Java.lang.Integer
package1.ArraysExample.main([ArraysExample.java:9](#)) konumunda

22E. Java Dizileri - Bir Dizi Üzerinde Yineleme Yapmak

Java Dizileri:

Java'da bir dizi üzerinde nasıl yineleme yapabilirsiniz?

Bir for döngüsü veya her bir döngü için bir dizi kullanarak yineleme yapabilirsiniz.

Örnek: for döngüsünü kullanma

```
paket paket1;

halk sınıf DizilerÖrnek {

    halk statik void main(String[] args ) {

        String arr1 [] = { "subbu" , "selenium" , "öğreticiler" };

        için ( int ben=0; ben < arr1 . uzunluk ; ben ++ ) {
            Sistem. out .println( arr1 [ i ] );
        }

    }

}
```

Örnek: Her döngü için kullanma

```
paket paket1;

halk sınıf DizilerÖrnek {

    halk statik void main(String[] args ) {

        String arr1 [] = { "subbu" , "selenium" , "öğreticiler" };

        for (Dize öğesi : arr1 ) {
            Sistem. out .println( öğe );
        }

    }

}
```

22F. Java Dizileri - Bir Dizideki Öğeleri Sıralama

Java Dizileri:

Bir dizideki öğeler nasıl sıralanır?

Dizi öğeleri Arrays.sort() yöntemi kullanılarak sıralanabilir.

```
paket paket1;

içe aktarın ;

halk sınıf DizilerÖrnek {

    halk statik void main(String[] args ) {

        String arr1 [] = { "subbu" , "selenium" , "java" , "öğreticiler" };

    }
```

```

Diziler. sıralama ( arr1 );

için ( int ben =0; ben < arr1 . uzunluk ; ben ++ ) {
    Sistem. out .println( arr1 [ i ] );
}

int dizi2 [] = {5,3,6,4,8,7,2,1,10,9};
Diziler. sıralama ( arr2 );

Sistem. out .println(Arrays.toString ( arr2 ));

}

}

```

Sonuç:

```

[java, selenium, subbu, öğretmenler]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

```

22G. Java Dizileri - Dizi Arama

Java Dizileri:

Orada bir öğenin olup olmadığını kontrol etmek için bir dizi nasıl aranır?

For döngüsü kullanarak bir öğeyi arayabilirsiniz. Ayrıca önce diziyi sıralayabilir ve ardından öğeyi kontrol etmek için Arrays.binarySearch() yöntemini kullanabilirsiniz.

For döngüsünü kullanma:

```

paket paket1;

içer aktarmak java.util.Arrays ;

halk sınıf DizilerÖrnek {

    halk statik void main(String[] args ) {

        String arr1 [] = { "subbu" , "java" , "selenium" , "öğreticiler" };

        String str_to_check = "Subbu" ;
        boolean kontrol = yanlış ;

        için ( int ben =0; ben < arr1 . uzunluk ; ben ++ ) {

            if ( arr1 [ i ]. equalsIgnoreCase ( str_to_check )) {
                kontrol = doğru ;
                kırmak ;
            }
        }

        eğer ( kontrol et ) {
            Sistem. out .println( str_to_check + " dizide bulundu" );
        }
    }
}

```

```

        başka {
            Sistem. out .println( str_to_check + " dizide bulunamadı" );
        }
    }
}

```

Arrays.binarySearch() yöntemini kullanarak:

```

paket paket1;

içe aktarın ;

halk sınıf DizilerÖrnek {

    halk statik void main(String[] args ) {

        String arr1 [] = { "subbu" , "java" , "selenium" , "öğreticiler" };

        Diziler. sıralama ( arr1 );

        Sistem. out .println(Arrays.binarySearch ( arr1 , " subbu" ));

    }

}

```

Ayrıca dizileri List veya Set'e dönüştürebilir ve öğeyi kontrol etmek için include() yöntemini kullanabilirsiniz.

22H. Java Dizileri - Object.clone() kullanarak bir Diziyi kopyalama

Java Dizileri:

Object.clone() yöntemini kullanarak bir diziyi kopyalayabilirsiniz.

```

paket paket1;

içe aktarın ;

halk sınıf DizilerÖrnek {

    halk statik void main(String[] args ) {

        String arr1 [] = { "subbu" , "selenium" , "java" , "öğreticiler" };

        String arr2 [] = yeni String[5];

        arr2 = arr1 .clone();

        arr2 [2] = "merhaba" ;

        Sistem. out .println(Arrays.toString ( arr1 ) );
        Sistem. out .println(Arrays.toString ( arr2 ) );

    }

}

```

```
}
```

22I. Java Dizileri - System.arraycopy() kullanarak bir Diziyi kopyalamak

Java Dizileri:

System.arraycopy() yöntemini kullanarak bir diziyi kopyalayabilirsiniz.

System.arraycopy() aşağıdaki bağımsız değişkenleri alır.

İlk argüman: Kaynak dizisi

İkinci argüman: Kaynak dizide kopyalamanın başlaması gereken dizin

Üçüncü argüman: Hedef dizisi

Dördüncü argüman: Hedef dizide öğelerin kopyalanması gereken dizin

Beşinci argüman: Kopyalanacak öğelerin sayısı.

```
paket paket1;

içe aktarın ;

halk sınıf DizilerÖrnek {

    halk statik void main(String[] args ) {

        String arr1 [] = { "subbu" , "selenium" , "java" , "öğreticiler" };

        String arr2 [] = yeni String[4];

        Sistem. dizikopyası ( dizi1 , 0, dizi2, 0, 4);

        Sistem. out .println(Arrays.toString ( arr1 ) );
        Sistem. out .println(Arrays.toString ( arr2 ) );

    }

}
```

22J. Java Dizileri - Arrays.copyOf() kullanarak bir Diziyi kopyalama

Java Dizileri:

Arrays.copyOf() yöntemini kullanarak bir diziyi kopyalayabilirsiniz.

Arrays.copyOf() yöntemi iki argüman alır.

İlk Argüman: Bu kaynak dizisidir

İkinci Argüman: 0 indeksinden başlayarak kaynak diziden kopyalanması gereken eleman sayısı

```
paket paket1;
```

```

içe aktarın ;

halk sınıf DizilerÖrnek {

    halk statik void main(String[] args ) {

        String arr1 [] = { "subbu" , "selenium" , "java" , "öğreticiler" };

        String arr2 [] = yeni String[4];

        arr2 = Diziler. copyOf ( arr1 , 3);

        Sistem. out .println(Arrays.toString ( arr1 ) );
        Sistem. out .println(Arrays.toString ( arr2 ) );

    }

}

```

22K. Java Dizileri - Arrays.copyOfRange() kullanarak bir Diziyi kopyalamak

Java Dizileri:

Arrays.copyOfRange() yöntemini kullanarak bir diziyi kopyalayabilirsiniz.

Arrays.copyOfRange() yöntemi üç bağımsız değişken alır.

İlk Argüman: Kaynak Dizisi

İkinci Argüman: Öğelerin kopyalanması gereken başlangıç dizini

Üçüncü Bağımsız Değişken: Kopyalanması gereken öğelerin bitiş dizini. Öğeler kopyalandığında bu dahil edilmez.

```

paket paket1;

içe aktarın ;

halk sınıf DizilerÖrnek {

    halk statik void main(String[] args ) {

        String arr1 [] = { "subbu" , "selenium" , "java" , "öğreticiler" };

        String arr2 [] = yeni String[4];

        arr2 = Diziler. copyOfRange ( arr1 , 1, 3);

        Sistem. out .println(Arrays.toString ( _arr1 ) );
        Sistem. out .println(Arrays.toString ( arr2 ) );

    }

}

```

22L. Java Dizileri - Çok Boyutlu Diziler

Java Dizileri:

Çok Boyutlu Dizi:

Sözdizimi:

- dataType[][] arrayRefVar;
- dataType [][]arrayRefVar;
- dataType arrayRefVar[][];
- dataType []arrayRefVar[];

Çok Boyutlu Bir Dizin Örneklenmesi:

```
int[][] testarray = new int[2][2];
```

Çok Boyutlu Dizin Başlatılması:

```
testdizisi[0][0] = 1;
```

```
testdizisi[0][1] = 2;
```

```
testdizisi[1][0] = 3;
```

```
testdizisi[1][1] = 4;
```

Çok boyutlu bir diziyi aşağıdaki şekilde tanımlayabilir, başlatabilir ve başlatabilirsiniz.

```
Int[][] testarray = {{1,2},{3,4}};
```

Örnek:

```
paket FPPaket;
```

```
halk sınıf DizilerDemo {
```

```
    halk statik void main(String[] args ) {
        int [][] arr = {{1,2,3},{4,5,6},{7,8,9}};
        için ( int ben =0; ben <3; ben ++ ) {
            için ( int j =0; j <3; j ++ ) {
                Sistem. out .println( arr [ i ] [ j ] );
            }
        }
    }
}
```

22M. Java Dizileri - Soruları

Java'da uzunluk ve uzunluk() yöntemi arasındaki fark nedir?

uzunluk() : String sınıfında, dizedeki karakter sayısını döndürmek için kullanılan uzunluk() yöntemimiz vardır. Örn: String str = "Merhaba Dünya"; System.out.println(str.length()); Str.length(), boşluk dahil 11 karakter döndürecektir.

uzunluk : dizilerde, dizideki değerlerin veya nesnelerin sayısını döndürecek uzunluk örneği değişkenimiz var. Örneğin: String günler[]={ " Paz", "Pzt", "Çarşamba", "Per", "Cuma", "Cumartesi"}; Days dizisindeki değer sayısı 6 olduğundan 6 değerini döndürecektir.

Bir diziyi oluşturduktan sonra boyutunu değiştirebilir misiniz?

Hayır. Bunu değiştiremezsin.

Java'da dizi öğelerine nasıl erişilir?

İndeks ve indeks kullanımı 0'dan başlar.

Dizi bellekte nerede saklanır?

Dizi, JVM belleğinin yığın alanında oluşturulur. Dizi Java'da nesne olduğundan, bir yöntem veya blok içinde yerel olarak dizi oluştursanız bile, nesneye her zaman yığından bellek ayrılır.

Bir dizeyi Java'da bir tamsayı dizisine kaydedebilir misiniz?

Hayır. Derleme zamanı hatası alırsınız.

```
paket paket1;
```

```
halk sınıf DizilerÖrnek {
```

```
    halk statik void main(String[] args ) {
```

```
        int arr1 [] = yeni int [10];
```

```
        arr1 [0] = new Tamsayı(9);
```

```
        arr1 [1] = "subbu"; //Hata: String'den int'ye dönüştürülemiyor
```

```
    }
```

```
}
```

ArrayIndexOutOfBoundsException İstisnası nedir?

ArrayIndexOutOfBoundsException istisnası, kodunuz belirli bir diziden geçersiz bir dizine erişmeye çalıştığında ortaya çıkar.

Java'da iki boyutlu dizi nedir?

Java'da bir dizi dizisidir. Üç satır ve üç sütundan oluşan iki boyutlu bir diziyi şu şekilde bildirebilirsiniz:

```
int[][] asal sayılar = yeni int[3][3]
```

ArrayStoreException nedir?

ArrayStoreException, dizi türünden farklı türde bir öğe sakladığınızda gelir.

Java'da bir dizi üzerinde nasıl yineleme yapabilirsiniz?

Bir for döngüsü veya her bir döngü için bir dizi kullanarak yineleme yapabilirsiniz.

Bir dizideki öğeler nasıl sıralanır?

Dizi öğeleri Arrays.sort() yöntemi kullanılarak sıralanabilir.

Orada bir öğenin olup olmadığını kontrol etmek için bir dizi nasıl aranır?

For döngüsü kullanarak bir öğeyi arayabilirsiniz. Ayrıca önce diziyi sıralayabilir ve ardından öğeyi kontrol etmek için `Arrays.binarySearch()` yöntemini kullanabilirsiniz.

Bir diziyi başka bir diziye kopyalamak için = kullanabilir miyiz?

Bir diziyi başka bir diziye kopyalamak için = öğesini kullanamayız. Diziler değiştirilebilir ve bir diziyi kopyalamak için = kullandığımızda, aslında diziyi kopyalamaz ancak ikinci dizi, ilk dizinin aynı hafıza alanına işaret eder. Diziler değiştirilebilir olduğundan, bir dizideki bir değeri değiştirirsek, ikinci dizideki değer de değişir.

Örnek:

```
paket paket1;

içe aktarın ;

halk sınıf DizilerÖrnek {

    halk statik void main(String[] args ) {

        String arr1 [] = { "subbu" , "selenium" , "java" , "öğreticiler" };

        String arr2 [] = yeni String[5];

        dizi2 = dizi1 ;

        arr2 [2] = "merhaba" ;

        Sistem. out .println(Arrays.toString ( arr1 ) );
        Sistem. out .println(Arrays.toString ( arr2 ) );

    }

}
```

Sonuç:

```
[subbu, selenium, merhaba, öğreticiler]
[subbu, selenium, merhaba, öğreticiler]
```

Bir diziyi başka bir diziye nasıl kopyalarım?

1. `Object.clone()`
2. `System.arraycopy()`
3. `Diziler.copyOf()`
4. `Arrays.copyOfRange()`

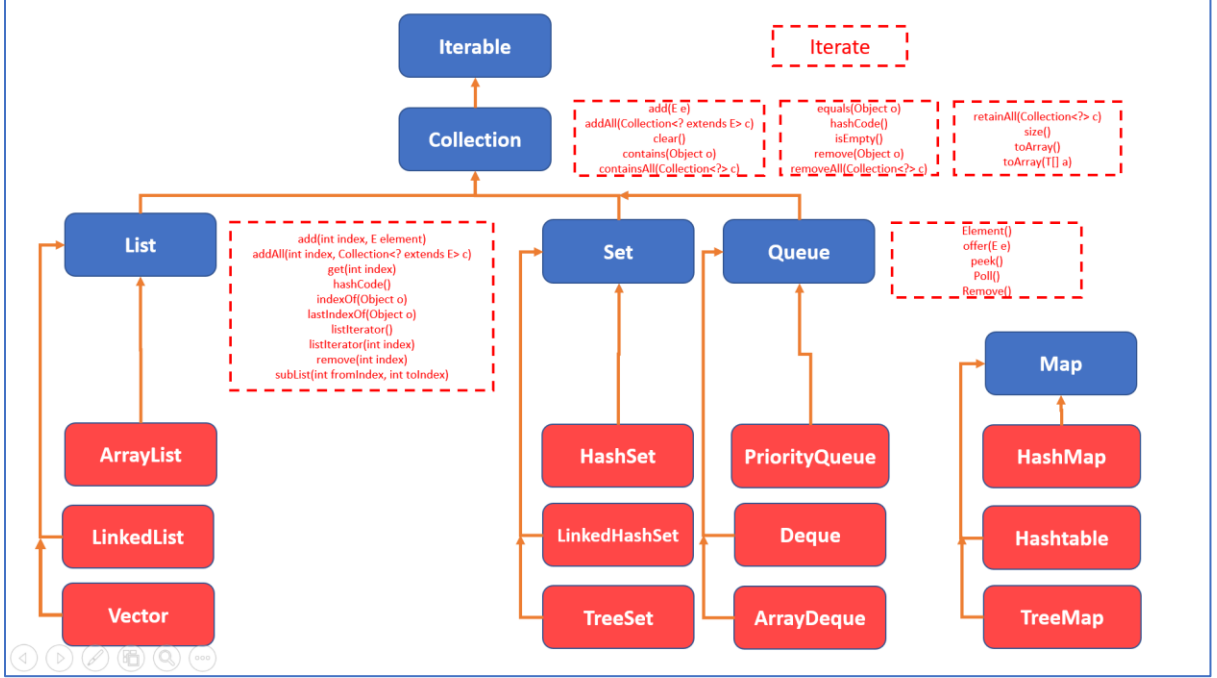
23. Koleksiyon Çerçevesi – Giriş

Java Koleksiyonları Çerçevesi:

Dizilerin ana dezavantajı sabit boyutta olmalarıdır. Uzunluğu 3 olan bir dizi bildirirsek, içine yalnızca 3 eleman ekleyebiliriz. Dizinin boyutunu değiştiremeyiz. Bunu artıramayız veya azaltamayız.

Daha fazla veya daha az öğeye sahip olmak istiyorsak, genişletilebilir veya daraltılabilir bir diziye ihtiyacımız var ve Koleksiyonlar çerçevesi bunun için bir cevaptır.

Liste, Küme, Kuyruk ve Harita, Collections Framework'ün dört önemli arayüzüdür. Harita, Koleksiyon arayüzünü uygulamaz ancak Koleksiyon Çerçevesinin bir parçasıdır.



Koleksiyonlar çerçevesindeki ilk arayüz “Yinelenebilir” arayüzdür ve koleksiyonlar üzerinde yinelenen bir “Yineleme” yöntemi tanımlamıştır.

İkincisi ise ek yöntemlerin tanımlandığı “Koleksiyon” arayüzüdür.

Sonraki arayüzler List, Set ve Queue'dur ve bunların hepsi "Collection" çerçevesini genişletiyor ve bazı ek yöntemler tanımlıyor.

Bu arayüzleri uygulayan ArrayList, HashSet vb. farklı sınıflar geliştirildi.

Harita, Koleksiyon arayüzünü genişletmiyor ancak hâlâ koleksiyon çerçevesi altında.

Geliştirilen arayüzler ve sınıflar hakkında daha fazla bilgi için aşağıdaki bağlantıya (Oracle dokümanları) göz atabilirsiniz.

<https://docs.oracle.com/javase/7/docs/api/java/lang/Iterable.html>

Liste, Set, Kuyruk ve Harita Arasındaki Farklar:

Java'daki liste, kopyalar içerebilecek sıralı ve indekslenmiş koleksiyon sağlar.

Set arayüzü benzersiz nesnelerin sırasız bir koleksiyonunu sağlar, yani Set kopyalara izin vermez.

Kuyruk FIFO (İlk Giren İlk Çıkar) esasına göre çalışır. FIFO'da ilk öğe ilk önce kaldırılır ve son öğe sonuncudur.

Harita, anahtar değer çiftine dayalı bir veri yapısı sağlar.

