Java İnterview Soruları (1-80 Arası)

S1: Java platformu neden bağımsızdır?

C1: Herhangi bir sistemde çalışabilen bayt kodlarından dolayı.

Aciklama: Java'nın platform bağımsız olabilmesinin nedeni **doğrudan makine diline derlenmemesidir**. Fakat bu Java'nın derlenmediği anlamına gelmez. Java kaynak kodları sadece Java Sanal Makinesinin-JSM (Java Virtual Machine-JVM) yorumlayabileceği bitkodlarına (bytecode) derlenirler. Bitkodları işlemci seçmezler.

Url: https://www.youtube.com/watch?v=CiOYIWVBTds

S2: Java neden %100 nesne yönelimli değil?

C2: Çünkü nesne olmayan 8 primitive data türünü kullanır.

Aciklama: Java, nesne olmayan sekiz ilkel veri türü (boolean, byte, char, int, float, double, long ve short) kullandığından % 100 nesne yönelimli değildir.

Url: https://web.cs.hacettepe.edu.tr/~bbm102/misc/java_notes_by_oa.pdf

Url2: https://bilisimprofesyonelleri.com/30-soruda-java-gelistirici-mulakati/

S3: public static void main (String [] args) Java'da yer alan bu yapıyı açıklayınız.

C3: Bu, herhangi bir Java programı için giriş noktasıdır

public: bu yönteme kimin erişebileceğini belirtmek için kullanılan bir erişim değiştiricidir ve bu yöntemin herhangi bir sınıf tarafından erişilebilir olacağı anlamına gelir.

static: Java'da sınıf tabanlı olduğunu tanımlayan bir anahtar kelimedir. main (), bir sınıfın oluşumunu oluşturmadan erişilebilmesi için Java'da static yapılır.

void: methodun return type dır.

main(): JVM tarafından aranan methodun adıdır.

JVM: Java sanal makinesi

String args ise main methoda iletilen parametredir.

Aciklama 1: Main metodu programin ana metodu olarak gecer. Cogu dilde bu boyledir. Main metodlarinin farkli sekillerde yazildigi farkli diller mevcuttur. Ana metodlar olmadan programin giris bolgeleri belirlenemez.

Hikaye ornegi verebilirim.

giris'i olmadan gelisme ve sonuc'u olan hikayeler olur mu? :)

Main metodunu her dosya icine eklemeyeceksiniz. Genelde main metodu ilk dosyanizda olur. Siz yeni siniflarinizi o metod altinda turetirsiniz. Ornegin

```
public static void Main(String[] args)
{
   Sinif snf = new Sinif();
   snf.uyeIslev(Burada islemler);
}
```

Kolay gelsin

****

Aciklama 2: JVM kodları çalıştırırken yazılmış onlarca class arasından nereden başlayacağını bilmesi gerekmektedir ve JVM standartlaştırılmış **public static void main(String[ ] args)** metodunu arar ve oradan çalışmaya başlar. İşte bu yüzden JVM e "sen ilk olarak bu metodu çalıştır" demek için bunu yapmalıyız.

S4: Wrapper Class'lar nelerdir?

C4: Wrapper Class'lar Java primitive datalarını o sınıfın başvuru türlerine veya o Class'ın nesnelerine dönüştürürler.

Primitive Data Type	Wrapper Class
char	Character
byte	Byte
short	Short
long	Integer
float	Float
double	Double
boolean	Boolean

Wrappler hakkında bilmemiz gerekenler şunlardır:

- Javada 8 tane primitive type vardır. Bunlara karşılık gelen sınıflara wrapper adı verilir.
- Wrapper'lar parametreleri String kabul edebilirler.
- Boolean wrapper, parametreye string olarak true verildiği sürece true döner, diğer durumlarda false döner.
- Character wrapper, ascii veya karakter alabilir.
- Url: https://www.youtube.com/watch?v=L91h88SMOBw

S5: Java da Constructor nedir?

C5: Bir nesneyi başlatmak için kullanılan bir kod bloğunu ifade eder. Sınıfın adıyla aynı ada sahip olmalıdır.

Aciklama: Oluşturulan sınıf yapılarının nesne olarak tanımlanması durumunda proje dosyasının alt yapısını hazırlayan, kurucu rol üstlenerek çeşitli ilk işlemleri gerçekleştiren, kullanılan sınıf yapısı ile aynı isme sahip olan, geriye değer döndürmeyen özel metot yapılarıdır.

- Url: https://www.youtube.com/watch?v=6UfXKx2Q59Q
- Url2: https://www.youtube.com/watch?v=HKGsaeaXueM

S6: Kaç çeşit Constructor vardır?

C6: 2 çeşit Constructor vardır.

- 1) Default Constructor: herhangi bir girdi ya da değer almaz.
- 2) Parmeterized Constructor: durum değişkenlerini sağlanan değerlerle
- Başlatabilir

Aciklama: Varsayılan Yapıcı Nedir?

- Yapıcı, bir nesne oluşturulduğunda çağrılır. Ayrıca o nesne için bellek ayırır. Ayrıca, sınıftaki örnek değişkenlere başlangıç değerlerinin verilmesine yardımcı olur. Programcı bir kurucu tanımlamazsa, program varsayılan kurucuyu otomatik olarak çağırır. Tüm üye değişkenleri sıfır veya boş olarak başlatır.