Liste – ArrayList ve LinkedList arasındaki farklar

Dizi Listesi:

1. ArrayList, öğeleri depolamak için dahili olarak dinamik diziye kullanır
2. ArrayList ile manipülasyon, dahili olarak dizi kullandığından **yavaştır** . Diziden herhangi bir öğe çıkarılırsa bellekteki tüm bitler kaydırılır.
3. ArrayList, verileri depolamak ve verilere erişmek için daha iyidir .

Bağlantılı liste:

1. öğeleri depolamak için dahili olarak çift bağlantılı listeyi kullanır .
2. LinkedList ile manipülasyon ArrayList'ten daha hızlıdır çünkü çift bağlantılı liste kullanır, dolayısıyla bellekte bit kaydırma gerekmez.
3. LinkedList verileri işlemek için daha iyidir .

23A. Koleksiyonlar Çerçevesi – ArrayList

Dizi Listesi:

ArrayList, kendi içerisinde ek elemanları barındıracak şekilde genişletilebilen ve elemanlar çıkarıldığında daha küçük bir boyuta küçülebilen bir veri yapısıdır. Başka bir deyişle dinamik bir dizidir.

Örnek: add(), size() ve kaldır() yöntemleri

paket FPPaket;

içe aktarın ;

halk sınıf ArrayListDemo {

```

    halk statik void main(String[] args ) {
        Arraylist bildirimi
        Dizi Listesi bir = yeni DiziListesi();

        //Dizi listesine eleman ekliyoruz
        a .add( "Subbu" );
        a .add( "Raj" );
        a .add( "Krishna" );
        a .add( "Vişnu" );
        a .add( "Raghu" );

        // Arraylist'ten elementler alınıyor
        Sistem. out .println( a .get(0));
        Sistem. out .println( a .get(1));
        Sistem. out .println( a .get(2));

        Arraylist'teki eleman sayısını alıyoruz
        Sistem. out .println( a .size());

        Arraylist'i yazdırmak
        Sistem. out .println( a );

        //Dizi listesinden bir elemanın kaldırılması
        a .remove( "Vishnu" );
    }

```

```

Sistem. out .println( a .size());
Sistem. out .println( a );

//Bir öğeyi dizine göre kaldırma
a .remove(2);

Sistem. out .println( a .size());
Sistem. out .println( a );

// Dizi listesinin bir değer içerip içermediğini kontrol ediyoruz
Sistem. out .println( "Vişnu İçerir " + a .contains( "Vişnu" ));
}
}

```

Yukarıdaki dizi listesi her şeyi ve hatta tam sayıları da kabul edebilir.

Örn: Farklı veri türlerine sahip ArrayList

```

paket FPPaket;

içe aktarmak java.util.ArrayList ;

halk sınıf ArrayListDemo {

    halk statik void main(String[] args ) {
        Arraylist bildirimi
        Dizi Listesi bir = yeni DiziListesi ();

        //Dizi listesine eleman ekliyoruz
        a .add( "Subbu" );
        a .add( "Raghu" );
        a .add(1);

        Arraylist'i yazdırmak
        Sistem. out .println( a );
    }
}

```

Burada önceki string değerlerine 1 tamsayı değerini ekliyoruz. Bu herhangi bir soruna neden olmaz ve şu sonucu üretir:

```
[Subbu, Raghu, 1]
```

Ancak eğer dizi listesini yalnızca belirli bir veri tipini kabul edecek şekilde kısıtlamak istiyorsak, bunu aşağıdaki şekilde yapabiliriz.

```
ArrayList<String> a = new ArrayList<String>();
```

Şimdi yukarıdaki sorun tamsayıları kabul etmeyeceği için bir derleme hatası üretiyor.

23B. Koleksiyonlar Çerçevesi - ArrayList aracılığıyla yineleme

Bir arraylist aracılığıyla yineleme:

İki yol:

1. Yineleyiciyi Kullanma

```

paket FPPaket;

içe aktarın ;
içe aktarın ;

halk sınıf ArrayListDemo {

    halk statik void main(String[] args ) {
        ArrayList bildirimi
        ArrayList<String> a = new ArrayList<String>();

        //Dizi listesine eleman ekliyoruz
        a .add( "Subbu" );
        a .add( "Raghu" );
        a .add( "Venü" );
        a .add( "Rajesh" );

        //Listede yinelemeler yapıyoruz
        Yineleyici itr = a .iterator();
        while ( itr .hasNext()) {
            Sistem. out .println( itr .next());
        }
    }
}

```

2. For Döngüsünü Kullanmak:

```

paket FPPaket;

içe aktarın ;
içe aktarmak java.util.Iterator ;

halk sınıf ArrayListDemo {

    halk statik void main(String[] args ) {
        ArrayList bildirimi
        ArrayList<String> a = new ArrayList<String>();

        //Dizi listesine eleman ekliyoruz
        a .add( "Subbu" );
        a .add( "Raghu" );
        a .add( "Venü" );
        a .add( "Rajesh" );

        //Listede yinelemeler yapıyoruz
        for (String s : a ) {
            Sistem. out .println( s );
        }
    }
}

```

23C. Koleksiyonlar Çerçevesi - Bir Listeyi Başka Bir Listeye Ekleme

addAll yöntemi:

```

paket FPPaket;

içe aktarın ;
içe aktarmak java.util.Iterator ;

```

```

halk sınıf ArrayListDemo {

    halk statik void main(String[] args ) {
        ArrayList bildirimi
        ArrayList<String> a = new ArrayList<String>();

        //Dizi listesine eleman ekliyoruz
        a .add( "Subbu" );
        a .add( "Raghu" );
        a .add( "Venü" );
        a .add( "Rajesh" );

        ArrayList<String> b = new ArrayList<String>();
        b .add( "Ganesh" );
        b .add( "Vamşi" );

        //b listesini a'ya ekleme
        a .addAll( b );

        //Listede yinelemeler yapıyoruz
        for (String s : a ) {
            Sistem. out .println( s );
        }
    }
}

```

23D. Koleksiyonlar Çerçevesi - Bir Listeyi Başka Bir Listedden Kaldırma

RemoveAll() Örnek:

```

paket FPPaket;

içe aktarın ;
içe aktarmak java.util.Iterator ;

halk sınıf ArrayListDemo {

    halk statik void main( String [] args ) {
        ArrayList bildirimi
        ArrayList< String > a = new ArrayList< String >();

        //Dizi listesine eleman ekliyoruz
        a .add( "Subbu" );
        a .add( "Raghu" );
        a .add( "Venü" );
        a .add( "Rajesh" );

        ArrayList< String > b = new ArrayList< String >();
        b .add( "Raghu" );
        b .add( "Rajesh" );
        b .add( "Vamşi" );

        //b listesini a'ya ekleme
        a .removeAll( b );

        //Listede yinelemeler yapıyoruz
        ( Dize için ) s : a ) {
            Sistem. out .println( s );
        }
    }
}

```

```

    }
}

```

23E. Koleksiyonlar Çerçevesi - keepAll()

keepAll() Örnek:

```

paket FPPaket;

içe aktarın ;
içe aktarmak java.util.Iterator ;

halk sınıf ArrayListDemo {

    halk statik void main(String[] args ) {
        Arraylist bildirimi
        ArrayList<String> a = new ArrayList<String>();

        //Dizi listesine eleman ekliyoruz
        a .add( "Subbu" );
        a .add( "Raghu" );
        a .add( "Venü" );
        a .add( "Rajesh" );

        ArrayList<String> b = new ArrayList<String>();
        b .add( "Raghu" );
        b .add( "Rajesh" );
        b .add( "Vamşi" );

        //b listesini a'ya ekleme
        A . keepAll( b );

        //Listede yinelemeler yapıyoruz
        for (String s : a ) {
            Sistem. out .println( s );
        }
    }
}

```

23F. Koleksiyonlar Çerçevesi - Liste Listesi

Liste Listesi:

```

paketi ;

içe aktarın ;
içe aktarın ;

halk sınıf Liste Listesi {

    halk statik void main(String[] args ) {
        List<List<String>> lls = new ArrayList<List<String>>();

        List<String> ls = new ArrayList<String>();
        ls .add( "subbu" );
        ls .add( "selenium" );
        lls .add( ls );
    }
}

```

```

List<String> ls1 = new ArrayList<String>();
ls1 .add( "java" );
ls1 .add( "öğreticiler" );
lls .add( ls1 );

Sistem. out .println( lls );

için ( int ben =0; i < lls .size(); ben ++ ) {
    Sistem. out .println( lls .get( i ).get(0));
    Sistem. out .println( lls .get( i ).get(1));
}

}

}

```

23G. Collections Framework - Listeyi sıralama

Java'da bir listeyi sıralama:

```

paket ;

içe aktarın ;
içe aktarın ;
içe aktarın ;

halk sınıf Liste Listesi {

    halk statik void main(String[] args ) {
        List<String> ls = new ArrayList<String>();
        ls .add( "subbu" );
        ls .add( "selenyum" );
        ls .add( "java" );
        ls .add( "öğreticiler" );

        Sistem. out .println( ls );

        //Artan sırada
        Koleksiyonlar. sıralama ( ls );
        //Azalan sırayla
        //Collections.sort( ls_, Collections.reverseOrder());

        Sistem. out .println( ls );

    }

}

```

23H. Koleksiyonlar Çerçevesi – LinkedList

Bağlantılı liste:

Bağlantılı liste sırasız bir listedir.

LinkedList aşağıdaki gibi bildirilebilir.

1. LinkedList nesnesi = yeni LinkedList();
Her türlü veri türünü kabul eder
2. LinkedList<String> nesnesi = new LinkedList<String>();
Yalnızca String türünü kabul eder.

Örnek:

Aşağıdaki örnekte aşağıdaki işlemlerle çalışacağız.

1. LinkedList'e öge ekleme
2. LinkedList'in boyutunu alma
3. Her döngü için kullanarak LinkedList'in öğelerini alma
4. Iterator kullanarak LinkedList'in öğelerini alma
5. Bir öğeyi değere göre kaldırma
6. Bir öğeyi dizine göre kaldırma
7. Listenin ilk elemanını kaldırmak
8. Listenin son öğesinin kaldırılması
9. Bağlantılı Liste'nin belirli bir değer içerip içermediğini kontrol etme
10. İndeks kullanarak değer alma
11. İndeks kullanarak değer ayarlama

```
paket paket1;
```

```
içe aktarın ;
```

```
içe aktarın ;
```

```
halk sınıf BağlantılıListeÖrneği {
```

```
    halk statik void main(String[] args ) {
```

```
        //Bağlantılı listeden bir nesne oluşturuluyor
        LinkedList<String> l1 = new LinkedList<String>();
```

```
        //Bağlantılı listeye eleman ekliyoruz
```

```
l1 .add( "Subbu" );
l1 .add( "Selenyum" );
l1 .add( "Java" );
l1 .add( "Öğreticiler" );
l1 .add( "Abone Ol" );
l1 .add( "Youtube" );
l1 .add( "Kanal" );
```

```
        //Bağlantılı listeyi yazdırma
```

```
Sistem. out .println( l1 );
```

```
        //Bağlantılı listenin boyutu
```

```
Sistem. out .println( "Bağlantılı listenin boyutu: " + l1 .size());
```

```
        //for her döngü kullanılarak her elemanın yazdırılması
```

```
Sistem. out .println( "Değerler her döngü için kullanılarak  
yazdırılıyor:" );
```

```
    for (Dize dizisi : l1 ) {
        Sistem. out .println( str );
    }
```

```
        //Bir listede yineleme ve değerleri yazdırma
```



```

Sistem. out .println( "Listede yinelemeler yapılarak değerler
yazdırılıyor:" );
Yineleyici itr = ll .iterator();
while ( itr .hasNext() ) {
    Sistem. out .println( itr .next());
}

//Değer kullanılarak bir elemanın kaldırılması
ll .remove( "Kanal" );
Sistem. out .println( ll );

//İndeks kullanarak bir elemanın kaldırılması
ll .remove(5);
Sistem. out .println( ll );

//İlk elemanı kaldırıyoruz
ll .removeFirst();
Sistem. out .println( ll );

//Son elemanı kaldırıyoruz
ll .removeLast();
Sistem. out .println( ll );

//listenin belirli bir öge içerip içermediğini kontrol ediyoruz
boolean kontrol = ll .contains( "Java" );
eğer ( kontrol et ) {
    Sistem. out .println( "Liste Java içeriyor" );
}
başka {
    Sistem. out .println( "Java içermiyor" );
}

//Listeden bir eleman alıyoruz
String str = ll .get(1);
Sistem. out .println( "1. dizindeki dize: " + str );

//Listedeki bir elemanın ayarlanması
ll .set(2, "Alıştırma" );
Sistem. out .println( ll );
}

}

```

231. Koleksiyonlar Çerçevesi – HashSet

Ayarla – HashSet:

HashSet sırasız bir koleksiyondur. Öğelerin eklenme sırasını korumaz.

Yinelenen değerlere izin vermez. Null değerlere izin verir.

Örnek:

Aşağıdaki örnek gösterir

1. HashSet nasıl bildirilir
2. HashSet'in boş olup olmadığını kontrol edin
3. Hashset'e öge ekleme

4. Kümeye yinelenen öğeler ekleme ve yinelenen öğelere izin verip vermediğini kontrol etme
5. Kümenin boş değere izin verip vermediğini kontrol edin
6. HashSet'in boyutunu kontrol edin
7. Her döngü için değerleri kullanarak değerleri yazdırın
8. Yineleyiciyi kullanarak değerleri yazdırın
9. Bir öğeyi değere göre kaldırma
10. Bir öğeyi dizine göre kaldırma
11. Belirli bir öğeyi içerip içermediğini görmek için listeyi kontrol edin

```
paket paket1;
```

```
içe aktarın ;
```

```
içe aktarın ;
```

```
halk sınıf HashSetÖrneği {
```

```
    halk statik void main(String[] args ) {
```

```
        //Bağlantılı listeden bir nesne oluşturuluyor
        HashSet<String> hh = new HashSet<String>();
```

```
        //HashSet'in boş olup olmadığı kontrol ediliyor
        if ( hh .isEmpty())
            Sistem. out .println( "HashSet boş" );
```

```
        //Bağlantılı listeye eleman ekliyoruz
```

```
        hh .add( "Subbu" );
        hh .add( "Selenyum" );
        hh .add( "Java" );
        hh .add( "Öğreticiler" );
        hh .add( "Abone Ol" );
        hh .add( "Youtube" );
        hh .add( "Kanal" );
```

```
        //Bağlantılı listeyi yazdırma
        Sistem. out .println( hh );
```

```
        // Yinelenen değerlere izin verilip verilmediğini kontrol edin
        hh .add( "Subbu" );
        Sistem. out .println( hh );
```

```
        //Boş değerlere izin verilip verilmediğini kontrol ediyoruz
        hh .add( "" );
        Sistem. out .println( hh );
```

```
        // Hash Kümesinin Boyutu
        Sistem. out .println( "HashSet'in boyutu " + hh .size());
```

```
        //for her döngü kullanılarak her elemanın yazdırılması
        Sistem. out .println( "Değerler her döngü için kullanılarak
yazdırılıyor:" );
```

```
        for (Dize dizisi : hh ) {
            Sistem. out .println( str );
        }
```

```
        //Bir kümeyi yineleme ve değerleri yazdırma
        Sistem. out .println( "Küme boyunca yinelenerek değerler
yazdırılıyor:" );
```

```

Yineleyici itr = hh .iterator();
while ( itr .hasNext()) {
    Sistem. out .println( itr .next());
}

//Değer kullanılarak bir elemanın kaldırılması
hh . kaldır( "Kanal" );
Sistem. out .println( hh );

//İndeks kullanarak bir elemanın kaldırılması
hh . kaldır( 5 );
Sistem. out .println( hh );

//listenin belirli bir öge içerip içermediğini kontrol ediyoruz
boolean kontrol = hh .contains( "Java" );
eğer ( kontrol et ) {
    Sistem. out .println( "Küme Java içeriyor" );
}
başka {
    Sistem. out .println( "Java içermiyor" );
}
}
}

```

23J. Koleksiyonlar Çerçevesi – HashMap

Haritalar – HashMap:

HashMap, verileri Anahtar Değer çiftlerinde saklar. Harita Arayüzünü uygular. HashMap'in değerini bilmek için onun anahtarını bilmek gerekir. Dahili olarak Hashing adı verilen bir teknik kullanır.