Student.java

```java
package com.exl;

public class Student {

	private int id;
	private String name;

	public static void main(String[] args) {

		Student s = new Student();
		System.out.println("Student id: " +s.id);
		System.out.println("Student name : "+s.name);
	}
}
```

Problems | Javadoc | Declaration | Console

```
<terminated> Student (1) [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (Nov 6, 2018,
Student id: 0
Student name : null
```

Parametreli Yapıcı Nedir?

- Parametrelendirilmiş kurucu, parametreleri kabul eden bir kurucudur. Bir veya daha fazla parametre olabilir. Parametreli bir kurucu olduğunda, program varsayılan kurucuyu çağırmaz. Ayrıca programcı, yapıcı adından sonra parantez içindeki parametreleri bildirebilir.
- Yanda Calculation adlı bir sınıf var. num1 ve num2 adında iki örnek değişkeni vardır. 7. satırda parametreli bir kurucu var. İki argüman x ve y alır ve bu değerleri num1 ve num2 örnek değişkenlerine atar.
- Ayrıca sum diye bir yöntem var. Bu iki sayının toplamını döndürür. Ana yöntemde bir Hesaplama nesnesi vardır. Toplam yöntemi obj1 kullanılarak çağrılır. Son olarak, sonuçlar konsolda yazdırılır.

Calculation.java

```java
package com.exl;
public class Calculation {

	private int num1;
	private int num2;

	public Calculation(int x, int y){
		num1 = x;
		num2 = y;
	}

	public int sum(){
		return (num1+num2);
	}

	public static void main(String[] args) {

		Calculation obj1 = new Calculation(2,3);
		int result= obj1.sum();
		System.out.println("Answer : " + result);
	}
}
```

Problems | Javadoc | Declaration | Console

```
<terminated> Calculation [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (N
Answer : 5
```

S7: Bir Class'i nasıl singleton yapabiliriz?

C7: O Class'ta ki Constructor i private yaparak.

- Aciklama: **Singleton** tasarım kalıbının genel formu aşağıdaki gibidir. Bu tasarım kalının kullanımı oldukça basittir. **Singleton** uygulanacak sınıfın constructor medodu private **yapılır** ardından sınıfın içinde kendi türünden static **bir** sınıf tanımlanır. Tanımlanan bu sınıfa erişebilmek için **bir** metod sınıfa eklenir.
- Url: https://www.youtube.com/watch?v=5DOEhu7LwKg

S8: Java da ArrayList ve Vector arasındaki fark nedir?

C8: ArrayList hızlı olması için senkronize edilmemiştir ancak Vector, iş parçacığı açısından güvenli olduğundan yavaş olması için senkronize edilmiştir.

- Url:https://www.youtube.com/watch?v=WSrZAv-efB4

Karşılaştırma için temel	ArrayList	Vektör
Temel	ArrayList sınıfı Senkronize edilmedi.	Vektör sınıfı senkronize edildi.
Eski sınıf	ArrayList standart bir Koleksiyon sınıfıdır.	Vector, koleksiyon sınıfını desteklemek için yeniden tasarlanmış eski bir sınıftır.
Sınıf Beyanı	sınıf ArrayList	sınıf vektör
Yeniden tahsis	Belirtilmediğinde, bir ArrayList boyutunun yarısı kadar artırılır.	Belirtilmediğinde, bir vektör boyutunu iki katına çıkarmak için artırılır.
performans	ArrayList senkronize edilmediğinden, Vector öğesinden daha hızlı çalışır.	Vector senkronize edildiğinde ArrayList'ten daha yavaş çalışır.
Sayım / Iterator	ArrayList, ArrayList'te depolanan nesneleri dolaşmak için Iterator arabirimini kullanır.	Vector, Vektörler'de depolanan nesneleri geçiş yapmak için Numaralandırma ve Yineleyici arabirimini kullanır.

S9: Java da equals () ve = = işareti arasındaki fark nedir?

C9: equals () operatörü primitive dataları ve Objectleri karşılaştırmak için kullanılır.

- == ise iki tane Object i karşılaştırmak için kullanılır.
- **Aciklama:** (==) ve Equals() methodları'nın ikisi de farklı 2 değeri karşılaştırmak için kullanılır. (==) operator'ü 2 nesneyi karşılaştırırken, Equals() methodu nesnenin içerdiği string'i karşılaştırır. Yani kısaca (==) operatörü 2 nesnenin referans değerlerini karşılaştırırken Equals() methodu sadece içeriği karşılaştırır.
- Url: https://www.youtube.com/watch?v=jT06ibYdEXo

Q10: Java'da Heap ve Stack Memory arasındaki farklar nelerdir?