Aşağıdaki örnek şunu gösterir:

1. HashMap nasıl bildirilir
2. HashMap'in boş olup olmadığını kontrol edin
3. Anahtar değer çiftlerini haritaya ekleme
4. Haritayı yazdırma
5. Haritanın bir anahtar içerip içermediğini kontrol edin
6. Anahtar kullanarak bir değer alın
7. Haritanın yinelenen anahtar/değer çiftlerini kabul edip etmediğini kontrol edin
8. Anahtar kullanarak anahtar/değer çiftini kaldırma
9. Haritanın bir değer içerip içermediğini kontrol edin
10. clear() kullanarak tüm anahtar değer çiftlerini kaldırma

```
paket paket1;
```

```
içe aktarın ;
```

```
halk sınıf HashMapÖrneği {
```

```

    halk statik void main(String[] args ) {
        // TODO Otomatik oluşturulan yöntem saplaması
        HashMap<String, String> hm = new HashMap<String, String>();
    }
}

```

```

//Haritanın boş olup olmadığını kontrol et
if ( hm .isEmpty()) {
    Sistem. out .println( "Harita boş" );
}

//Anahtar değer çiftlerini haritaya ekliyoruz
hm .put( "isim" , "Subbu" );
hm .put( "rol" , "Test Yöneticisi" );
hm .put( "şirket" , "IBM" );

//Harita yazdırılıyor
Sistem. out .println( hm );

//haritanın bir anahtar içerip içermediğini kontrol et
if ( hm .containsKey( "isim" )) {
    Sistem. out .println( "Harita anahtar adını içeriyor" );
}

//Anahtarı kullanarak bir değer alıyoruz
Sistem. out .println( hm .get( "isim" ));

//Haritanın yinelenen anahtar değer çiftlerini kabul edip etmediği
kontrol ediliyor
hm .put( "isim" , "Subbu" );
Sistem. out .println( hm );

//Bir anahtar/değer çiftinin kaldırılması (Şirket:IBM)
hm .remove( "şirket" );
Sistem. out .println( "IBM şirketi kaldırıldı" );
Sistem. out .println( hm );

// Haritanın belirli bir değer içerip içermediğini kontrol edin
if ( hm .containsValue( "Subbu" )) {
    Sistem. out .println( "Harita Subbu değerini içeriyor" );
}

//Bir haritadaki tüm anahtar değer çiftlerini kaldırıyoruz
hm .clear();
Sistem. out .println( "Tüm anahtar değer çiftleri kaldırıldı" );
Sistem. out .println( "Haritanın boyutu: " + hm .size());
}

}

```

23K. Koleksiyonlar Çerçevesi - Harita Haritası

Haritaların Haritaları:

```

paket i ;

içe aktarın ;
içe aktarın ;

halk sınıf Harita Haritaları {

    halk statik void main(String[] args ) {

```

```

        Map<String, Map<String, String>> mpmp = new HashMap<String,
Map<String, String>>();

        Map<String, String> mp = new HashMap<String, String>();

        mp .put( "isim" , "Subbu" );
        mp .put( "rol" , "test yöneticisi" );

        mpmp .put( "1" , mp );

        Sistem. out .println( mpmp );

        Map<String, String> mp1 = new HashMap<String, String>();

        mp1 .put( "isim" , "Venkat" );
        mp1 .put( "rol" , "test lideri" );

        mpmp .put( "2" , mp1 );

        Sistem. out .println( mpmp );

        Sistem. out .println( mpmp .get( "1" ));
        Sistem. out .println( mpmp .get( "2" ));

        Sistem. out .println( mpmp .get( "1" ).get( "isim" ));
        Sistem. out .println( mpmp .get( "1" ).get( "rol" ));

        Sistem. out .println( mpmp .get( "2" ).get( "isim" ));
        Sistem. out .println( mpmp .get( "2" ).get( "rol" ));
    }
}

```

23L. Koleksiyonlar Çerçevesi – ConcurrentHashMap

Haritalar – ConcurrentHashMap:

ConcurrentHashMap, verileri Anahtar Değer çiftlerinde saklar. Harita Arayüzünü uygular. ConcurrentHashMap'in değerini bilmek için onun anahtarını bilmek gerekir. Dahili olarak Hashing adı verilen bir teknik kullanır. ConcurrentHasMap arıza korumalıdır, bu da iş parçacığının güvenli olduğu anlamına gelir.

Selenium testlerini paralel olarak çalıştırıyorsak ve harita kullanmamız gerekiyorsa ConcurrentHashMap'i kullanın.

Aşağıdaki örnek şunu gösterir:

1. ConcurrentHashMap nasıl bildirilir
2. ConcurrentHashMap'in boş olup olmadığını kontrol edin
3. Anahtar değer çiftlerini haritaya ekleme
4. Haritayı yazdırma
5. Haritanın bir anahtar içerip içermediğini kontrol edin
6. Anahtar kullanarak bir değer alın
7. Haritanın yinelenen anahtar/değer çiftlerini kabul edip etmediğini kontrol edin
8. Anahtar kullanarak anahtar/değer çiftini kaldırma

9. Haritanın bir değer içerip içermediğini kontrol edin
10. clear() kullanarak tüm anahtar değer çiftlerini kaldırma

paket paket1;

içe aktarın ;

halk sınıf ConcurrentHashMapExample {

```

    halk statik void main(String[] args ) {
        // TODO Otomatik oluşturulan yöntem saplaması
        ConcurrentHashMap<String, String> hm = new ConcurrentHashMap<String,
String>();

        //Haritanın boş olup olmadığını kontrol et
        if ( hm .isEmpty()) {
            Sistem. out .println( "Harita boş" );
        }

        //Anahtar değer çiftlerini haritaya ekliyoruz
        hm .put( "isim" , "Subbu" );
        hm .put( "rol" , "Test Yöneticisi" );
        hm .put( "şirket" , "IBM" );

        //Harita yazdırılıyor
        Sistem. out .println( hm );

        //haritanın bir anahtar içerip içermediğini kontrol et
        if ( hm .containsKey( "isim" )) {
            Sistem. out .println( "Harita anahtar adını içeriyor" );
        }

        //Anahtarı kullanarak bir değer alıyoruz
        Sistem. out .println( hm .get( "isim" ));

        //Haritanın yinelenen anahtar değer çiftlerini kabul edip etmediği
kontrol ediliyor
        hm .put( "isim" , "Subbu" );
        Sistem. out .println( hm );

        //Bir anahtar/değer çiftinin kaldırılması (Şirket:IBM)
        hm .remove( "şirket" );
        Sistem. out .println( "IBM şirketi kaldırıldı" );
        Sistem. out .println( hm );

        // Haritanın belirli bir değer içerip içermediğini kontrol edin
        if ( hm .containsValue( "Subbu" )) {
            Sistem. out .println( "Harita Subbu değerini içeriyor" );
        }

        //Bir haritadaki tüm anahtar değer çiftlerini kaldırıyoruz
        hm .clear();
        Sistem. out .println( "Tüm anahtar değer çiftleri kaldırıldı" );
        Sistem. out .println( "Haritanın boyutu: " + hm .size());

    }
}

```

23M. Koleksiyonlar Çerçevesi - Soruları

Koleksiyon çerçevesi nedir?

Çerçeve, bir işlevsellik oluşturmak için kullanılan sınıflar ve arayüzlerden oluşur.

Java koleksiyon çerçevesi, koleksiyonların depolanması ve işlenmesi için bir dizi arayüz ve sınıf sağlar. Koleksiyonlar çerçevesi, Java.util paketi ve Java.util.concurrent paketlerindeki sınıfları ve arayüzleri içerir.

Koleksiyonlar çerçevesinin avantajları veya faydaları:

- 1) Yüksek performans
- 2) Bu çerçeveyi kullanarak farklı türde koleksiyonlar oluşturabiliriz
- 3) Kendi koleksiyonumuzu oluşturabilir ve koleksiyonu genişletebiliriz.
- 4) Programlama çabasını azaltır.
- 5) Hızı ve kaliteyi artırır: Koleksiyon çerçevesi yüksek performans, kullanışlı veri yapılarının ve algoritmaların uygulanmasını sağlar.

Koleksiyon nedir?

Koleksiyon, nesne grubunu tutan bir kaptır. Koleksiyon, nesneleri kolayca yönetmenin bir yolunu sağlar. Koleksiyonlar, nesne grubunu tek bir birim olarak yönetir. Örnekler arasında dizelerin, tamsayıların vb. listesi yer alır.

Koleksiyonlar üzerinde yaptığımız birkaç temel işlem şunlardır:

- 1) Koleksiyona nesne ekleme.
- 2) Nesneleri koleksiyondan kaldırmak veya silmek.
- 3) Nesnenin koleksiyondan alınması.
- 4) Yinelenen koleksiyon.

Java'daki Koleksiyon arayüzünü açıklayın?

Koleksiyon, Koleksiyonlar çerçevesindeki temel ve kök arayüzdür. Koleksiyon, Yinelenebilir arayüzü genişletir ve Iterator nesnesini döndüren yineleyici yöntemini devralır.

İmza: genel arayüz Koleksiyon yinelenebilirliği genişletir { }

Koleksiyon arayüzündeki yöntemler:

boolean ekle(E e);	Koleksiyona bir öğe ekler. Öğе eklenirse true değeri döndürür.
boolean kaldır(Nesne o);	Bir nesne koleksiyonda mevcutsa, bir nesneyi koleksiyondan kaldırır. Eşleşen nesne koleksiyondan kaldırılırsa true değeri döndürün.
boolean addAll(Collection<? extends E> c);	Koleksiyonda belirtilen tüm öğeleri bu koleksiyona ekler. Tüm öğeler eklenirse true değeri döndürür.

<code>boolean removeAll(Collection<?> c);</code>	Diğer koleksiyonda belirtilen tüm öğeleri bu koleksiyondan kaldırır. Tüm öğeler kaldırılırsa true değerini döndürür.
<code>int boyut();</code>	Koleksiyondaki öğelerin sayısını döndürür.
<code>boolean isEmpty();</code>	Koleksiyonun öğe içerip içermediğini kontrol eder. Hiçbir öğe mevcut değilse false değerini döndürür.
<code>boolean içerir(Nesne o);</code>	Belirtilen nesnenin koleksiyonda olup olmadığını kontrol eder. Nesne koleksiyondaysa true değerini döndürür.
<code>Yineleyici<E> yineleyici();</code>	Koleksiyon üzerinde yineleme yapmak için kullanılır. Yinelenen öğelerin sırası konusunda garanti yoktur.
<code>boolean keepAll(Collection<?> c);</code>	Belirtilen koleksiyonda olmayan tüm öğeleri kaldırır. Yalnızca koleksiyonda belirtilen öğeleri diğer öğeleri kaldırarak döndürür.
<code>Object[] toArray();</code>	Koleksiyondaki öğelerin bir dizisini döndürür.

Koleksiyonlar çerçevesinin temel arayüzleri nelerdir?

Kök arayüzü yinelenebilir ve ardından Koleksiyon çerçevemiz var. Sonra Liste, Ayarla ve Kuyruğa sahibiz. Bunlar Koleksiyonlar çerçevesinin arayüzleridir.

Harita arayüzü neden Koleksiyon arayüzünü genişletmiyor?

Haritalar, Koleksiyonlar çerçevesine ait olsalar bile koleksiyon değildir. Haritalar, anahtar değer çiftlerine dayalı işlevsellik sağlar. Koleksiyonlar bir grup öğeden oluşur. Haritalar ve Koleksiyonların çalışma şekli farklıdır, dolayısıyla Harita arayüzü koleksiyon arayüzünü genişletmez.

Yinelenebilir nedir?

Yinelenebilir, Koleksiyon arayüzü tarafından uygulanan arayüzdür. Bir koleksiyon üzerinde yineleme yapmak için kullanılan `iterator()` adı verilen bir yöntem sağlar.

Bir liste üzerinde yineleme yapmanın farklı yolları nelerdir?

Bir liste üzerinde iki şekilde yineleme yapabiliriz.

1. Yineleyici
2. For – her döngü

Koleksiyon arayüzünü genişleten arayüzleri listelemek ister misiniz?

- 1) Liste
- 2) Ayarla
- 3) Sıra
- 4) Deque

Liste Arayüzünün uygulamaları listelensin mi?

- 1) Dizi Listesi
- 2) Vektör
- 3) Bağlantılı Liste

Liste arayüzünü açıklayın?

Liste arayüzü, koleksiyondaki öğelerin sırasını depolamak için kullanılan koleksiyon arayüzünü genişletir. Hatta yinelenen öğeleri listede bile saklayabiliriz. Dizilerde yaptığımız gibi indeks kullanarak listedeki öğelere erişebilir veya öğelere erişebiliriz. Liste sıralı bir koleksiyondur.

Listede yapabileceğimiz işlemlerden bazıları:

- 1) Belirtilen dizine bir öğe eklemek.
- 2) Belirtilen dizindeki bir öğenin kaldırılması.
- 3) Öğenin indeksini almak için List, Koleksiyon arayüzü yöntemleri dışında bazı özel yöntemler içerir.

Liste arayüzüne özgü yöntemleri açıklayın?

<code>boolean addAll(int index, Koleksiyon<? extends E>c);</code>	Bu yöntem, belirtilen koleksiyondaki tüm öğeleri, belirtilen konumdaki listeye ekler.
<code>E get(int indeks);</code>	Bu yöntem listede belirtilen konumdaki bir öğeyi döndürür.
<code>E set(int indeks, E elemanı);</code>	Bu yöntem, listede belirtilen konumdaki öğeyi belirtilen öğeyle değiştirir.
<code>void add(int indeksi, E elemanı);</code>	Bu yöntem, belirtilen öğeyi belirtilen dizine ekler.
<code>E kaldır(int indeks);</code>	Bu yöntem, belirtilen dizindeki öğeyi kaldırır ve kaldırılan öğeyi döndürür.
<code>int indexOf(Object o);</code>	<code>indexOf()</code> yöntemi, belirtilen öğenin son oluşumunun dizinini döndürür. Listedeki öğe yoksa öğeyi kaldırır.
<code>ListIterator<E> listIterator();</code>	<code>ListIterator<E> listIterator()</code> ; Listedeki öğelerin liste yineleyicisini döndürür.
<code>List<E> subList(int fromIndex, int toIndex);</code>	Bu yöntem belirtilen dizinler arasındaki öğelerin listesini döndürür.

ArrayList'i açıklayın?

ArrayList, Liste arayüzünü genişleten ve koleksiyon arayüzünü uygulayan sıralı bir koleksiyondur. ArrayList'i esas olarak listedeki öğelerin daha hızlı erişimine ve hızlı yinelenmesine ihtiyaç duyduğumuzda kullanırız. ArrayList'e boş değerler ekleyebiliriz. ArrayList büyütülebilir bir diziden başka bir şey değildir.

Avantajları:

- 1) Daha hızlı ve daha kolay erişim.
- 2) Öğelere rastgele erişim için kullanılır.

Dezavantajları:

- 1) Listenin ortasından eleman ekleyemiyoruz veya silemiyoruz.

Array ve ArrayList arasındaki fark nedir?

Diziler, aynı türdeki ilkelleri veya nesneleri veya aynı türün alt sınıfları olan değişkenleri depolamak için kullanılır.

ArrayList: Dinamik olarak büyüyen sıralı bir koleksiyondur. Listeye boş değerler ekleyebiliriz ve liste yinelenen öğelere izin verir.

Sıralamak	Dizi Listesi
Dizi oluştururken boyutunu bilmemiz gerekiyor.	Ancak ArrayList oluşturulurken boyutun bilinmesine gerek yoktur çünkü arraylist dinamik olarak büyümektedir.
Diziye bir öğe koymak için aşağıdaki sözdizimini kullanırız: String dizi[] = yeniDize[5];dizi[1] = "java";Diziye bir öğe eklemek için belirli konumu bilmeliyiz. Eğer elementi aralık dışında olan bir indekse koymaya çalışırsak arrayIndexOutOfBoundsException Exception ile karşılaşırız.	ArrayList'e aşağıdaki söz dizimi ile öğe ekleyebiliriz: List<String> stringList = new ArrayList<String>();stringList.add("java");
Diziler statiktir	ArrayList dinamiktir
Nesneleri ve ilkelleri saklayabiliriz	Yalnızca 1.5'tan önceki ilkelleri saklayabiliriz. 1.5'ten itibaren nesneleri bile saklayabiliriz.
) Eleman ekleme ve çıkarma mantığını manuel olarak yazmamız gerekiyor.	Yalnızca bir yöntem çağırısı, listedeki öğeleri ekler veya listeden çıkarır.
Diziler daha hızlı	ArrayList daha yavaştır.
	ArrayList diziler kullanılarak uygulanır

Vektör nedir?

Vektör, rastgele erişim için kullanılan arraylist'e benzer.

Vector, arraylist gibi dinamik bir dizidir.

Öğeler eklendiğinde ve çıkarıldığında boyut artar veya azalır.

Vektör senkronize edilir.

ArrayList ve vektör arasındaki fark?

Hem ArrayList hem de vektör dinamik olarak büyür.

ArrayList ve vektör arasındaki farklar şunlardır:

- 1) Dizi listesi senkronize edilmez ve vektör senkronize edilir.
- 2) Performans açısından vektör yerine arraylist kullanılması önerilir çünkü varsayılan olarak vektör senkronize edilir, bu da yalnızca bir iş parçasının erişmesi durumunda performansı azaltır.

Kümeler hakkında bilgi verir misiniz?

Küme, kopyalara izin vermeyen bir koleksiyondur. Set, kopyalara izin vermeyen equals() yöntemini dahili olarak uygular. Bir kümeye yinelenen bir öğenin eklenmesi göz ardı edilir. Set arayüzü java.util.set paketinde uygulanır.

Set arayüzünün herhangi bir ek metodu yoktur. Sadece toplama yöntemleri vardır. Bir küme en fazla bir boş değer içerebilir. ArrayList sıralı bir koleksiyondur. Arraylistlerde sıra, eklendikleri yerle aynı kalır. Ancak bunu ayarlamaya sırasız bir koleksiyon geliyor.

Sette yapılabilecek önemli işlemler:

- 1) Ayarlanacak bir öğenin eklenmesi.
- 2) Bir elemanın setten çıkarılması.
- 3) Kümede bir elemanın olup olmadığını kontrol edin.
- 4) Küme boyunca yineleme.

Set arayüzünün uygulamaları?

- 1) HashSet
- 2) Bağlantılı HashSet
- 3) Ağaç Seti

Java'daki Harita arayüzünü açıklayın?

Harita, anahtar/değer çiftlerinin birleşimidir. Haritadaki hem anahtarlar hem de değerler nesnelerdir.

Haritanın özellikleri:

- 1) Haritalar yinelenen anahtarlara sahip olamaz ancak yinelenen değer nesnelerine sahip olabilir.
- 2) Bir haritadan değer elde etmek için anahtarı bilmeniz gerekir.

Harita Arayüzü Uygulamaları:

- 1) Hash Haritası
- 2) Karma Tablo
- 3) Ağaç Haritası

Java'da arıza hızlı sistem nedir?

Bir sorun oluştuğunda hızlı arıza yapan sistem anında arızalanır. Java'da bunu yineleyicilerin davranışıyla bulabiliriz. Bir koleksiyon nesnesindeki bir koleksiyon nesnesinde bir yineleyici çağırırsanız ve diğer iş parçacığı koleksiyon nesnesini değiştirmeye çalışırsa, eşzamanlı değişiklik istisnası atılır. Bu, hızlı aramadır.

Örn: ArrayList, Vektör, HashMap

Java'da arıza korumalı sistem nedir?

Arızaya Karşı Korumalı yineleyiciler, koleksiyon yinelenirken değiştirilirse herhangi bir istisna oluşturmaz. Çünkü gerçek koleksiyonu değil, koleksiyonun klonunu yineliyorlar.

Örn: ConcurrentHashMap

24. İstisnalar - İstisnalar nelerdir ve onlarla nasıl çalışırız?