- C10: Heap hafızası bir Application in tüm parçaları tarafından kullanılır. Objectler Heap içinde depolanır ve herkesçe genel kullanıma müsaittir. Stack hafıza ise sadece bir yürütme dizisi tarafından kullanılır ve diğerleri tarafından asla kullanılamaz.
- **Aciklama:** Stack ve Heap Arasındaki Farklar
- **Stack bellekten statik olarak yer tahsisi için kullanılırken, Heap dinamik olarak yer tahsisi etmeyi sağlar**. Her ikisi de Ram bölgesinde bulunur. Stack'te yer alan veriler direk bellek içine yerleştirilir dolayısıyla erişimi çok hızlıdır.
- Heap ve Stack arasında ki en önemli farklardan birisi heap de veriler karışık bir şekilde saklanır stack de ise artan ya da azalan adres mantığında (big and little endian) çalışır. Buna bağlı olarak heap de ki bir veriye erişmek stack de ki bir veriye erişmeye göre daha maliyetli bir işlemdir. Başka bir farkı ise stack de ki veri hemen silinirken heap de ki veri Garbage Collector (Çöp Toplayıcı) algoritmasına bağlıdır.
- Url: https://www.youtube.com/watch?v=450maTzSIvA
- Url2: https://medium.com/t%C3%BCrkiye/stack-ve-heap-kavram%C4%B1-59adcb29d454

S11: Java da package nedir?

İlgili Class'ların toplandığı yerdir.

Paketin Yapısı:

package paket_adı ;

import paket_listesi;

{

//paket gövdesi

}

Paket gövdesine konacak sınıflar ve arayüzler, bilindik şekilde (şimdiye dek yaptığımız gibi) tanımlanırlar. Paket içindeki bir sınıfın kendi alt sınıfları, constructor'ları, anlık ve static değişkenleri, metotları, metotlar içinde yerel değişkenleri var olabilir.

S12:Package'ların avantajları nelerdir

1. İsim çatışmasını önler.

2. Kodlara kolay ulaşım sağlar.

3. Gizli Class'ları barındırır ki Class dışından görülemesinler.

S-13. Neden pointer lar Java da kullanılmazlar?

Java'da göstergeçler var, ama C dilinde olduğu şekilde işlemiyorlar. Bu yazımda Java'da göstergeçlerin neden olmadığını aktarmaya çalışacağım.

C dilinde aşağıdaki şekilde bir göstergeç tanımlaması yapılabilmektedir:

```
1  int  i, *ptr;
2  ptr = &i;
3  *ptr=10;
```

Bu örnekte *i* isminde ve int veri tipinde bir değişken ve yine int veri tipine sahip ve *ptr* isminde bir göstergeç tanımladık. C dilinde göstergeçler bir hafıza alanına işaret ederler. Yukarıdaki örnekte *ptr* in sahip olduğu değer *i* değişkeninin hafızadaki adresidir. *ptr* üzerinde yapılan her işlem *i* değişkenini etkileyebilir, çünkü *ptr* dolaylı olarak *i* değişkeninin sahip olduğu değere işaret etmektedir. Göstergeç aritmetiği sayesinde hafıza alanlarını doğrudan adreslemek ve o hafıza alanlarında yer alan değerler üzerinde işlem yapmak mümkündür.

Java'da göstergeç tanımlamaları ve hafıza alanları üzerinden doğrudan işlem yapmak mümkün değildir, çünkü:

- Java sanal makinesi (Java Virtual Machine) hafıza alanını kendisi yönetir.
- Java'da array ve list nesnelerinin uzunlukları JVM tarafından kontrol edilir. Olmayan array elemanları üzerinde işlem yapmaya çalışmak ArrayIndexOutOfBoundsException hatasını oluşturur.
- Java dilinde array dilin bir parçasıdır ve string işlemleri için String sınıfı kullanılır. C dilinde göstergeçler aracılığı ile bu yapılar oluşturulur ve dilin bir parçası değildirler.
- Java nesne referanslarını JVM bünyesinde yer alan garbage collector aracılığı ile yönetir.
- Java dilinde göstergeçler sadece nesnelere işaret eden değişken isimleri olarak kullanılırlar, ama:
- Bu değişkenler üzerinde göstergeç aritmetiği yapılamaz
- Sadece nesnelere işaret ederler, adres alanlarına değil

Java'da bu nesne göstergeçlerine referans ismi verilmektedir.

Aşağıdaki Java kodunu koşturduğunuzda NullPointerExceptin oluşmaktadır:

```
1  Nesne nesne = null;
2  nesne.birMethod();
```

Peki neden oluşan hatanın ismi NullPointerException seklindedir de NullReferenceException değildir? Bu sorunun cevabını bende bilmiyorum. Java'da NullReferenceException olmalıydı. NullPointerException sınıfının kaynak koduna baktığımızda, sınıfın kim tarafından geliştirildiği görememekle birlikte, JDK1.0 dan beri Java'nın bir parçası olduğunu görmekteyiz. Bu sanırım Java dilini oluşturanların C/C++ kökenli olmalarından kaynaklanıyor. Ama C# dilinde NullReferenceException isminde ve aynı vazifeyi gören bir sınıf mevcut. C# dil geliştiricileri aynı hatayı yapmamış.

```
   /**
1   * @author  unascribed
2   * @version %I%, %G%
3   * @since   JDK1.0
4   */
5  public
6  class NullPointerException extends RuntimeException {
7  }
8
```

Şüphesiz Java dilinde göstergeçlerin ve göstergeç aritmetiğinin olmaması Java'da geliştirilen kodun daha kolay anlaşılır olmasını sağlamaktadır. C kodu çok kısa zamanda göstergeçlerin yerli, yersiz kullanılmalarından dolayı kaotik bir yapıya bürünebilir. Ama bunun karşılığında göstergeçler yardımı ile donanıma çok yakın kalınarak, uygulama geliştirmek mümkündür. Java'da bu ne yazık ki mümkün değil. Sizinle donanım arasında her zaman JVM olacak.