İstisnalar:

Java'da iki tür hata vardır.

Derleme zamanı hataları ve Çalışma zamanı hataları.

Derleme zamanı hataları, derleme sırasında rapor edilecek hatalardır.

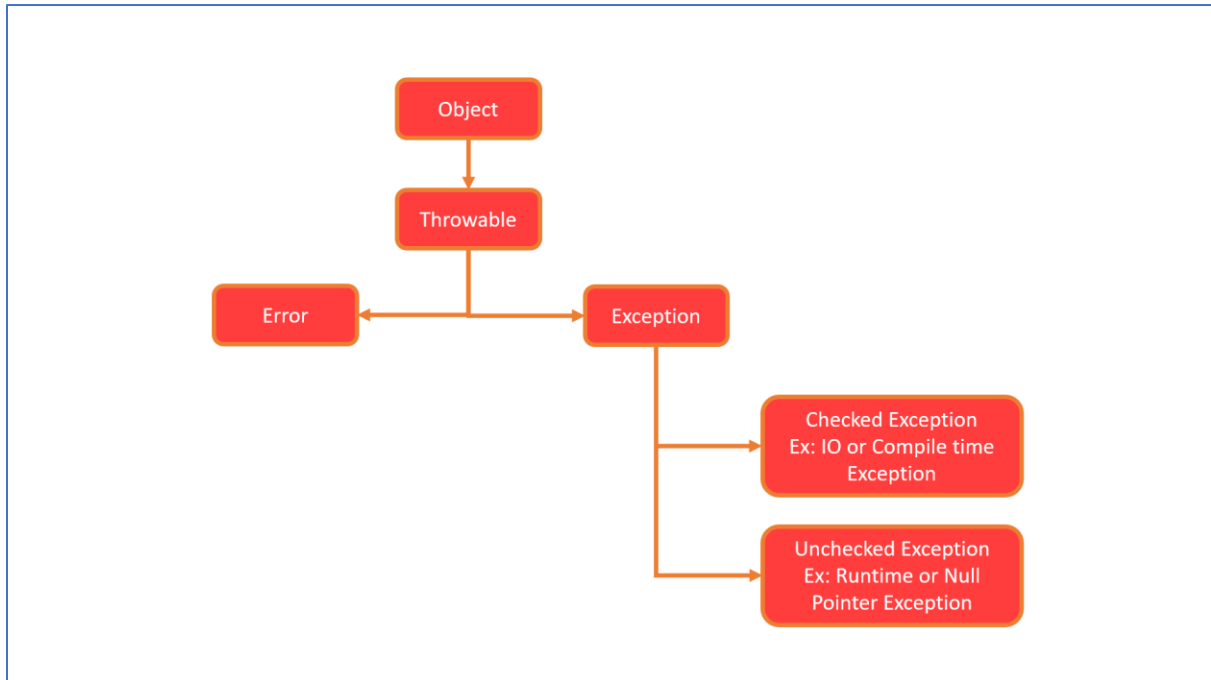
Örnek:

Bir tamsayı değişken bildirirken "int a" yerine "in a" bildirirseniz bu bir derleme zamanı hatası olacaktır.

Çalışma zamanı hatalarına istisna denir.

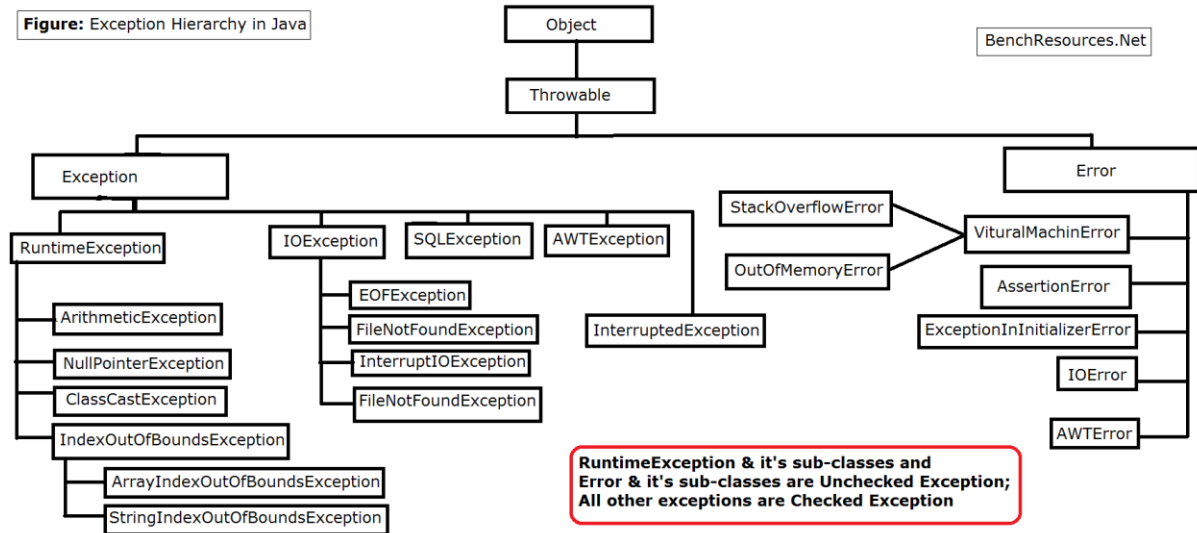
Örnek:

Bir şeyi sıfıra bölmek ArithmeticException'ı oluşturacaktır.



Hata ve İstisna, temel sınıf olan Throwable sınıfının alt sınıflarıdır.

Figure: Exception Hierarchy in Java



java.lang.Atılabilir:

- Throwable, istisnanın kök sınıfıdır ve alt türü, hatası ve alt türleri
- Başka bir deyişle, istisna ve hata için süper sınıftır
- Java.lang.Throwable sınıfı, Java.lang.Object sınıfını genişletir (yukarıdaki şekilde gösterildiği gibi)
- 2 alt sınıfı tanımlar; İstisna ve Hata

java.lang.İstisna:

- java.lang.Exception tüm İstisna türleri için süper sınıftır
- Java.lang.Throwable sınıfını genişletir
- İstisna programatik mantıktan kaynaklanmaktadır
- Ve kurtarılabılır
- İstisna, kontrol edilen istisna ve denetlenmeyen istisna olarak kategorize edilir
- Örnek: RuntimeException, SQLException, IOException, FileNotFoundException, ArithmeticException, NullPointerException

java.lang.Hata:

- java.lang.Error her türlü Hata için süper sınıftır
- Java.lang.Throwable sınıfını genişletir
- Hata, sistem kaynaklarının eksikliğinden kaynaklanıyor
- Ve kurtarılamaz
- Tüm hatalar, çalışma zamanında sistem kaynaklarının yetersizliği nedeniyle ortaya çıktığı için denetlenmeyen istisna kategorisine girer.
- Bu tür bir hata tahmin edilemediğinden programlama kapsamı dışındadır, iyi planlanmış olabilir, bu tür Hatalardan kaçınmak için özen gösterilebilir
- Örnek: VirtualMachineError, AssertionError, ExceptionInInitializerError, StackOverflowError, OutOfMemoryError, LinkageError, InstantiationException

Java'da istisnalar nasıl ele alınır:

Java İstisna yönetimi beş anahtar kelimeyle yönetilir: dene, yakala, sonunda fırlat ve fırlat.

dene – yakala – sonunda:

Belirli bir kod parçasının istisna attığını düşünüyorsak, bunu try bloğuna koyarız ve istisnayı yakalayacak kodu catch bloğuna yazarız.

Try catch bloğu tamamlandıktan sonra çalıştırılması gereken kodlar, nihayet bloğuna konur.

Try catch bloğunu kullanarak işlediğimiz istisnalara denetlenmeyen istisnalar denir.

Örn: ArithmeticException, ArrayIndexOutOfBoundsException vb.

Atar:

Bazen bazı yöntemlerin bazı istisnalar attığını biliyoruz ve bunlar, bir sınıfı tanımlarken throws anahtar kelimesini kullanarak çözülebilir. Bunlar işaretlenmiş istisnalardır.

Örn: IOException, FileNotFoundException, SQLException vb.

Fırlatmak:

Java'da Throw anahtar sözcüğünü kullanarak kendi istisnalarımızı tanımlayabiliriz. Bunlar, kullanıcı tanımlı veya özel istisnalar olarak bilinir.

İstisnalar Nasıl Ele Alınır:

Try Catch Bloğunu Kullanma:

Sözdizimi:

```
denemek{
İfadeler;
}
catch(İstisna e){
İfadeler;
}
Sonunda{
İfadeler;
}
```

Bir istisnanın ortaya çıkıp çıkmamasına bakılmaksızın yürütülecek olan nihayet bloğunu kullanabilirsiniz. Bu isteğe bağlıdır.

Örnek:

```
paket FPPaket;

halk sınıf StringsDemo {

    halk statik void main(String[] args ) {

        int dizi1 [] = {10,20,30};
        int dizi2 [] = {2,5,10};
        için ( int ben =0; ben <3; ben ++ ) {
            için ( int j =0; j <3; j ++ ) {
                Sistem. out .println( dizi1 [ i ]/ dizi2 [ j ] );
            }
        }
    }
}
```

```

    }
}

```

Yukarıdaki örnekte arr1'in her elemanını arr2'nin her elemanına bölüyoruz ve sonuçları yazdırıyoruz. Ancak arr2'nin herhangi bir değeri 0 ise ArithmeticException'ı oluşturur.

Eski:

```

paket FPPaket;

halk sınıf StringsDemo {

    halk statik void main(String[] args ) {

        int dizi1 [] = {10,20,30};
        int dizi2 [] = {2,5,0};
        için ( int ben =0; ben <3; ben ++ ) {
            için ( int j =0; j <3; j ++ ) {
                Sistem. out .println( dizi1 [ i ]/ dizi2 [ j ] );
            }
        }
    }
}

```

5 ve 2'yi yazdıracak ve ardından aritmetik istisnayı rapor edecektir.

Bu tür istisnaları ele almak için try catch bloklarını kullanıyoruz ve devam edeceğiz.

24A. İstisnalar - Blokları Yakalamayı Deneyin

Yakalama Bloğu'nu deneyin:

Try catch blokları istisnaları yakalamak için kullanılır.

```

paket paket1;

halk sınıf İstisnaÖrneği {

    halk statik void main(String[] args ) {

        int dizi1 [] = {10,20,30};
        int dizi2 [] = {2,0,10};
        için ( int ben =0; ben <3; ben ++ ) {
            için ( int j =0; j <3; j ++ ) {
                denemek {
                    Sistem. out .println( dizi1 [ i ]/ dizi2 [ j ] );
                }
                catch (İstisna e ) {
                    Sistem. out .println( e );
                }
            }
        }
    }
}

```

24B. İstisnalar - Anahtar Kelimeyi Atar

Anahtar Kelimeyi Atar:

Throws anahtar sözcüğü, bir yöntemin istisna atabileceğini bildirmek için kullanılır.

Örneğin bir dosya oluştururken IOException oluşabilir.

Bunlar işaretlenmiş istisnalardır. Derleyici bir istisnanın oluşabileceğini biliyor ve bu yüzden onları yakalamamız gerekiyor ve derleyici bu istisnaları yakalamamız gerektiğine dair bir hata veriyor ve bu istisnalara kontrol edilen istisnalar adı veriliyor. İşaretlenen istisnalar, try catch bloğu veya throws anahtar sözcüğüyle ele alınabilir.

```
paket FPPaket;

içe aktarın ;
içe aktarın ;

halk sınıf StringsDemo {

    halk statik void main(String[] args ) {
        denemek {
            FileWriter dosyası = new FileWriter( "K:\\Data1.txt" );
            dosya .write( "Subbu" );
            dosya .close();
        }
        yakala (IOException e ){
            Sistem. dışarı . println( "Bazı hatalar oluştu." );
        }
    }
}
```

Sistemimde K sürücüsü yok bu yüzden IOException'ı atacak. Ama try-catch bloklarıyla yakalıyoruz.

Bunun gibi çok fazla istisna varsa, o zaman çok sayıda catch bloğu yazmamız gerekir ve kod hantal hale gelir, bu yüzden bunu önlemek için throws anahtar kelimesini kullanırız ve beklediğimiz tüm istisnalardan bahsederiz.

```
paket FPPaket;

içe aktarın ;
içe aktarın ;

halk sınıf StringsDemo {

    halk statik void main(String[] args ) IOException { ' i atar
    FileWriter dosyası = new FileWriter( "K:\\Data1.txt" );
        dosya .write( "Guru99" );
        dosya .close();
    }
}
```

24C. İstisnalar - Anahtar Kelimeyi Atın

Anahtar Kelimeyi Atın:

Java'da throw anahtar sözcüğü, bir yöntemden veya kod bloğundan açıkça bir istisna atmak için kullanılır. Kontrol edilen bir istisna veya kontrol edilmeyen bir istisna atabiliriz.

Eski:

```
Throw new ArithmeticException("Üzgünüm! Mümkün değil");
```

Örnek:

```

halk sınıf istisnası {

    statik geçersiz doğrulama ( int) yaş ) {
        eğer ( yaş <18) {
            denemek {
                fırlatmak new ArithmeticException( "geçerli değil" );
            }
            catch (ArithmeticException e) {
                Sistem. out .println( e );
            }
        }
        başka
        Sistem. out .println( "oy vermeye hoş geldiniz" );
    }
    halk statik void main(String[] args ) {
        doğruLa (8);
        Sistem. out .println( "kodun geri kalanı" );
    }
}

```

24D. İstisnalar - Soruları

Java'da bir istisna nedir?

Bir istisna, çalışma zamanı hatasıdır. Java'da istisna bir nesnedir. Programımızda anormal durumlar ortaya çıktığında istisnalar yaratılır. İstisnalar JVM veya uygulama kodumuz tarafından oluşturulabilir. Tüm İstisna sınıfları Java.lang'da tanımlanmıştır.

Java'da istisnaların ortaya çıkabileceği bazı durumları belirtin?

- 1) Dizide bulunmayan bir öğeye erişim.
- 2) Bir tamsayıyı sıfıra bölmek (Aritmetik istisna)
- 2) Sayının dizeye ve dizinin sayıya dönüştürülmesi geçersiz. (NumberFormatException)
- 3) Geçersiz sınıf seçimi (Sınıf dönüşümü İstisnası)
- 4) Arayüz veya soyut sınıf için nesne oluşturmaya çalışmak (Örnekleme İstisnası)

Java'da İstisna yönetimi nedir?

İstisna yönetimi, programda anormal bir durum ortaya çıktığında ne yapılacağına ilişkin bir mekanizmadır. Programda bir istisna ortaya çıktığında, bu durum düzgün şekilde yönetilmediğinde programın sonlandırılmasına neden olur. Bir programın aniden sonlandırılmaması ve programın geri

kalanına normal şekilde devam edilebilmesi için istisna işlemenin önemi burada ortaya çıkar. Bu, İstisna yönetiminin yardımıyla yapılabilir.

Java'da istisna işlemeyi kaç farklı şekilde yapabiliriz?

İstisnaları iki yoldan biriyle ele alabiliriz:

- 1) İstisnayı yakalayabileceğimiz try catch bloğunu belirterek.
- 2) throws cümleciğiyle bir yöntem bildirmek.

İstisna işlemeyle ilgili beş anahtar kelimeyi listelemek ister misiniz?

- 1) Deneyin
- 2) Yakala
- 3) atmak
- 4) atar
- 5) nihayet

Java'da try and catch anahtar kelimelerini açıklayın?

Try bloğunda istisnaya neden olan tüm kodları tanımlarız. Java'da try ve catch bir birim oluşturur. Catch bloğu, önceki try bloğu tarafından oluşturulan istisnayı yakalar. Catch bloğu başka bir try bloğu tarafından atılan bir istisnayı yakalayamaz. Programımızda koda neden olan bir istisna yoksa veya kodumuzda istisna ortaya çıkmıyorsa, jvm try catch bloğunu yok sayar.

Sözdizimi:

```
denemek {  
}
```

```
Yakala(İstisna e) {  
}
```

Catch bloğu olmadan try bloğunu kullanabilir miyiz?

Her try bloğu, en az bir catch bloğu veya nihayet bloğu gerektirir. catch veya nihayet içermeyen bir try bloğu derleyici hatasına neden olur. Catch'i ya da sonunda bloğu atlayabiliriz ama ikisini birden atlayamayız.

Bir try bloğu için birden fazla catch bloğumuz olabilir mi?

Bazı durumlarda kodumuz birden fazla istisna atabilir. Böyle bir durumda iki veya daha fazla catch cümleciği belirtebiliriz; her bir catch farklı türdeki istisnaları ele alır. Bir istisna oluşturulduğunda, jvm her bir catch ifadesini sırayla kontrol eder ve istisna türüyle eşleşen ilk ifade yürütme olur ve kalan catch blokları atlanır.

Try ve catch blokları arasında herhangi bir kod alabilir miyiz?

Try ve catch bloğu arasında herhangi bir kod bildirmemeliyiz. Catch bloğu try bloğundan hemen sonra başlamalıdır.

Java'da nihayet bloğun önemini açıklayın?

Son olarak blok, bağlantıları, yuvaları vb. kapatmak için kullanılır. Eğer try bloğu istisnasız olarak yürütülürse, try bloğundan sonra catch bloğu yürütülmeden çağrılır. Try bloğunda bir istisna atılırsa, sonunda blok, catch bloğundan hemen sonra yürütülür. Bir istisna atılırsa, no catch bloğu istisnayı ele alsa bile, nihayet blok yürütülür.

Tek catch bloğunda birden fazla istisna yakalayabilir miyiz?

Java 7'den itibaren tek bir catch bloğu ile birden fazla istisnayı yakalayabiliyoruz. Bu tür bir işlem kod tekrarını azaltır.

Not: Tek bir catch bloğunda birden fazla istisna yakaladığımızda, catch parametresi implicitly final olur. Catch parametresine herhangi bir değer atayamayız.

Örn: `catch(ArrayIndexOutOfBoundsException | ArithmeticException e) {`
`}`

Yukarıdaki örnekte e finaldir, catch ifadesinde herhangi bir değer atayamayız veya e'yi değiştiremeyiz.

Örnek:

```
paket paket1;

halk sınıf İstisnaÖrneği {

    halk statik void main(String[] args ) {

        int dizi1 [] = {10,20,30};
        int dizi2 [] = {2,0,10};
        için ( int ben =0; ben <3; ben ++ ) {
            için ( int j =0; j <4; j ++ ) {
                denemek {
                    Sistem. out .println( dizi1 [ i ]/ dizi2 [ j ] );
                }
                catch (ArrayIndexOutOfBoundsException |
ArithmeticException e ) {
                    Sistem. out .println( e );
                    //
                    //catch(ArithmeticException e) {
                    //    System.out.println(e);
                    // }
            }
        }
    }
}
```

İşaretlenen istisnalar nelerdir?

1) Throwable sınıfının hata, Çalışma Zamanı istisnası ve alt sınıfları dışındaki tüm alt sınıfları kontrol edilen istisnalardır.

2) Kontrol edilen istisna, anahtar kelime atmalarla atılmalı veya try catch bloğu sağlanmalıdır, aksi takdirde program derlenmez.

Derleme hatası alıyoruz.

Örnekler:

- 1) IOException,
- 2) SQLException,
- 3) FileNotFoundException,
- 4) InvokasyonTargetException,
- 5) CloneNotSupportedException
- 6) ClassNotFoundException
- 7) Örnekleme İstisnası

Java'da denetlenmeyen istisnalar nelerdir?

RuntimeException'in tüm alt sınıflarına denetlenmeyen istisnalar adı verilir. Bunlar denetlenmeyen istisnalardır çünkü derleyici bir yöntemin istisnaları işleyip işlemediğini veya atıp atmadığını kontrol etmez. İstisnayı yakalayamasak veya istisnayı atsak bile program derlenir. Programda bir istisna meydana gelirse program sonlandırılır. Bu istisnaları ele almak zordur çünkü istisnalara neden olan birçok yer olabilir.

Örnek :

- 1) Aritmetik İstisna
- 3) ArrayIndexOutOfBoundsException
- 4) ClassCastException
- 5) IndexOutOfBoundsException
- 6) NullPointerException
- 7) NumberFormatException
- 8) StringIndexOutOfBoundsException
- 9) Desteklenmeyen Operasyon İstisnası

Java'da işaretli ve işaretsiz istisnalar arasındaki farkları açıklayın?

Denetlenmeyen İstisnalar	Kontrol Edilen İstisnalar
RuntimeException'in tüm alt sınıflarına denetlenmeyen istisna adı verilir.	Throwable sınıfının RuntimeException dışındaki tüm alt sınıfları kontrol edilen istisnalar olarak çağrılır
derleme zamanında ele alınmasına gerek yoktur	İşaretli istisnaların derleme zamanında ele alınması gerekir.
programımızdaki kodlama hatalarından kaynaklanmaktadır .	
ArrayIndexOutOfBoundsException, ClassCastException, IndexOutOfBoundsException	SQLException, FileNotFoundException, ClassNotFoundException

Java'da varsayılan istisna yönetimi nedir?

JVM istisnaya neden olan kodu tespit ettiğinde aşağıdaki bilgileri ekleyerek yeni bir istisna işleme nesnesi oluşturur.

- 1) İstisnanın Adı
- 2) İstisnaya İlişkin Açıklama
- 3) İstisna Yeri.

JVM tarafından nesne oluşturulduktan sonra istisna işleme kodunun olup olmadığı kontrol edilir. İstisna işleme kodu varsa, istisna işlenir ve programa devam edilir. Herhangi bir istisna işleme kodu yoksa JVM, istisna işleme sorumluluğunu varsayılan işleyiciye verir ve aniden sona erer. Varsayılan İstisna işleyicisi, istisnanın açıklamasını görüntüler, istisnanın yığın izlemesini ve konumunu yazdırır ve programı sonlandırır.

Not: Bu varsayılan istisna işlemenin ana dezavantajı programın aniden sona ermesidir.

Örnek:

```
paket paket1;

halk sınıf İstisnaÖrneği {

    halk statik void main(String[] args ) {

        int dizi1 [] = {10,20,30};
        int dizi2 [] = {2,0,10};
        için ( int ben =0; ben <3; ben ++ ) {
            için ( int j =0; j <3; j ++ ) {
                Sistem. out .println( dizi1 [ i ]/ dizi2 [ j ] );
            }
        }
    }
}
```

Sonuç:

```
İş parçacığı "ana"
java.lang.ArithmeticException : / sifıra yakın
package1.ExceptionExample.main'de ( ExceptionExample.java:11 )
```

Java'da try ifadelerini iç içe yerleştirebilir miyiz?

Evet try ifadeleri iç içe yerleştirilebilir. Başka bir try ifadesinin bloğunun içinde try ifadelerini bildirebiliriz.

Fırlatılabilir sınıfın önemini ve yöntemlerini açıklayın?

Fırlatılabilir sınıf, İstisnaların kök sınıfıdır. Tüm istisnalar bu atılabilir sınıftan türetilir. Fırlatılabilir'in iki ana alt sınıfı İstisna ve Hata'dır.

Fırlatılabilir sınıfta tanımlanan üç yöntem şunlardır:

- 1) void printStackTrace():
Bu, istisna bilgilerini aşağıdaki formatta yazdırır: İstisnanın adı, açıklama ve ardından yığın izleme.

- 2) getMessage()
Bu yöntem yalnızca İstisna'nın açıklamasını yazdırır.
- 3) toString():
İstisnanın adını ve açıklamasını yazdırır.

25. Java Dosya İşlemleri - Bölüm 1 - Java Giriş Çıkış ve Dosya Yolu İşlemleri

Java I/O, girdiyi işlemek ve çıktı üretmek için kullanılır. Bir giriş akışı dosyayı okur ve işlemdeki verileri saklar. Çıkış akışı işlemde okur ve hedef dosyaya yazar.

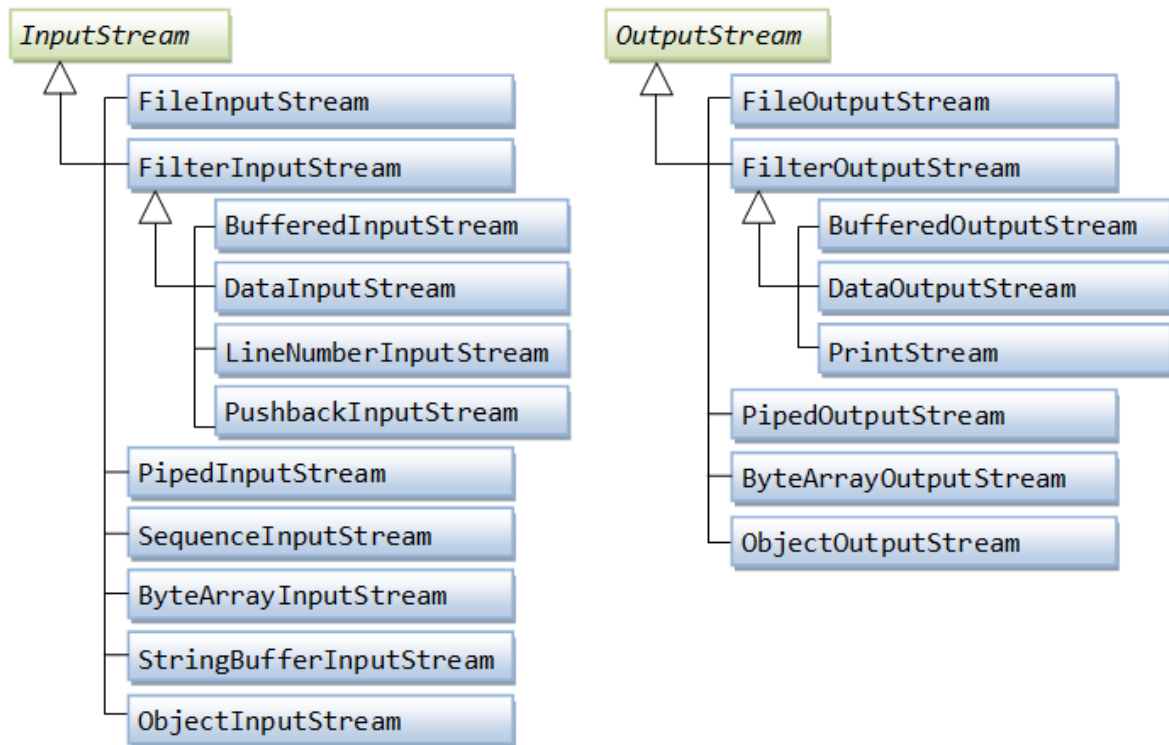
İki tür akış vardır: Bayt Akışları ve Karakter Akışları

Bayt akışları JDK 1.0 ile tanıtıldı ve ASCII karakterleri içeren dosyalar üzerinde çalışıyor. Java'nın Unicode karakterleri olarak da bilinen diğer dil karakterlerini desteklediğini biliyoruz. Tasarımcılar, Unicode karakterleri içeren dosyaları okumak için JDK 1.1 ile karakter akışlarını tanıttı. ASCII, Unicode'un bir alt kümesi olduğundan, İngilizce karakterlerin dosyaları için bayt akışları veya karakter akışları ile gidebiliriz .

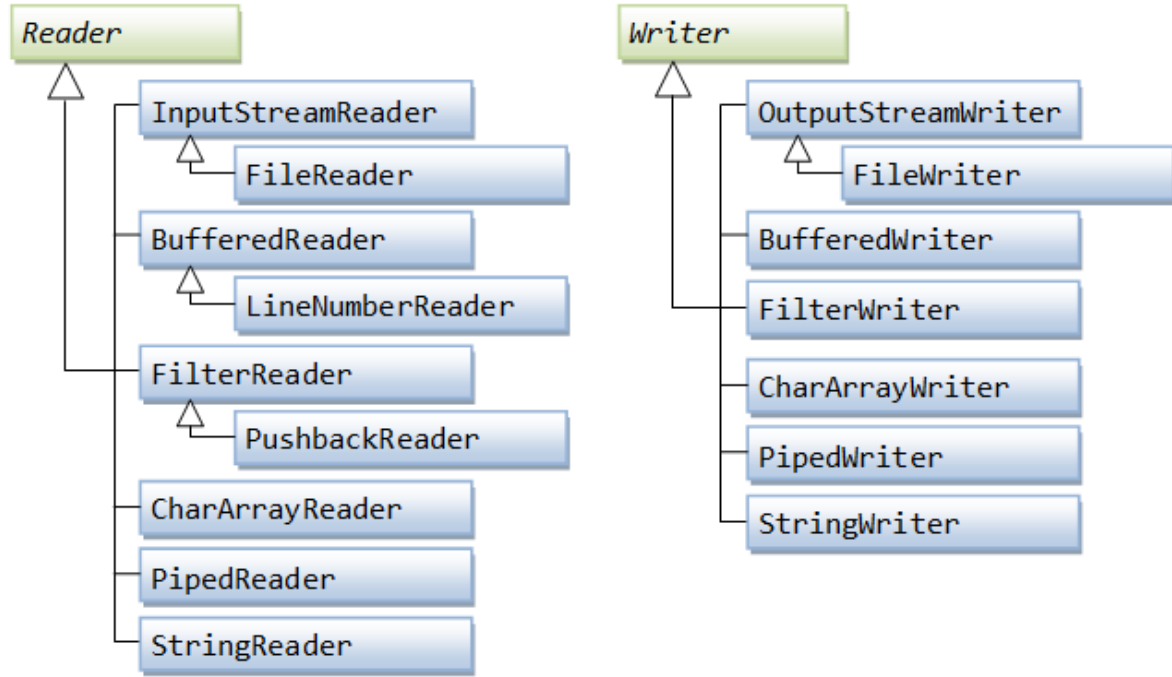
Tüm Bayt akışları iki türe ayrılabilir: Giriş Akışı ve Çıkış Akışı

Tüm Karakter akışları iki türe ayrılabilir: Okuyucu ve Yazıcı.

Bayt G/Ç Akışları:



Karakter G/Ç Akışları:



Java'da Dosya/Dizin İşlemleri:

Dosya/Dizin işlemleri, dizinde bir dosyanın bulunup bulunmadığını kontrol etmek, yeni bir dizin oluşturmak, yeni bir dosya oluşturmak vb. gibi dosya okuma veya yazma dışındaki işlemlerdir.

Bu işlemler Java.io.File sınıfı tarafından sağlanır.

Bu yöntemleri aşağıdakilere ayırabiliriz.

1. Dosya sistemini sorgulayan yöntemler
2. Dosya sistemini değiştiren yöntemler

Dosya sistemini sorgulayan yöntemlerden bazıları şunlardır:

1. canRead()
2. yazabilir()
3. var()
4. isDirectory()
5. isFile()
6. Gizlidir()
7. getAbsolutePath()
8. son düzenleme()
9. uzunluk()
10. listeDosyaları()
11. listeKökler()

Dosya sistemini değiştiren yöntemlerden bazıları şunlardır:

1. yeniDosya oluştur()
2. mkdir()
3. yeniden adlandır()
4. silmek()

5. deleteOnExit()
6. setReadOnly()
7. setLastModified()

25A. Java Dosya İşlemleri - Bölüm 2 - Java Dosyası ve Dizin Yolu İşlemleri Örneği

Dosya/Dizin işlemlerini kontrol etme örneği:

```
paket FPPaket;

içe aktarın ;
içe aktarın ;

halk sınıf CheckingFiles {

    halk statik void main(String[] args ) IOException {' i atar
        Dosya tmpDir = new Dosya( "D:\\Selenium Course\\FileOperations" );
        Dosya tmpFile = yeni Dosya( tmpDir + "\\Sample.txt" );

        if ( tmpDir .isDirectory()) {
            Sistem. out .println( "Bu bir dizindir" );
        }

        if ( tmpDir .exists()) {
            Sistem. out .println( "Dizin mevcut" );
        }

        if ( tmpFile .isFile()) {
            Sistem. out .println( "Bu dosyadır" );
        }

        if ( tmpFile .exists()) {
            Sistem. out .println( "Dosya mevcut" );
        }

        if ( tmpFile .canRead()) {
            Sistem. out .println( "Dosya okunabilir" );
        }
        if ( tmpFile .canWrite()) {
            Sistem. out .println( "Dosya yazılabilir" );
        }

        if ( tmpFile .isHidden()) {
            Sistem. out .println( "Dosya gizlidir" );
        }
        başka {
            Sistem. out .println( "Dosya gizli değil" );
        }

        Dosya yolu = tmpFile .getAbsolutePath();
        Sistem. out .println( yol );

        Dosya dosyası = yeni Dosya( tmpDir + "\\Sample1.txt" );

        if ( dosya .createNewFile()) {
            Sistem. out .println( "Dosya başarıyla oluşturuldu." );
        }
    }
}
```



```

    }

    Dosya dizini = yeni Dosya( tmpDir + "\\test" );
    if ( dir .mkdir()) {
        Sistem. out .println( "Dizin Başarıyla Oluşturuldu" );
    }
}

}

```

25B. Java Dosya İşlemleri - Bölüm 3 - FileWriter ve FileReader

Dosya Yazarı:

dosyaya yazmak için kullanılır . Java'da dosya işleme için kullanılan karakter odaklı sınıftır .

dizeyi doğrudan yazma yöntemi sağladığı için dizeyi bayt dizisine dönüştürmenize gerek yoktur .

Örnek:

```

paket FPPaket;

içe aktarın ;
içe aktarın ;

halk sınıf FileWriterExample {

    halk statik void main(String[] args ) IOException { ' i atar
        FileWriter fw = new FileWriter( "C:\\JavaExamples\\test.txt" );
        String s = "Merhaba Merhaba Merhaba" ;
        fw .write( s );
        fw .close();
        Sistem. out .println( "Başarılı!" );
    }
}

```

Dosya Okuyucusu:

Dosyadan veri okumak için Java FileReader sınıfı kullanılır. Verileri FileInputStream sınıfı gibi bayt biçiminde döndürür .

Java'da dosya işleme için kullanılan karakter odaklı sınıftır .

Örnek:

```

paket FPPaket;

içe aktarın ;

halk sınıf DosyaOkuyucuÖrneği {

    halk statik void main(String[] args ) {
        denemek {
            FileReader fr = new FileReader( "C:\\JavaExamples\\test.txt"
        );
    }
}

```

```

        int Ben ;
        while (( i = fr .read())!= -1){
            Sistem. out .print( ( char ) i );
        }
        fr .close();
    }
    catch (İstisna e ) {
        Sistem. out .println( e );
    }
}
}
}

```

25C. Java Dosya İşlemleri - Dosya Giriş ve Çıkış Soruları

Akış nedir ve Akış türleri ve Akış sınıfları nelerdir?

Akış, bir veri dizisidir. Java'da bir akış baytlardan oluşur. Akarsu olarak adlandırılmasının nedeni akmaya devam eden bir su akıntısına benzemesidir. İki tür Akış vardır:

Bayt Akışları: Baytların giriş ve çıkışını yönetmek için uygun bir yol sağlar.

Karakter Akışları: Karakterlerin giriş ve çıkışını yönetmek için uygun bir yol sağlar.

Byte Streams sınıfları: GirişStream ve OutputStream olmak üzere iki soyut sınıf kullanılarak tanımlanır.

Karakter Akışı sınıfları: Reader ve Writer olmak üzere iki soyut sınıf kullanılarak tanımlanır.

GÇ akışı nedir?

Kaynaktan hedefe doğru akan bir veri akışıdır. Bunun iyi bir örneği dosya kopyalamadır. İki akış söz konusudur - giriş akışı ve çıkış akışı. Bir giriş akışı dosyadan okur ve süreçteki verileri (genellikle geçici bir değişkende) saklar. Çıkış akışı işlemiden okur ve hedef dosyaya yazar.

Bayt akışları ve karakter akışları olmak üzere iki tür akışın gerekliliği nedir?

Bayt akışları JDK 1.0 ile tanıtıldı ve ASCII karakterleri içeren dosyalar üzerinde çalışıyor. Java'nın Unicode karakterleri olarak da bilinen diğer dil karakterlerini desteklediğini biliyoruz. Tasarımcılar, Unicode karakterleri içeren dosyaları okumak için JDK 1.1 ile karakter akışlarını tanıttı. ASCII, Unicode'un bir alt kümesi olduğundan, İngilizce karakterlerin dosyaları için bayt akışlarını veya karakter akışlarını kullanabiliriz.