S14:Java da JIT derleyici (compiler) nedir?

JIT (Just-in-time compilation; dinamik çeviri olarak da bilinir;) **bilgisayar kodunu çalıştırmanın bir yoludur**. Yürütülmeden önce bir program yürütülürken çalışma zamanında derleyici içerir. Genellikle bu, kaynak kodu ve daha sonradan makine diline bytecode kod çevirisini içerir ve bu kod doğrudan doğruya çalıştırılır.

S15:Java da access modifier lar nelerdir?

1. Default
2. Private
3. Public
4. Protected.

S16:Java'da ki access modifier lari açıklayınız?

Bir java metodu, değişkeni ya da sınıf oluşturulurken bu öğelere kimlerin erişebileceğini belirtme olanağımız vardır. Bu eylemi gerçekleştirecek olan anahtar kelimelere *Erişim Belirleyiciler(Access Modifiers)* adını veririz.

→ 4 seviyede erişim belirleyici mevcuttur. En genişten en dara(kısıtlı) doğru sıralama şu şekildedir;

1. **public :** Bulunduğu paketin ve sınıfın dışındaki başka paketler ve sınıflardan erişilebilir.
2. **protected :** Farklı paketlerden erişim sağlanamaz.
3. **default :** Programda herhangi bir belirleyici mevcut olmadığı takdirde metodun ya da sınıfın erişim belirleyicisi default olur. Bu takıyı alan metotlar alt sınıftan ve bulunduğu paketten erişilebilir.
4. **private :** Yalnızca bulunduğu sınıftan erişilebilir.

Public

Sınıf tanımlarken bir örnek değişkenini(alanını) public erişim belirleyici ile deklare ettiğimizde; bu sınıftan oluşturduğumuz bir nesnenin bu değişkenine başka bir sınıf içerisinden nesneAdi.degiskenAdi şeklinde erişebiliriz.

Protected

Bir sınıf içerisindeki değişkenlere ve metotlara dış alt sınıflardan erişebilmesini, ama başka paketlerin içindeki deyimlerin erişimini engellemek isteyebiliriz. Bunun için protected anahtar kelimesini kullanmamız yeterli olacaktır.

protected, default ya da public erişim belirleyici ile aynıymış gibi görülür. Programcılıkta kalıtım olgusunda oldukça fazla kullanılır.

Private

Sınıf tanımlarken bir değişkeni private erişim belirleyici ile deklare ettiğimizde; bu sınıftan oluşturduğumuz bir nesnenin bu değişkenine başka bir sınıf içerisinden nesneAdi.degiskenAdi şeklinde artık *erişemeyiz*.

S17:Java da class i tarif ediniz?

Sınıf (class) soyut bir veri tipidir. Nesne (object) onun somutlaşan bir cismidir.

Java'da sınıf (class) kavramını doğada cins isimlere benzetebiliriz. Bir cins kendi başına belirli bir nesne değildir; ancak belirli türden nesnelerin ortak özelliklerini belirten soyut bir kavramdır. Örneğin, ağaç bir cins isimdir. Ama bahçedeki bir elma-ağacı ya da sokaktaki bir çınar-ağacı belirli varlıklardır. Onlar, ağaç sınıfının birer nesnesidir (üyesidir). Cinsler alt-cinslere ayrılabilir. Alt-cinsler, üst-cinslerin özelikleri yanında kendilerine has başka özelikler de taşırlar. Örneğin, memeli-hayvanlar geniş bir sınıftır. Filler, kaplanlar, şempanzeler, balinalar, insanlar vb. memeliler sınıfının (üst-sınıf) birer alt-sınıfıdır. Alt-sınıftakiler, üst-sınıfın özeliklerini taşımakla birlikte, birbirlerinden kesinlikle farklıdırlar. Her cinsin ve her alt-cinsin kendine özgü özelikleri (nitelikler ve davranışlar) vardır. Bu özelikler, onları diğer cinslerden ayırır.

S18:Java da obje (Object) nedir ve nasıl oluşturulur?

C18: Obje, örnek değişkenlere sahip sınıfın bir örneğidir. Java'da **new** anahtar sözcüğü kullanılarak bir Obje oluşturulur.

S19:OOP nesne yönelimli programlama nedir?

C19: Bu bir programlama modeli veya yaklaşımı, büyük ve karmaşık kodlu programlar için idealdir ve aktif olarak güncellenmesi ve sürdürülmesi gerekir. Programlar, mantık ve işlevler yerine objeler etrafında düzenlenmiştir.

Nesne Yönelimli Programlama
Nesne yönelikli programlamaya örnek verecek olursak gerçek hayatta gördüğümüz araba, radyo, bina… gibi nesnelerin bilgisayar ortamına aktarılmasına denir.
Örneğin lambanın açık-kapalı olması durumu veya radyonun rengi, markası, üretim yılı… gibi özelliklerinin bilgisayar ortamında gösterilmesidir.