Tüm akışların en süper sınıfları nelerdir?

Tüm bayt akışı sınıfları iki kategoriye (giriş akışı sınıfları ve çıkış akışı sınıfları) ve tüm karakter akışı sınıfları da iki kategoriye (okuyucu sınıfları ve yazar sınıfları) ayrılabilir. Tüm bu akışların türetildiği dört soyut sınıf vardır. Tüm bayt akışı sınıflarının en üst sınıfı Java.io.InputStream'dir ve tüm çıktı akışı sınıfları için Java.io.OutputStream'dir. Benzer şekilde tüm okuyucu sınıfları için Java.io.Reader ve tüm yazar sınıfları için Java.io.Writer'dir.

Reader/Writer sınıfı hiyerarşisi ile OutputStream/OutputStream sınıfı hiyerarşisi arasındaki fark nedir?

Reader/Writer sınıfı hiyerarşisi karakter odaklıdır ve OutputStream/OutputStream sınıf hiyerarşisi bayt odaklıdır

Bir dosyanın sonuna ulaşıldığında read() hangi değeri döndürür?

Read () yöntemi , bir dosyanın sonuna ulaştığında -1 değerini döndürür .

ObjectInputStream sınıfının paket adı nedir?

java.io

StringBufferInputStream sınıfının Ana Sınıfı hangisidir?

Giriş Akışı

Not: GirişStream ile biten her şey, GirişStream'in alt sınıflarıdır.

OutputStream ile biten her şey OutputStream'in alt sınıflarıdır.

Reader ile biten her şey Reader'ın alt sınıflarıdır.

Writer ile biten her şey Writer'ın alt sınıflarıdır.

FileInputStream ve FileOutputStream nedir?

Bu ikisi, programcının dosyayı dosyaya kopyalamak için sıklıkla kullandığı genel amaçlı sınıflardır. Bu sınıflar, performans açısından çok zayıf olduğundan, birkaç bin bayttan daha az veri içeren dosyalarla iyi çalışır. Daha büyük veriler için BufferedInputStream (veya BufferedReader) ve BufferedOutputStream (veya BufferedWriter) kullanılması tercih edilir.

Hangisini kullanmak daha iyi hissediyorsunuz: bayt akışlarını mı yoksa karakter akışlarını mı?

Kişisel olarak karakter akışlarını en yeni oldukları için takip etmeyi düşünüyorum. Karakter akışlarında bayt akışlarında bulunmayan birçok özellik vardır: a) BufferedInputStreams ve DataInputStream yerine BufferedReader kullanmak (iki kişilik bir akış) ve b) sonraki satıra geçmek için newLine() yöntemini kullanmak ve bu etki için gitmemiz gerekenler bayt akışlarında ekstra kodlama vb.

System.out.println() nedir?

"println()", PrintStream sınıfının bir yöntemidir. "out", "System" sınıfında tanımlanan PrintStream sınıfının statik bir nesnesidir. Sistem, programcı tarafından temel işletim sistemiyle etkileşimde bulunmak için kullanılan Java.lang paketindeki bir sınıftır.

Filtre akışları nedir?

Filtre akışları, sorumluluğu mevcut akışlara, kaynak dosyada bulunmayan hedef dosyadaki satır numaralarını vermek veya kopyalama performansını artırmak vb. gibi ekstra işlevsellik (avantaj) eklemek olan bir IO akışları kategorisidir.

Mevcut filtre akışlarını adlandırın mı?

Java.io paketinde dört filtre akışı vardır; ikisi bayt akışı tarafında ve ikisi karakter akışı tarafında. Bunlar FilterInputStream, FilterOutputStream, FilterReader ve FilterWriter'dır. Bu sınıflar soyut sınıflardır ve bu sınıflara ait nesneleri oluşturamazsınız.

Bayt akışının okuma tarafındaki filtre akışı sınıflarını adlandırın mı?

Dört sınıf vardır - LineNumberInputStream (ekstra işlevsellik, hedef dosyaya satır numaraları eklemesidir), DataInputStream (bir int, bir double ve bir dize okuyabilen readInt(), readDouble() ve readLine() vb. gibi özel yöntemler içerir) BufferedInputStream (performansı en üst düzeye çıkaran tamponlama etkisi sağlar) ve PushbackInputStream (gerekli karakteri sisteme geri iter).

SequenceInputStream'in işlevselliği nedir?

Birden fazla kaynak dosyayı tek bir hedef dosyaya çok daha az kodla kopyalamak çok kullanışlıdır.

PrintStream ve PrintWriter nedir?

İşlevsel olarak her ikisi de aynıdır ancak iki farklı kategoriye aittir: bayt akışları ve karakter akışları. `println()` yöntemi her iki sınıfta da mevcuttur.

Dosya kopyalamada maksimum performans elde etmek için hangi akışların kullanılması önerilir?

Bayt akışları tarafında `BufferedInputStream` ve `BufferedOutputStream` ve karakter akışları tarafında `BufferedReader` ve `BufferedWriter`.

Borulu akışlar nelerdir?

Dört adet borulu akış vardır: `PipedInputStream`, `PipedOutputStream`, `PipedReader` ve `PipedWriter`. Bu akışlar, çalışan iki iş parçacığı (örneğin, işlemler) arasında veri aktarmak için çok kullanışlıdır.

Dosya sınıfı nedir?

Bir dosyanın oluşturulduğu (veya değiştirildiği), okuma ve yazma izinlerine sahip olduğu, boyutu vb. gibi özelliklerini bilmek için kullanılan, akış dışı (dosya işlemleri için kullanılmayan) bir sınıftır.

26. Java Veritabanı Bağlantısı - XAMPP MySql Veritabanını Yerel Makineye Yükleme

Uygulama için MySql veritabanını yerel olarak kurmak:

MySQL'i [mysql.com](https://www.mysql.com/) adresinden indirip kurabilirsiniz. Ama biraz karmaşık. Yani MySQL'i indirmek yerine MySQL'in birlikte geldiği yerel bir sunucuyu indirebilirsiniz. İndirebileceğiniz birkaç yerel sunucu var ve bunlardan bazıları XAMPP, WAMPP vb.

MySql kurulumu için XAMPP'yi indirip kuracağız. Bununla Apache sunucusunu ve MySql veritabanını alıyoruz.

Not: Bunu selenyum üzerine bir proje yaptığımızda da kullanırız. Bu yüzden bunu yüklemenizi öneririm.

Google'a gidin ve XAMPP indirmeyi arayın.

XAMPP'yi indirebileceğiniz birçok kaynak bulacaksınız. Aşağıdaki bunlardan biri.

<https://xampp-windows.en.softonic.com/download>

Home > Windows > Development & IT > XAMPP Windows > Download



Download XAMPP Windows

 Compatible with your OS
  Free Download
  In English

Version: 7.2.3

Sign in to start the Download

 Sign in with Facebook
 or
  Sign in with Google +

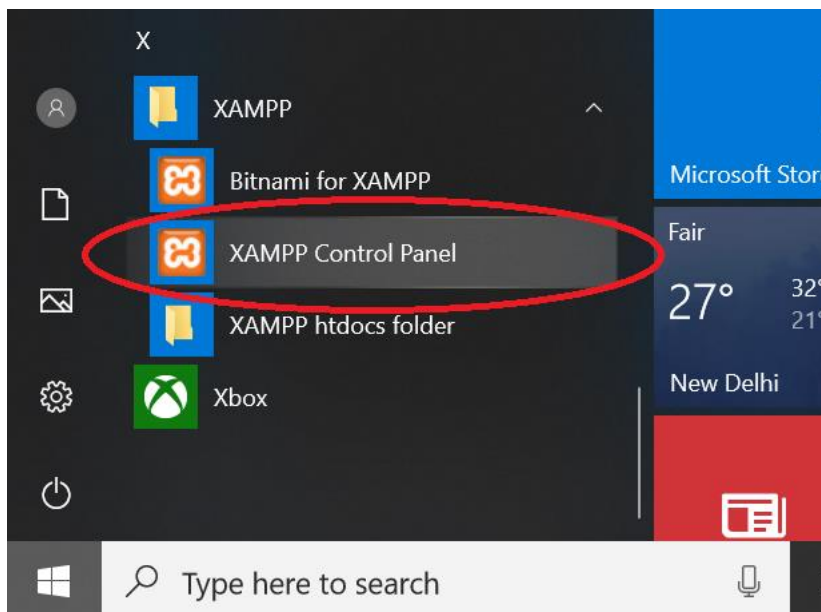
XAMPP Windows free download. **Always available from the Softonic servers**

-  Free & fast download
-  Always available
-  Tested virus-free

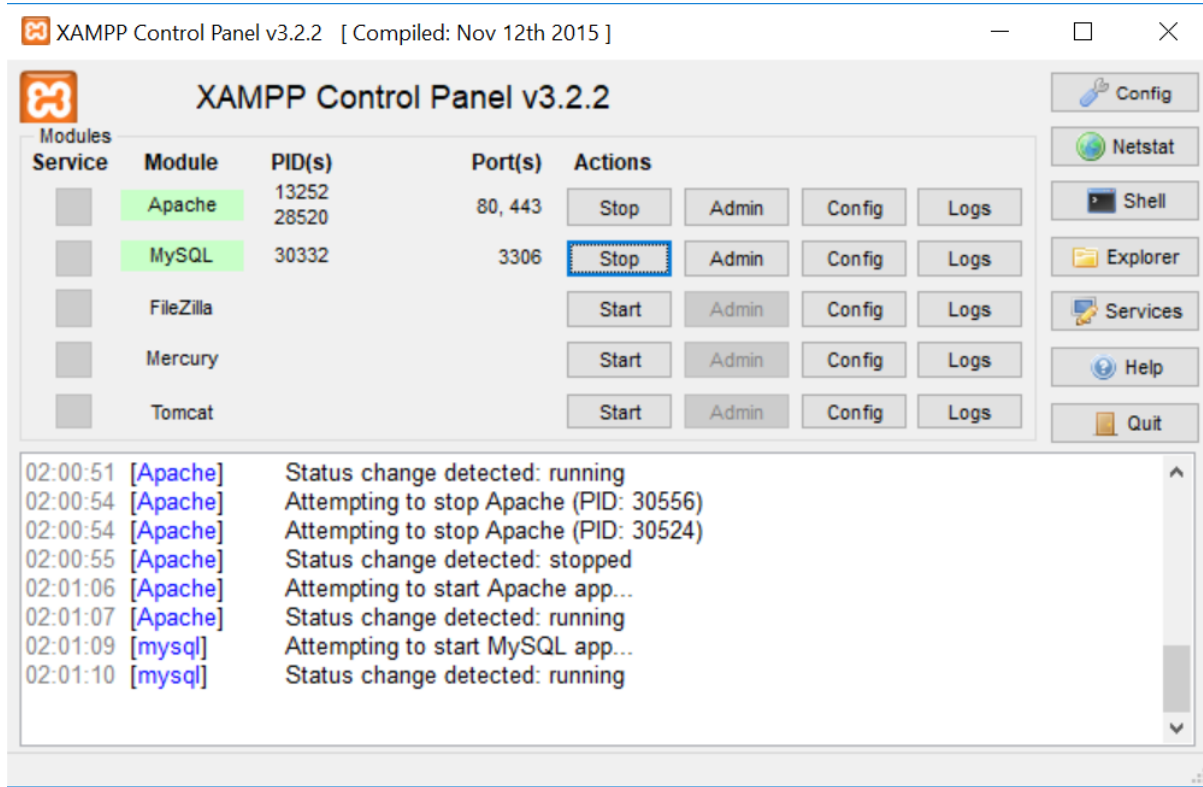
Alternative XAMPP Windows download from external server (availability not guaranteed)



XAMPP'yi indirdikten sonra sisteminize yükleyin.



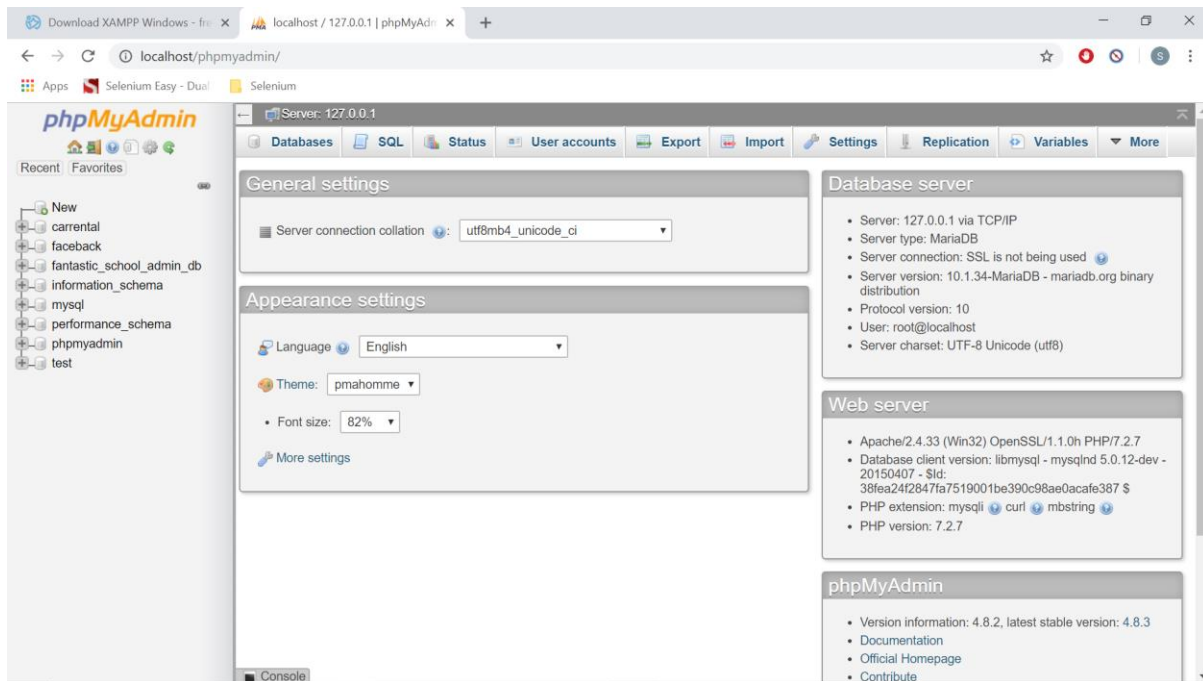
Başarıyla yüklendikten sonra Windows başlat düğmesine gidin ve XAMPP klasörüne gidin; XAMPP Kontrol Panelini göreceksiniz. XAMPP kontrol panelini açın.



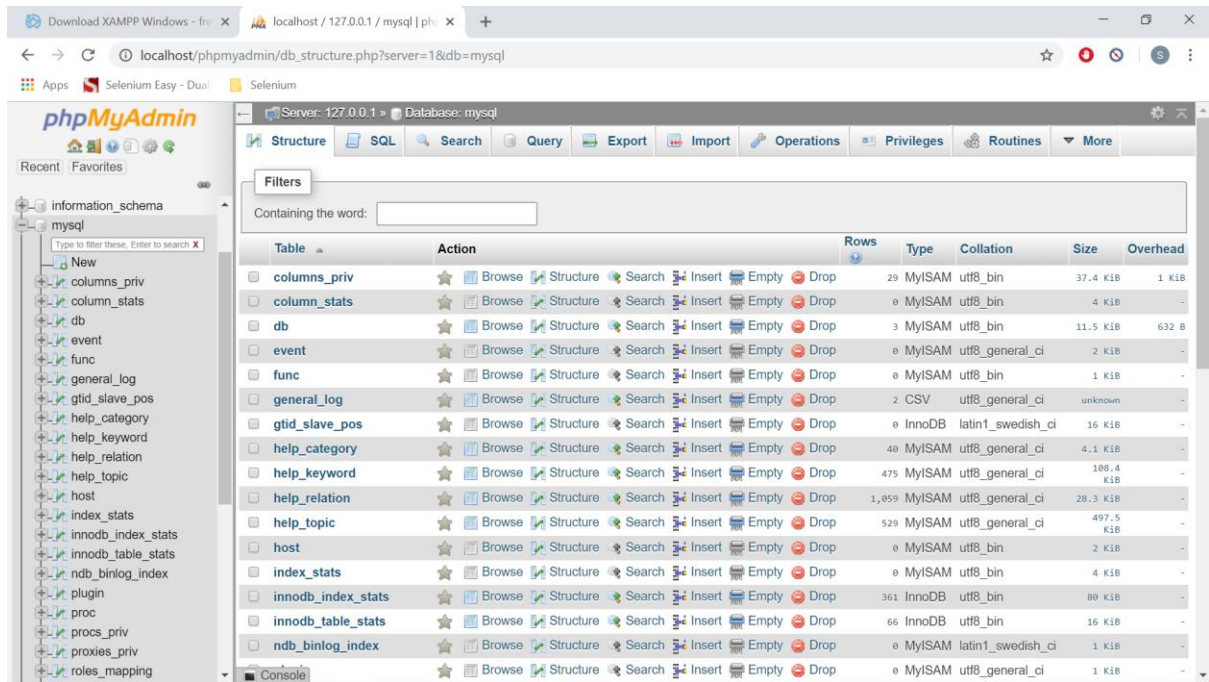
Başlat butonlarına tıklayarak Apache Sunucusunu ve MySQL sunucusunu başlatın. Her ikisinin de hatasız başlatıldığını görmelisiniz.

Artık aşağıdaki URL'yi kullanarak veritabanına erişebilirsiniz.