Nesne Yönelikli Programlamanın sağladığı kolaylıklar

- Gerçek dünyadaki nesnelerin tasarımları sınıf içinde yapılır.
- Sınıftan nesne üretilip değişiklik yapılmak istendiğinde tüm programda değişiklik yapmak gerekmez, sadece oluşturulan nesnenin sınıf içinde değişiklik yapılması yeterlidir.
- Oluşturulan nesneler birbirinden bağımsız olduğu için bilgi gizleme olanağı artar.
  Örneğin A nesnesi B nesnesinin özelliklerini kullanamaz ve erişemez.
- Nesne oluşturma, bir sınıf içerisinde gerçekleştirilir ve bu kodlar başka projelerde kullanılabilir.
  Örneğin bir A nesnesi oluşturduysak bunu birçok yerde kullanabiliriz.
- Oluşturulan sınıflar yardımıyla daha az kod ile daha fazla iş yapılıp kod tekarı önlenir.
  Örneğin insan sınıfında isim, soyisim, yaş… gibi özellikleri bir defa oluşturup istediğimiz kadar kullanabiliriz.
- Kod tekrarı önlendiği için geliştirme sürecinin verimliliği artar.

S20:Java da OOP nin ana konsepti nelerdir?

Nesne Yönelimli Programlamada 4 temel özellik vardır. Bu özelliklerden birini sağlamayan programlama dili nesne yönelimli programlama dili olarak sayılmaz.

1.) Soyutlama (Abstraction)
2.) Kapsülleme (Encapsulation)
3.) Miras Alma (Inheritance)
4.) Çok biçimlilik (Polymorphism)

Soyutlama: Bir sınıfta davranış ve özelliklerin tanımlanmasına soyutlama diyoruz.
Örneğin araba sınıfında rengi, modeli, tekerlek sayısı, motor gücü, özellikleridir. Hızlanması, fren yapması, durması davranışlarıdır ve metotlar ile tanımlanır.

Kapsülleme: Davranış ve özellikler sınıfta soyutlanarak kapsüllenir. Kapsülleme ile hangi özellik ve davranışın dışarıya sunulup sunulmayacağını belirleriz.
Örneğin İnsan sınıfında yemek alışkanlığı bizi iliglendirmiyorsa bunu kapalı (private) yapıp gizleriz. Ancak isim soyisim gibi bilgiler bizi ilgilendirdiği için bunlar açık bırakılır. Bu olaya bilgi saklama yani kapsülleme denilmektedir. Bilgi saklama erişim belirteçleri (public, private, protected) ile gerçekleştirilir.

public : herkesin kullanabileceği özellik ve davranışlardır
private : sadece kendi sınıfında kullanılabilen özellik ve davranışlardır
protected : sınıf içinde miras alınan alt sınıflar tarafından kullanılır.

Kalıtım: Sınıflar birbirinden türeyebilir. Alt sınıf üst sınıfın özelliklerini alabilir.
Örneğin araba ve bisiklet sınıflarında ortak özellik olarak tekerlek sayısı, hızı… gibi özelliklerini tekrar yazmak yerine bu özellikleri içeren bir sınıf oluşturup miras alabiliriz. Bir sınıftan birden fazla miraz alınıyorsa buna *çoklu kalıtım* denir.

Çok Biçimlilik: Alt sınıflar üst sınıfın gösterdiği davranışları göstermek zorunda değildir. Alt sınıfların farklı davranışları göstermesine Çok biçimlilik denilmektedir.
Örneğin Gemi ve araba sınıflarını ele aldığımızda bunların hareket tipleri bulunmaktadır. Gemi su üzerinden giderken araba karada hareket etmektedir. Kısaca farklı nesnelerin *(araba ve gemi gibi)* aynı olaya (hareket tipine) farklı şekilde cevap vermesidir.

S21: Bunları (OOPs nin ana konseptleri) açıklayınız?

C21:

Inheritance (kalıtım-miras) : Bir sınıfın başka bir sınıfın özelliklerini elde ettiği miras aldığı bir süreçtir. Yeni oluşturduğumuz bir class'a istedeiğimiz özellikleri (variable veya method) barındıran eski (var olan) class'lardan herhangi birini parent tanımlayarak, parent class'daki erişilebilir olan tüm özellikleri child class'ın kullanabileceği bir konseptir.

Encapsulation-kapsülleme: Verileri sarmalama mekanizmasıdır. Java'da class üyelerine erişimi sınırlandırırz. Access modifier'da aynı işlemi yapar ama aralarında bir fark var. Access modifier'da write ve read işlemlerini ayıramıyoruz. Encapsulation sayesinde bu yapılabilir. Bir class üyesini istersek sadece Write-only (setter) yani değer ataması , yazılabilir , yapılır ama değeri okunamaz. Veya sadece Read-only (getter) yani sadece değeri okunabilir.

Abstraction-soyutlama: Kullanıcıdan var olan detayları gizleme metodolojisidir. Ortak özellikleri olan nesneleri tek bir çatı altında toplamak için kullanılır. Abstract class'lardan obje üretilmez. Tüm child class'larda olmasını istediğimiz dinamik özellikleri (methods) abstract class'da abstract method olarak oluştururuz .