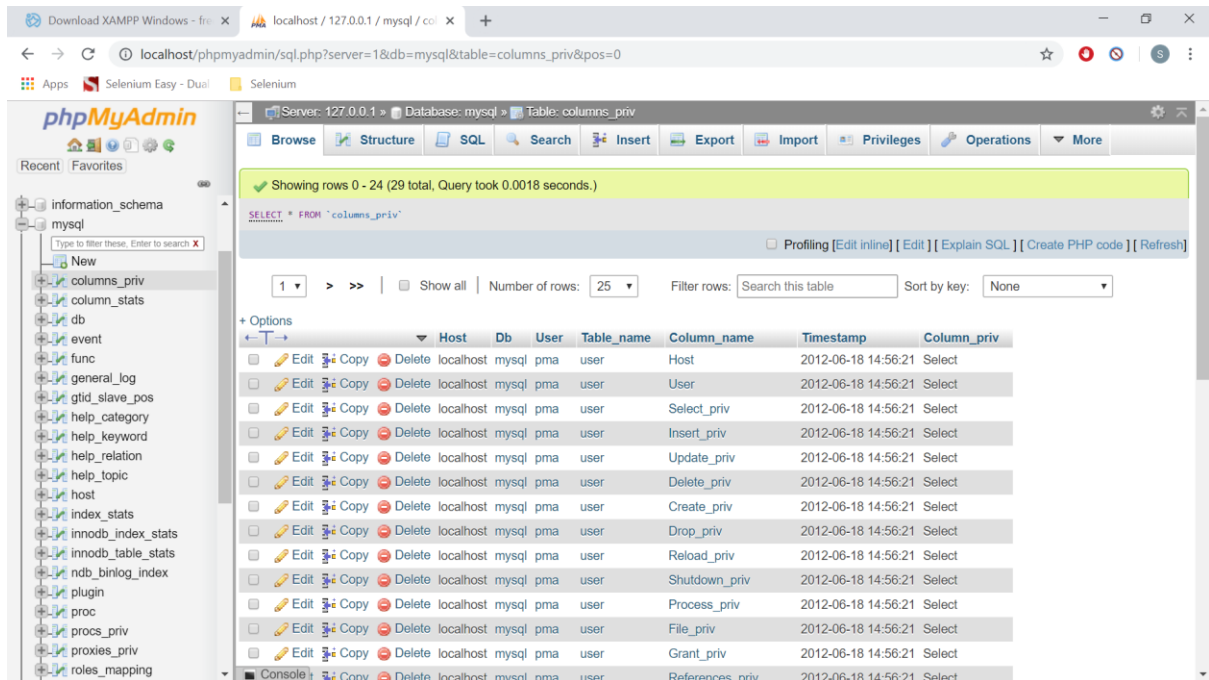
<http://localhost/phpmyadmin>



Sol tarafta farklı veritabanılarını göreceksiniz.



Herhangi bir veritabanına tıkladığınızda, onunla ilişkili farklı tabloları göreceksiniz.



Herhangi bir tabloya tıkladığınızda tabloların verilerini göreceksiniz.

Kullanıcı Adı ve Parola Ayarlama:

XAMPP'yi yükledikten sonra phpMyAdmin için varsayılan kullanıcı adı ve şifre şu şekildedir:

Kullanıcı adı: root

Şifre: (şifre yok).

JDBC kullanarak herhangi bir veritabanına bağlanmak için yukarıdaki ayrıntıları kullanabilirsiniz. Ancak yeni bir kullanıcı adı ve şifre oluşturalım.

Sol üst taraftaki ana sayfa düğmesine veya phpMyAdmin resmine tıklayarak ana sayfaya gidin. Şimdi “Kullanıcı Hesapları”na tıklayın; burada tüm kullanıcı adlarını görebilmeniz gerekir.

“Kullanıcı Hesabı Ekle” bağlantısını tıklayın.

Beğendiğiniz kullanıcı adını ve şifreyi girin.

Server: 127.0.0.1

Databases SQL Status **User accounts** Export Import Settings

User accounts overview User groups

User accounts overview

⚠ A user account allowing any user from localhost to connect is present. This will prevent other users from connecting if connection from any (%) host.

User name	Host name	Password	Global privileges	User group	Grant	Action
<input type="checkbox"/> Any	%	No	USAGE	No		Edit privileges Export
<input type="checkbox"/> Any	localhost	No	USAGE	No		Edit privileges Export
<input type="checkbox"/> pma	localhost	No	USAGE	No		Edit privileges Export
<input type="checkbox"/> root	127.0.0.1	No	ALL PRIVILEGES	Yes		Edit privileges Export
<input type="checkbox"/> root	::1	No	ALL PRIVILEGES	Yes		Edit privileges Export
<input type="checkbox"/> root	localhost	No	ALL PRIVILEGES	Yes		Edit privileges Export


⬆ ☐ Check all With selected: Export

New

Beğendiğiniz kullanıcı adını ve şifreyi girin. (Kullanıcı adı: subbu123 ve şifreyi: subbu123 ekliyorum)

Ana bilgisayarı localhost olarak seçin. (Bu önemlidir, aksi takdirde işe yaramaz)

Add user account

Login Information	
User name:	<div>Use text field: ▼</div> <div>subbu123</div>
Host name:	<div>Local ▼</div> <div>localhost</div> <div></div>
Password:	<div>Use text field: ▼</div> <div>.....</div> <div>Strength: <div></div></div>
Re-type:	<div>.....</div>
Authentication Plugin	<div>Native MySQL authentication ▼</div>
Generate password:	<div>Generate</div> <div></div>

Global Ayrıcalıklar bölümünün altında, “Tümünü İşaretle” onay kutusunu seçin ve kaydedin; tüm ayrıcalıklara sahip yeni bir kullanıcı oluşturulmalıdır.

Global privileges ☒ **Check all**

Note: MySQL privilege names are expressed in English.

<input checked="" type="checkbox"/> Data	<input checked="" type="checkbox"/> Structure	<input checked="" type="checkbox"/> Administration	Resource limits
<input checked="" type="checkbox"/> SELECT	<input checked="" type="checkbox"/> CREATE	<input checked="" type="checkbox"/> GRANT	<i>Note: Setting these opt</i>
<input checked="" type="checkbox"/> INSERT	<input checked="" type="checkbox"/> ALTER	<input checked="" type="checkbox"/> SUPER	MAX QUERIES PER HOU
<input checked="" type="checkbox"/> UPDATE	<input checked="" type="checkbox"/> INDEX	<input checked="" type="checkbox"/> PROCESS	MAX UPDATES PER HOU
<input checked="" type="checkbox"/> DELETE	<input checked="" type="checkbox"/> DROP	<input checked="" type="checkbox"/> RELOAD	MAX CONNECTIONS PER
<input checked="" type="checkbox"/> FILE	<input checked="" type="checkbox"/> CREATE TEMPORARY TABLES	<input checked="" type="checkbox"/> SHUTDOWN	MAX USER_CONNECTION
	<input checked="" type="checkbox"/> SHOW VIEW	<input checked="" type="checkbox"/> SHOW DATABASES	
	<input checked="" type="checkbox"/> CREATE ROUTINE	<input checked="" type="checkbox"/> LOCK TABLES	
	<input checked="" type="checkbox"/> ALTER ROUTINE	<input checked="" type="checkbox"/> REFERENCES	
	<input checked="" type="checkbox"/> EXECUTE	<input checked="" type="checkbox"/> REPLICATION CLIENT	
	<input checked="" type="checkbox"/> CREATE VIEW	<input checked="" type="checkbox"/> REPLICATION SLAVE	
	<input checked="" type="checkbox"/> EVENT	<input checked="" type="checkbox"/> CREATE USER	
	<input checked="" type="checkbox"/> TRIGGER		

Bu adım önemlidir, çünkü bu onay kutusunu işaretlemesiniz yeni kullanıcı tüm ayrıcalıklara sahip olmayacaktır.

26A. Java Veritabanı Bağlantısı - Temel SQL Bilgisi

Veritabanı Testi için Temel SQL Bilgisi:

Veritabanlarıyla çalışmak için minimum SQL bilgisine sahip olmanız gerekir. Bu bölümde bu sql ifadelerini ele alacağım.

İlk önce phpMyAdmin sayfasında sol üst taraftaki ana sayfa düğmesine tıklayın.

localhost/phpmyadmin/sql.php?server=1&db=mysql&table=columns_p

phpMyAdmin

Recent Favorites

information_schema
mysql

Type to filter these, Enter to search X

New

columns_priv
column_stats
db
event
func
general_log
ntid_slave_pos

Server: 127.0.0.1 » Database: mysql » Table:

Browse Structure SQL Search

Showing rows 0 - 24 (29 total, Query took 0.0018 sec)

SELECT * FROM `columns_priv`

1 > >> | Show all | Number of rows

+ Options

	Host	Db	User
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	localhost	mysql	pma
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	localhost	mysql	pma

Veritabanları linkine tıklayın ve veritabanının adını “jdbctesting” olarak girin ve “Oluştur” butonuna tıklayın.

localhost/phpmyadmin/server_databases.php

phpMyAdmin

Recent Favorites

information_schema
mysql

Type to filter these, Enter to search X

New

columns_priv
column_stats
db
event
func
general_log
gtid_slave_pos
help_category
help_keyword
help_relation
help_topic
host

Server: 127.0.0.1

Databases SQL Status User accounts Export Import

Databases

Create database

jdbctesting latin1_swedish_ci Create

Database	Collation	Action
<input type="checkbox"/> carrental	latin1_swedish_ci	<input type="checkbox"/> Check privileges
<input type="checkbox"/> faceback	latin1_swedish_ci	<input type="checkbox"/> Check privileges
<input type="checkbox"/> fantastic_school_admin_db	latin1_swedish_ci	<input type="checkbox"/> Check privileges
<input type="checkbox"/> information_schema	utf8_general_ci	<input type="checkbox"/> Check privileges
<input type="checkbox"/> mysql	latin1_swedish_ci	<input type="checkbox"/> Check privileges
<input type="checkbox"/> performance_schema	utf8_general_ci	<input type="checkbox"/> Check privileges
<input type="checkbox"/> phpmyadmin	utf8_bin	<input type="checkbox"/> Check privileges

Şimdi veritabanı oluşturulmalı ve onu sol bölmede görmelisiniz.

Temel SQL Sorguları:

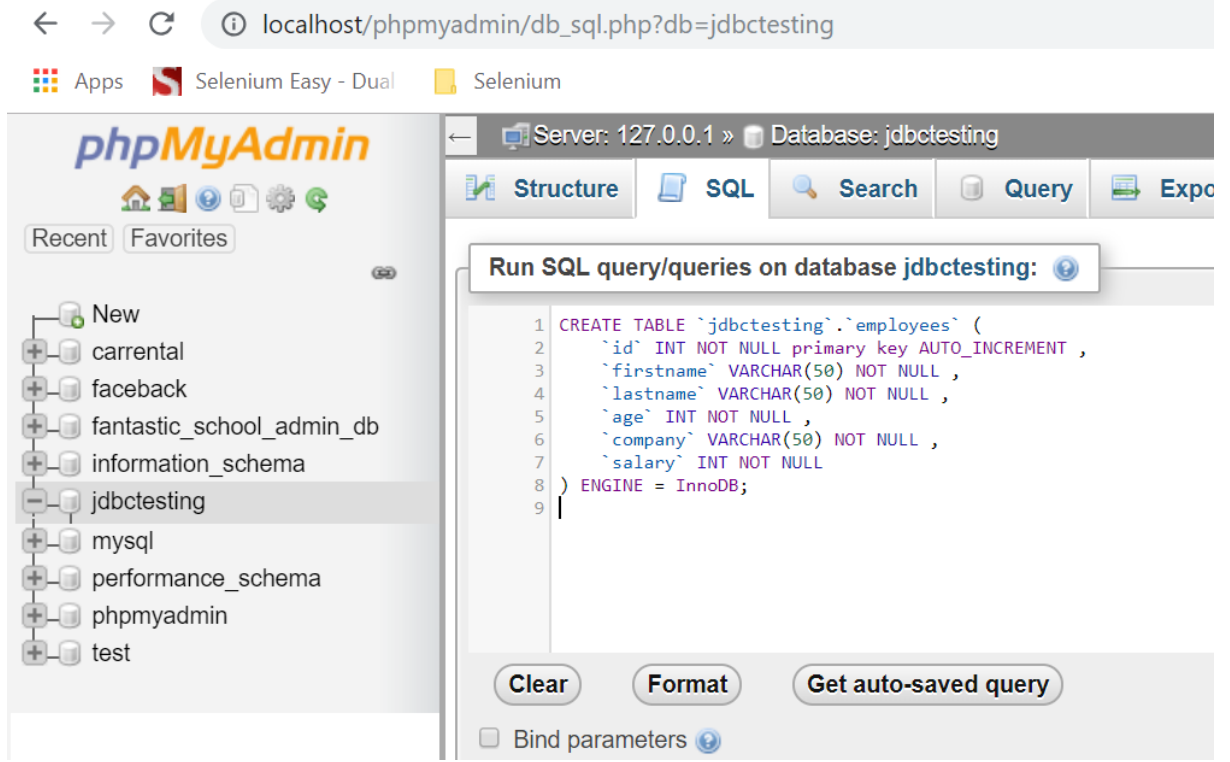
No'lu bir “çalışanlar” tablosu oluşturacağız. 6 sütundan oluşuyor ve bunlar

1. kimlik (otomatik artış)

2. ad (çalışanın adını yakalamak için)
3. soyadı (çalışanın soyadını yakalamak için)
4. yaş (çalışanın yaşını yakalamak için)
5. şirket (çalışanın şirket adını yakalamak için)
6. maaş (çalışanın maaşını yakalamak için)

Tablo ve Veri Oluşturma:

Şimdi SQL bağlantısına tıklayın.



Bu sayfada SQL sorgulama alıştırımları yapabilirsiniz.

Tablo Oluşturma:

```
TABLO OLUŞTUR `jdbctesting`.`çalışanlar` (
`id` INT NOT NULL birincil anahtar AUTO_INCREMENT,
`ad` VARCHAR(50) NULL DEĞİL,
`soyadı` VARCHAR(50) NULL DEĞİL,
`age` INT NULL DEĞİL,
`şirket` VARCHAR(50) NULL DEĞİL,
`maaş` INT NULL DEĞİL
) MOTOR = InnoDB;
```

Not: phpMyAdmin'de, yukarıdaki sorguyu çalıştırırken tek tırnak işaretleri (" ' ") yerine ters tırnak işaretleri (" ` ") kullanın. Aksi takdirde yukarıdaki sorgu çalışmayacaktır.

Tabloya Veri Ekleme:

Sorgu 1:

jdbctesting.employees'a ekleyin (ad, soyad, yaş, şirket, maaş)

DEĞERLER('james', 'demirci', 35, 'IBM', 1250000)

Sorgu 2:

jdbctesting.employees'a ekleyin (ad, soyad, yaş, şirket, maaş)

DEĞERLER('michael', 'demirci', 35, 'HONDA', 1800000)

Tablodan Veri Alma:

Tüm verileri almak istiyorsanız seçme sorgusunu kullanın.

* Çalışanlardan SEÇİN

Belirli satırları almak istiyorsanız, nerede koşuluyla seçme sorgusunu kullanın.

SELECT * FROM çalışanlarından WHERE kimliği = 1

Tablodaki Verileri Güncelleme:

GÜNCELLEME çalışanları SET ad = 'Robert' WHERE kimliği = 1

Where koşulunu kullandığınızdan emin olun, aksi takdirde tüm veriler güncellenecektir.

Tablodan Veri Silme:

Kimliği = 1 olan çalışanlardan SİL

Where koşulunu kullandığınızdan emin olun, aksi takdirde tablonun tamamı silinir.

Kullandığınız en önemli sorgular SELECT ve UPDATE'dir.

26B. Java Veritabanı Bağlantısı - Java kullanarak Veritabanına Bağlanma ve Sorguları Yürütme

Java Veritabanı Bağlantısı:

JDBC, Java veritabanı bağlantısıdır.

Java JDBC, veritabanına bağlanıp sorguları yürütmek için kullanılan bir Java API'sidir. JDBC API, veritabanına bağlanmak için jdbc sürücülerini kullanır.

Herhangi bir Java uygulamasını JDBC kullanarak veritabanına bağlamak için 5 adım vardır . Bu adımlar aşağıdaki gibidir:

- Sürücü sınıfını kaydedin
- Bağlantı oluştur
- Ekstre oluştur
- Sorguları yürütme
- Yakın bağlantı

1. Sürücü Sınıfını Kaydedin

Class sınıfının `forName()` yöntemi, sürücü sınıfını kaydetmek için kullanılır.

MySQL için bu:

```
Class.forName("com.mysql.jdbc.Driver");
```

Oracle için bu:

```
Class.forName("Oracle.jdbc.driver.OracleDriver");
```

2. Bağlantı nesnesi oluşturma

MySQL için:

```
Bağlantı con=DriverManager.getConnection( "jdbc:mysql://localhost:3306/dbname" , "root" , "root" );
```

Oracle için:

```
Bağlantı con=DriverManager.getConnection( "jdbc:oracle:thin:@localhost:1521:xe" , "system" , "password" );
```

3. İfade nesnesini oluşturun

MySQL/Oracle için:

```
Açıklama stmt=con.createStatement();
```

4. Sorguyu çalıştır

```
ResultSet rs=stmt.executeQuery( "emp'ten * seç" );
While(rs.next()){
System.out.println(rs.getInt(1)+" "+rs.getString(2));
}
```

Not: Verileri seçmek için `stmt.executeQuery`'yi kullanıyoruz

Veri eklemek, güncellemek ve silmek için `stmt.executeUpdate`'i kullanınız

5. Bağlantıyı kapat

```
con.close()
```

MySQL'e Bağlanma Örneği:

```
paket FPPaket;
```

```
içe aktarın ;
içe aktarın ;
içe aktarın ;
içe aktarın ;
```

```

halk sınıf JDBCtest {

    halk statik void main(String[] args ) {
        denemek {
            Sınıf. forName ( "com.mysql.cj.jdbc.Driver" );
            //Bağlantı bağlantısı =
            DriverManager.getConnection("jdbc:mysql:// localhost :3306/ carrental
?useSSL=false", "root", "vinayaka123");
            Bağlantı bağlantısı = DriverManager. getConnection (
"jdbc:mysql://localhost:3306/jdbctestng" , "subbu123" , "subbu123" );
            Açıklama stmt = con .createStatement();
            //stmt.executeUpdate("jdbctestng.employees'a ekleyin ( ad ,
soyad , yaş, şirket, maaş)VALUES(' robert ', 'smith', 30, 'HONDA', 1250000)");
            ResultSet rs = stmt .executeQuery( "jdbctestng.employees'dan
* seçin" );
            while ( rs .next())
                Sistem. out .println( rs .getInt(1)+ " " + rs
.getString(2)+ " " + rs .getString(3)+ " " + rs .getInt(4)+ " " + rs .getString(5)
+ " " + rs .getInt(6));
            con .close();
            //stmt.executeUpdate("UPDATE jdbctestng.employees SET name
='Mary', yaş=30, şirket=' Infosys ', maaş=1800000 burada id=2");
            //stmt.executeUpdate("jdbctestng.employees'DAN SİLİN; burada
kimlik=3");
        }
        catch (İstisna e ) {
            Sistem. out .println( e );
        }
    }
}

```

Sonuçlar:

```

1 Maruti 2017-06-18 21:54:34
2 BMW2017-06-18 21:54:50
3Audi 2017-06-18 21:55:03
4 Nissan 2017-06-18 21:55:13
5 Toyota2017-06-18 21:55:24
7 Marutiu 2017-06-19 11:52:13

```

Oracle'a bağlanma örneği:

paket FPPaket;

```

içe aktarın ;
içe aktarın ;
içe aktarın ;
içe aktarın ;

```

```

halk sınıf JDBCtest {

    halk statik void main(String[] args ) {
        denemek {
            Sınıf. forName ( "Oracle.jdbc.driver.OracleDriver" );

```

```

        Bağlantı bağlantısı = DriverManager. getConnection (
"jdbc:oracle:thin:@localhost:1521:carrental" , "root" , "password" );
        Açıklama stmt = con .createStatement();
        ResultSet rs = stmt .executeQuery( "carrental.tblbrands'dan *
seçin" );

        while ( rs .next())
            Sistem. out .println( rs .getInt(1)+ " " + rs
.getString(2)+ " " + rs .getString(3));
        con .close();
    }
    catch (İstisna e ) {
        Sistem. out .println( e );
    }
}
}
}

```

27. Java Yansımaları - Çalışma zamanında sınıfların, yapıcılarının ve yöntemlerin yüklenmesi.

Java Yansımaları – Çalışma Zamanında Sınıfların, Oluşturucuların ve Yöntemlerin Yüklenmesi:

Çalışma zamanında herhangi bir sınıfı, yapıcıcıyı veya yöntemleri yüklemek istiyorsak Java yansıma paketi kullanılır. Bunda sadece selenyum derslerimiz için neyin gerekli olduğunu görüyoruz.

Burada bir “actions” paketi ve onun altında da HomePage adında bir sınıf oluşturduk. İki statik değişken ve bu değişkenleri başlatmak için bir yapıcı oluşturduk. Ayrıca dört yöntem de oluşturduk.

Başka bir paket “drivers” ve bunun altında “ContactActions” sınıfı oluşturduk. Bu sınıfta HomePage sınıfı için herhangi bir nesne yaratmıyoruz ancak bu sınıfa, yapıcısına ve yöntemlerine çalışma zamanında dinamik olarak erişiyoruz.

Eylem paketinin HomePage.java'sı:

paket eylemleri;

```

halk sınıf Ana Sayfası {
    halk statik int küresel ;
    halk statik Dize globalstr ;

    genel Ana Sayfa( int a , Dize dizisi ) {
        küresela = a ;
        genel dizi = dizi ;
    }

    halk int getSumofTwoWithGlobal( int A ) {
        dönüş ( globala + a );
    }

    halk int getSumofThreeWithGlobal( int a , int B ) {
        dönüş ( globala + a + b );
    }

    public String concatTwoWithGlobal(String str1 ) {

```



```

        dönüş ( globalstr + str1 );
    }

    public String concatThreeWithGlobal(String str1 , String str2 ) {
        dönüş ( globalstr + str1 + str2 );
    }
}

```

Çalışma zamanında bir sınıfa erişmek için Class.forName() yöntemini kullanırız. Paket adını ve ardından "." ardından sınıf adı gelir. Bir Class nesnesi döndürür. Class<?> yazarak, herhangi bir türde olabilecek bir Class nesnesi bildirmiş olursunuz (? joker karakterdir). Class türü, bir sınıf hakkında meta bilgileri içeren bir türdür.

Yu, sınıfın sahip olduğu genel yöntemlerin bir dizisini döndüren getDeclaredMethods() işlevini kullanarak sınıfın tüm yöntemlerini elde edebilir. Yani tüm bunları Yöntem[]'e kaydediyoruz.

Artık Arrays.toString() yöntemini kullanarak tüm yöntemleri yazdırabilir veya Method[] dizisinde döngü yapıp tek tek yazdırabilirsiniz.

Herhangi bir yöntemi çağırmak için önce yöntemi yüklememiz gerekir. Bunu bununla yapıyoruz

```
Yöntem mc = cls .getDeclaredMethod( "getSumofTwoWithGlobal" , int . class );
```

Bir yöntemi yüklemek için getDeclaredMethod() yöntemini kullanırız ve yöntemin adını ve kullanılan değişkenlerin türünü aktarmamız gerekir. Yöntemde tam sayı parametreleri kullanırsak int.class'ı, string parametrelerini kullanırsak String.class'ı kullanırız.

Yöntemi çağırmadan önce yapıcıyı yüklememiz gerekiyor. GetConstructor() yöntemini ve yapıcının kullandığı değişken türlerini kullanıyoruz.

```
Yapıcı<?> yapıcı = cls .getConstructor( int . class , String. class );
```

Daha sonra, invoke() yöntemini kullanarak yöntemi çağırıyoruz. Önce parametre olarak struct.newInstance()'ı, ardından yöntemin kullandığı gerçek parametreleri aktarırız. Bu yöntem bir tamsayı döndürdüğü için bunu toplam değişkeninde yakalarız.

```
int toplam = ( int ) mc .invoke( yapıcı .newInstance(1, "Selenium" ), 2);
```

Sürücü paketinin ContactActions.java'sı:

```
paket sürücüler;
```

```
içe aktarın ;
java.lang.reflect.InvocationTargetException'ı içe aktarın ;
içe aktar ;
içe aktarın ;
```

```
halk sınıf ContactActions {
```

```

    halk statik void main(String[] args ) ClassNotFoundException,
    NoSuchMethodException, SecurityException, IllegalAccessException,
    IllegalArgumentException, InvokasyonTargetException, InstantiationException'ı atar
{

```

```

Sınıf<?> cls = Sınıf. forName ( "actions.HomePage" );

//Tüm genel yöntemleri yazdır
Yöntem[] publicMethods = cls .getDeclaredMethods();
Sistem. out .println(Arrays.toString ( publicMethods ) );
için ( int ben =0; i < publicMethods . uzunluk ; ben ++ ) {
    Sistem. out .println( publicMethods [ i ] );
}

//getSumofTwoWithGlobal yöntemini çağır
Yöntem mc = cls .getDeclaredMethod( "getSumofTwoWithGlobal" , int .
class );
Yapıcı<?> yapıcı = cls .getConstructor( int . class , String. class
);
int toplam = ( int ) mc .invoke( yapıcı .newInstance(1, "Selenium"
), 2);

Sistem. out .println( "getSumofTwoWithGlobal için toplam " + toplam
);

//getSumofThreeWithGlobal yöntemini çağır
mc = cls .getDeclaredMethod( "getSumofThreeWithGlobal " , int .class
, int .class );
yapıcı = cls .getConstructor( int . class , String. class );
toplam = ( int ) mc .invoke( yapıcı .newInstance(1, "Selenium" ), 2,
3);

Sistem. out .println( "getSumofThreeWithGlobal için toplam " +
toplam );

// concatTwoWithGlobal yöntemini çağır
mc = cls .getDeclaredMethod( "concatTwoWithGlobal" , String.class )
;
yapıcı = cls .getConstructor( int . class , String. class );
String cnt = (String) mc .invoke( constructor .newInstance(1,
"Selenium" ), " Otomasyon" );

Sistem. out .println( cnt );

//concatThreeWithGlobal yöntemini çağır
mc = cls .getDeclaredMethod( "concatThreeWithGlobal" , String.class
, String.class );
yapıcı = cls .getConstructor( int . class , String. class );
cnt = (String) mc .invoke( constructor .newInstance(1, "Selenium" ),
"Otomasyon" , "Test Etme" );

Sistem. out .println( cnt );
}
}

```

28. Java Tarih İşlevselliği - Güncel Tarihi ve Gelecek Tarihleri Alma

Güncel Tarihi ve Gelecek Tarihleri Alma:

içe aktarın ;

```

içe aktarın ;
içe aktarın ;

halk sınıf JavaDateÖrnekler {

    halk statik void main(String[] args ) {

        Tarih d1 = yeni Tarih();
        Sistem. out .println( "Geçerli Tarih: " + d1 );

        SimpleDateFormat sdf = new SimpleDateFormat( "dd/MM/yyyy" );
        Dize d2 = sdf .format( d1 );
        Sistem. out .println( "Biçimlendirilmiş tarih: " + d2 );

        Takvim c = Takvim. getInstance ();
        c .setTime( new Date()); // Şimdi bugünün tarihini kullanın.
        c .add(Calendar.DATE , 1); // 1 gün ekleniyor
        String fromdate = sdf .format( c .getTime());
        Sistem. out .println( tarihten );

    }

}

```

Java'da numaralandırmalar

Java'daki Enum, sabit bir sabit kümesi içeren bir veri türüdür.

Haftanın günleri, mevsimler, renkler vb. için kullanılabilir.

Bunlar sabit olduğundan, enum sabitlerini büyük harflerle bildirmeliyiz.

- Numaralandırmalar tür güvenliğini artırır
- Numaralandırmalar anahtarda kolayca kullanılabilir
- Numaralandırmalar geçilebilir
- Numaralandırmalarda alanlar, yapıcılar ve yöntemler bulunabilir
- Enum'lar birçok arayüzü uygulayabilir ancak herhangi bir sınıfı genişletemez çünkü dahili olarak Enum sınıfını genişletir

Enum'lar sınıf dışında, sınıf içinde veya ayrı bir program olarak bildirilebilir.

Enum sabitleri "enum adı"."enum sabiti" ile alınabilir.

Bir kurucu tarafından bir değer atanmışsa, değeri "enum name" kullanarak alabiliriz. "enum sabiti".değer. (değer kullandığımız değişkenin adıdır)

Sınıfın Dışındaki Numaralandırma:

Aşağıdaki örnek bir sınıfın dışındaki bir numaralandırma içindir.

```

paket numaralandırmaları;

numaralandırma rengi {
    KIRMIZI YEŞİL MAVİ ;
}

```

```

halk sınıf EnumOutsideClass1 {

    halk statik void main(String[] args ) {
        renk c1 = renk. KIRMIZI ;
        Sistem. out .println( c1 );
    }
}

```

Sonuç:
KİŞ

Sınıf İçi Numaralandırma:

Aşağıdaki örnek bir sınıf içindeki enum içindir.

paket numaralandırmaları;

```

halk sınıf EnumInsideClass2 {

    halk numaralandırma sezonu{
        KİŞ İLKBAHAR YAZ SONBAHAR ;
    }
    halk statik void main(String[] args ) {

        Enum_sabitini yazdırma
        sezon s = sezon. KİŞ ;
        Sistem. out .println( s );
    }
}

```

Sonuç: KİŞ

Enum'da Döngü Yapmak:

Aşağıdaki örnek bir numaralandırmada nasıl döngü yapılacağını gösterir. Values() bir dizi numaralandırma sabiti verir.

paket numaralandırmaları;

```

halk sınıf LoopingThroughEnum3 {

    halk numaralandırma sezonu{
        KİŞ İLKBAHAR YAZ SONBAHAR ;
    }
    halk statik void main(String[] args ) {

        numaralandırma üzerinde yineleme
        for (sezon s1 : sezon.değerler ()) {
            Sistem. out .println( s1 );
        }
    }
}

```

Sonuç:

KİŞ
BAHAR

YAZ
DÜŞMEK

Java derleyicisi bir numaralandırma oluşturduğunda dahili olarak value() yöntemini ekler. Values() yöntemi, numaralandırmanın tüm değerlerini içeren bir dizi döndürür.

Değerleri ve indeksi yazdırma:

Bir numaralandırmanın değerini yazdırmak için valueOf() yöntemini kullanırız. Dizini yazdırmak için önce valueOf() yöntemini, ardından ordinal() yöntemini kullanırız.

paket numaralandırmaları;

```
halk sınıf ValuesAndIndex4 {

    halk numaralandırma sezonu{
        KIŞ İLKBAHAR YAZ SONBAHAR ;
    }
    halk statik void main(String[] args ) {

        //Değerlerin yazdırılması
        Sistem. out .println(sezon.valueOf ( " KIŞ" ));
        Sistem. out .println(sezon.valueOf ( " BAHAR" ));
        Sistem. out .println(sezon.valueOf ( " YAZ" ));

        // İndeks yazdırılıyor
        Sistem. out .println(sezon.valueOf ( " KIŞ" ).ordinal());
        Sistem. out .println(sezon.valueOf ( " BAHAR" ).ordinal());
        Sistem. out .println(sezon.valueOf ( " YAZ" ).ordinal());
    }
}
```

Sonuç:

KIŞ
BAHAR
YAZ
0
1
2

Not: Enum sabitlerinin sonundaki noktalı virgül isteğe bağlıdır.

Yalnızca Ana Yöntemle Enum:

Ayrıca bir Enum'u yalnızca ana yöntemle tanımlayabilir ve çalıştırabilirsiniz.

paket numaralandırmaları;

```
halk enum OnlyEnumWithMain5 {

    KIŞ İLKBAHAR YAZ SONBAHAR ;

    halk statik void main(String[] args ) {
        Sistem. out .println(OnlyEnumWithMain5.WINTER );
    }
}
```

Sonuç:

KIŞ

Enum Sabitlerine Belirli Değerlerin Başlatılması:

Numaralandırmaların başlangıç değerleri 0, 1, 2, 3 vb. olacaktır. Ancak belirli değerleri numaralandırma sabitlerine başlatabiliriz. Ancak bunu yapmak için bir kurucu kullanmalıyız. Eğer başlatmazsak, sabitleri değer olarak kullanacaktır.

Değerleri Enum Sabitlerine Başlatma:

Değerleri enum sabitlerine başlatmak için bir yapıcı tanımlamamız gerekir.

```
paket numaralandirmaları;
```

```
halk sınıf BaşlatmaValuesToEnumConstants6 {

    numaralandırma sezonu{
        KIŞ (5), İLKBAHAR (10), YAZ (20), SONBAHAR (30);

        özel int değer1 ;
        özel sezon ( int değer1 ) {
            Bu . değer1 = değer1 ;
        }
    }

    halk statik void main(String[] args ) {
        for (sezon s : sezon.değerler ()) {
            Sistem. out .println( s + " " + s . değer1 );
        }
    }
}
```

Sonuç:

KIŞ 5
BAHAR 10
YAZ 20
30 GÜZ

Switch Case'deki numaralandırmalar:

Numaralandırmalar switch case ifadelerinde çok faydalıdır.

```
paket numaralandirmaları;
```

```
halk sınıf EnumsInSwitchCase {

    numaralandırma sezonu {
        KIŞ , YAZ , SONBAHAR , İLKBAHAR ;
    }

    halk statik void main(String[] args ) {
        sezon s = sezon. KIŞ ;

        anahtar ( lar ) {
            dava KIŞ :
                Sistem. out .println( "Kış geldi" );
        }
    }
}
```

```

        kirmak ;
dava YAZ :
    Sistem. out .println( "Yaz Geldi" );
    kirmak ;
dava DÜŞMEK :
    Sistem. out .println( "Sonbahar" );
    kirmak ;
dava BAHAR :
    Sistem. out .println( "Bahar Geldi" );
    kirmak ;
    }
}
}

```

Sonuç:

Kış

Ayrı bir dosyadaki numaralandırmalar:

Numaralandırmaları ayrı bir dosyada oluşturabilir ve bunlara farklı bir sınıftan erişebilirsiniz.

EnumInSeperateFile8.java

```

paket numaralandırmaları;

halk enum EnumInSeperateFile8 {
    SPRINT (10), KİŞ (20), YAZ (30), SONBAHAR (40);

    halk int değer ;
    özel EnumInSeperateFile8( int değer ){
        Bu . değer = değer ;
    }
}

```

AccessEnumInSeperateFile8.java

```

paket numaralandırmaları;

halk sınıf AccessEnumInSeperateFile8 {

    halk statik void main(String[] args ) {
        EnumInSeperateFile8 s = EnumInSeperateFile8. KİŞ ;

        Sistem. out .println( s + " " + s . değer );

        for (EnumInSeperateFile8 s1 :EnumInSeperateFile8.values ( )) {
            Sistem. out .println( s1 + " " + s1 . değer );
        }

    }

}

```

Görüşme soruları:

Enum'daki value() yönteminin amacı nedir?

Java derleyicisi bir numaralandırma oluşturduğunda dahili olarak value() yöntemini ekler. Values() yöntemi, numaralandırmanın tüm değerlerini içeren bir dizi döndürür.

Enum'daki valueOf() yönteminin amacı nedir?

Java derleyicisi bir numaralandırma oluşturduğunda dahili olarak valueOf() yöntemini ekler. ValueOf() yöntemi, verilen sabit numaralandırmanın değerini döndürür.

Enum'da ordinal() yönteminin amacı nedir?

Java derleyicisi bir numaralandırma oluşturduğunda dahili olarak ordinal() yöntemini ekler. ordinal() yöntemi, enum değerinin dizinini döndürür.

Enums'un başlangıç değerleri nelerdir?

0, 1, 2, 3 vb.

Enum yapıcısının erişim türü nedir?

Özeldir. Özel kurucuyu bildirmezseniz dahili olarak özel bir kurucu oluşturulur.

New anahtar kelimesini kullanarak bir Enum örneği oluşturabilir miyiz?

Hayır. Çünkü Enum'lar yalnızca özel kurucular içerir.

Enums'ta soyut yöntemlere sahip olabilir miyiz?

Evet, numaralandırmalarda soyut yöntemlere sahip olabiliriz ve bu yöntemlerin uygulanmasını sağlayabiliriz.