Polymorphism: Bir değişkenin birden çok form alma yeteneğidir. Kalıtım, başka bir sınıftan variable'ları ve method'arı miras almamızı sağlar. Polimorfizm, farklı görevleri gerçekleştirmek için bu yöntemleri kullanır. Bu, tek bir eylemi (methodu) farklı şekillerde gerçekleştirmemizi sağlar.

S22: Local variables ile instance variables arasındaki fark nedir?

C22:

Local variables: bir method, constructor , block, if, for vb. içinde kullanılır ve yalnızca yerel kapsamı vardır. method, constructor veya block scopeları dışında kullanılamazlar.

İnstance variables: Bir instance variable obje değişkenidir. Bundan dolayı instance variable objenin yaşam süresince varolur. Bir class içinde ancak method dışında initialization edilirler. Class içerisindeki non-static olan bütün methodlar tarafından kendisine erişilebilir.

S23: Java da Constructors ve Methods arasındaki farklar nelerdir?

C23:

- Constructors: Bir objenin durumunu başlatmak için kullanılır.
- Methods: Bir objenin davranışını temsil etmek için kullanılır.
- Constructors: Herhangi bir dönüş türü (return type) yoktur.
- Methods: Bir (return type) dönüş türüne sahip olmalıdır.
- Constructors: (Örtülü) İmplicitly olarak çağrılır
- Methods: (Açıkça) Explicitly çağrılması gerekiyor
- Constructors: Class adıyla aynı isimde olmak zorundadır.
- Methods: Olabilir veya olmayabilir de.

S24: Java'da final keyword unu açıklayınız?

C24:

Class, variable ve methodlar için kullanılan ve bunları değiştirilemez hale getiren non-access modifier (erişim olmayan değiştirici) olarak kullanılır.

S25: Final keyword un kullanımını açıklayınız?

C25:

- Final Variable Degeri degistirilmeyecek (constant) variable'lar Icin kullanilir, mutlaka deger atanmalidir, isimleri buyuk harfle yazilir(optional)
- Final Methods Override edilemeyecek method
- Final Classes Inherit edilemeyecek class

S26: Break ve Continue yapıları arasındaki farklar nelerdir?

C26:

1. Break anahtar sözcüğü for döngüsü, while döngüsü veya bir switch bloğunu kırmak için kullanılır.

Continue sadece Loop'larda kullanılır. Continue, geçerli iteration'ı (yinelemeyi) bir for döngüsünde (veya bir while döngüsünde) atlamak/ sonlandırmak için kullanılır ve bir sonraki iteration'a (yinelemeye) devam eder.

2. Break sistem çalışırken bulunduğu işlemi sonlandırır.

Continue işlemi sonlandırmaz sadece sıradaki diğer seçeneğe atlar.

S27: Java'da sonsuz döngü (Loop) nedir?

C27:

Java'da bir işlevsellik karşılanmadığında, bir talimat dizisi tarafından sonsuz döngüye girilir.

Çeşitli sebeplerle sonsuza kadar döngüye giren ve sonlanamayan bilgisayar programı komutu parçalarına verilen ad. Bu sebepler döngünün; bir sonlandırıcı koşulun bulunmaması, bulunsa da hiçbir zaman sağlanamayacak olması ya da bu koşulun döngünün her seferinde yeniden başlamasına neden olması olabilir.

S28: Java'da this() ve super() anahtar kelimeler arasındaki fark nedir?

C28:

this() aynı class in varsayılan constructor çağırmak için kullanılır, ancak super() parent class' tan constructor çağırmak için kullanılır.

Not1: this () ve super () keywords (anahtar sözcükleri) bir bloğun ilk satırında yer almalıdırlar.

Not2: super( ) ve this( ) bir constructor'da sadece 1 kere kullanilabilir. Ama super( ) ve this( ) birlikte ayni constructor'da kullanilabilir.

S29: Java da String Pool nedir?

C29:

Heap hafıza içinde String'lerin toplandığı havuz yani yerdir. Bir Object oluşturulduğu zaman onun bu havuzda önceden olup olmadığını kontrol eder.

String Pool, tasarruf sağlar yani boş yere heap de referanslar oluşturmaz. String oluşturmak için çift tırnak kullandığımızda, ilk önce String havuzunda aynı değer ile String'i arar, eğer bulursa referansı döndürür, bulamazsa havuzda yeni bir String oluşturur ve sonra referansı döndürür.

new operatörü kullanılarak, **heap** alanında yeni bir String nesnesi oluşturmuş oluruz. Havuza yerleştirmek için **intern()** yöntemini kullanabilir ve havuzda bu String nesnenin içerdiği dizeye bakılır, bulunursa havuzdaki dize döndürülür. Aksi takdirde, bu String nesnesi havuza eklenir ve bu String nesnesine bir referans döndürülür.

Unutmayınız ki; new operatörü her zaman yeni bir referans oluşturur ve == ile nesnelerin referanslarının eşitliğini kontrol eder. (https://koraypeker.com/2018/04/07/string-pool-nedir/)

S30: Static method ve non-static methodlar arasındaki farklar nelerdir?

C30:

1. Static method: Static keyword'ü method adından önce kullanılmalıdır. Ancak non-static method keyword'ü static kullanmaya gerek yoktur.

2. Static method: Static olmayan değişkenlere (variables) erişilemez. Ancak non-static method bunlara erişebilir.

3.Statik bir metod (static) bir sınıftan üretilmiş olan tüm nesneler için paylaşılır. Bu da pratikte aslında bizim için şu anlama gelir, static metodları nesne yaratmadan doğrudan kullanabiliriz (classAdi.methodAdi). Static olmayan metodları kullanabilmek için ise önce o sınıfa ait yeni bir nesne üretmemiz gerekir.

Araba arb1 = new Araba();

arb1.hiz();

S31. String'ler ve StringBuilder'lar arasındaki farklar?

A31:

1. saklama alanı; String'ler StringPool'da StringBuilder Heap hafızada

2. İş parçacığı güvenliği; String'ler evet StringBuilder no

3. form; String'ler Değişmez StringBuilder değiştirilebilir

4. Performans; String'ler Hızlı StringBuilder daha verimli, özellikle setter ve getter methodlarında.

String Vs Stringbuilder

String

System namespace altında

Değişmez (salt okunur) örnek

Sürekli değer değişikliği gerçekleştiğinde performans düşer

İş parçacığı güvenli

StringBuilder (değiştirilebilir dize)

System.Text ad alanı altında

Değişebilir örnek

Mevcut durumda yeni değişiklikler yapıldığından daha iyi performans gösteriyor

https://www.web-gelistirme-sc.com/tr/c%23/string-vs.-stringbuilder/957383731/

S32: Constructor'lar inherited edilebilir mi?

C32: Hayır edilemez.

Constructorlar inherit edilmez. Ancak türetilmiş sınıfın construtorları yazılırken, temel sınıfın constructorları çağrılmalıdır. Eğer çağrılmazsa, otomatik olarak hiç parametre almayan constructor çağrılır. Fakat biz temel sınıfta hiç parametre almayan constructor implement etmediysek, programımız hata verir. Implement ettiysek sorunsuz çalışır.

https://medium.com/kodcular/inheritance-extends-super-ve-overriding-kavramlari-b15ccef76bd8

https://duygubulut.wordpress.com/2017/11/18/inheritancemiras-alma/

https://hilmi.kulubevet.com/ders_notlari/Java/Nesneye_Yonelik_Programlama.pdf

S33: ClassLoader nedir?

C33: JVM in altkümesi ve class dosyalarının yüklenmesinden sorumlu yapı.

JVM (Java Virtual Machine) ilk çalışırken sınıfların yüklenmesinden sorumlu Class tır diyebiliriz. Java da Interpretation(yorumlama) işleminden önce Class Loading işlemleri gerçekleşir. Her Class uniq olarak ClassLoader tarafından JVM Memory e yüklenir. JVM Class Loading işlemlerini yaparken, yani bir java uygulamasını ilk olarak ayağa kaldırırken, daha uygulamamız hiç çalışmaya başlamadan önce, ClassLoader sınıfından extend ederek sınıf

yükleme işlemlerini özelleştirmemize olanak sağlar. Örneğin aynı isimli sınıfı iki kez özel olarak load edebiliriz, ya da uygulama açılmadan önce güvenlik amaçlı kontroller ya da şifrelemeler yapabiliriz.

S34: Kaç tane ClassLoader vardır ve nelerdir?

C34: Java 3 tane ClassLoader destekler

1. Bootstrap

2. Extension

3. System / Applicatio

Java'da 3 farklı ClassLoader vardır.

Bootstrap Class Loader

Base sınıf dosyalarını yüklemesinden sorumludur.

Extension Class Loader

Ek sınıf dosyalarının yüklenmesinden sorumludur.

Application class Loader

Uygulamamız çalışması gereken sınıf dosyalarını yüklemekten sorumludur.

İmplement ederek kendi Custom Loaderimizi yazmamız mümkündür.

Exception Handling ederek oluşabilecek sorunları yakalayabiliriz.

NoClassDefFoundError

ClassNotFoundException

Bu hataları yakalayarsak ClassLoader'dan oluşacak sorunlaru Runtime'da programı durdurmaz.

https://kerimfirat.medium.com/java-virtual-machine-1-class-loader-ff145a4167d0

http://yazilimgelistirmeyontemleri.blogspot.com/2015/06/java-da-classloader-nedir-nasl-calsr.html

S35: Java'da String'ler neden doğası gereği değişmezdir?

C35: Basitçe, String objesi oluşturulduktan sonra durumunun değiştirilemeyeceği anlamına gelir. Uygulamanın güvenlik önbelleğini, senkronizasyonunu ve performansını artırır.

String çeşitli nedenlerden dolayı değişmez bu nedenler:

Güvenlik: parametreler tipik olarak ağ bağlantılarında, veritabanı bağlantı URL'lerinde, kullanıcı adlarında/şifrelerinde vb. String olarak temsil edilir. Değişebilirse, bu parametreler kolayca değiştirilebilir.

Synchronization ve eşzamanlılık: String'in değişmez hale getirilmesi, dizileri otomatik olarak güvenli hale getirerek senkronizasyon sorunlarını çözer.

Caching: derleyici String nesnelerinizi optimize ettiğinde, iki nesnenin aynı değere sahip olması durumunda (a = "test" ve b = "test") görür ve böylece sadece bir string nesneye ihtiyacınız vardır (hem a hem de b için, bu ikisi aynı nesneye işaret edecektir).

Sınıf yükleme: String sınıf yükleme için değişken olarak kullanılır. Değişebilirse, yanlış sınıfın yüklenmesine neden olabilir (çünkü değişken nesneler durumlarını değiştirir).

https://www.youtube.com/watch?v=oNIJutMfpUY

https://www.web-gelistirme-sc.com/tr/java/string-javada-neden-degismez/1045006025/

S36: Array ve ArrayList arasındaki farklar nelerdir?

C36:

1. Array, farklı veri türlerinin değerlerini içeremez. ArrayList bunları içerebilir.

2. Array size'ı tanımlanmalıdır. ArrayList size'ı dinamik olarak değiştirilebilir.

3. Veri eklemek için Array'in indeksi belirtmesi gerekir. ArrayList buna ihtiyacı yok.

4. Array'ler primitive data ve nesneler (Objects) içerebilir. ArrayList yalnızca objeleri içerebilir.

Array Vs. Arraylist

Array	ArrayList
Array, dizi nesnesi oluşturulduktan sonra uzunluğu değiştirilemeyen sabit uzunlukta bir veri yapıdır.	ArrayList doğada dinamik olup, gerektiğinde büyümek üzere yeniden boyutlandırılabilir.
Bir dizinin boyutu, program boyunca statik kalır.	ArrayList boyutu, yük ve kapasiteye bağlı olarak dinamik olarak artabilir.
Öğeleri depolamak için atama işleci kullanır.	Öğe eklemek için add () özelliğini kullanır.
İlkel nesnelerin yanı sıra aynı veya farklı veri türünün nesnelerini içerebilir.	İlkellere ArrayList'te izin verilmez. Yalnızca nesne türleri içerebilir.
Diziler ve Generics el ele gitmez.	Generics'e ArrayList'te izin verilir.
Diziler çok boyutlu olabilir.	ArrayList tek boyutludur.
Bu, elemanların bitişik bellek konumlarında depolandığı yerli bir programlama bileşeni.	Nesnelerin bitişik yerlerde asla saklanmadığı Java koleksiyonları çerçevesinden bir sınıf.
Uzunluk değişkeni, Dizinin uzunluğunu belirlemek için kullanılır.	Size () yöntemi, ArrayList'in boyutunu belirlemek için kullanılır.
Belirtilen öğeleri veya nesneleri depolamak için ArrayList'ten daha az bellek kullanır.	Nesneleri saklamak için Array'den daha fazla bellek alır.
Bir dizi üzerinden yineleme, bir ArrayList üzerinde yineleme işleminden daha hızlıdır.	Bir ArrayList üzerinde yineleme, performans açısından önemli ölçüde yavaştır.

https://medium.com/@ufukakarsu/arraylist-ve-diziler-arrays-d302ce551efc

https://tr.weblogographic.com/difference-between-array-and-arraylist-845700

S37: Java da Map nedir?

C37: Map, benzersiz anahtarları (keys) değerlerle (values) eşleştiren bir Util Paketi ara yüzüdür. Yinelenen anahtarlar (key) içermez Her anahtar (key) en fazla bir değerle(value) eşlenebilir.

Map, Java Collections Framework 'un bir üyesidir. Map (gönderim) anahtarları değerlere eşleştiren bir nesnedir. Örneğin, bir ad listesinde her ada bir sıra numarası vermek bir Map (gönderim) işlemidir. Bu işlemde sıra numaralarının her biri bir anahtar, her ad bir değer olur. Liste karışmasın diye her ada ayrı bir sıra numarası verilir. Sıra numaraları anahtarlar, adlar ise değerlerdir. Sıra numaraları birbirlerinden farklıdır; ancak adlar farklı olmayabilir. Örneğin, aşağıdaki ad listesinde Selda adlı iki kişi var. Onların her birine ayrı sıra numarası verildi. Sıra numaraları (anahtarlar) içinde dublikasyon yoktur. Ama adlar (değerler) içinde dublikasyon olabilir.

1	Selda
2	Can
3	Selda
4	Murat

Benzer olarak, Map, her değere farklı bir anahtar eşler; anahtarlarda duplikasyon olamaz. Değerlerde duplikasyon olabilir. Anahtar sayısı değer sayısından az olamaz; daha çok olabilir. Yaptığı işe göre, Map bir fonksiyondur. Java, üç ayrı Map kılgılar: HashMap, TreeMap, LinkedHashMap. Bunların davranışları ve performansları, sırasıyla, HashSet,

TreeSet ve LinkedHashSet gibidir. Map arayüzü koleksiyonu üç türlü görmemizi sağlar: anahtarlar kümesi, değerler kümesi ve anahtar-değer eşleşmeleri kümesi. Map içindeki sıralama, koleksiyon üzerinde tanımlanan iteratörün öğeleri bize veriş sırasıdır. Ancak, TreeMap yapısında öğelerin sırası belirlidir. HashMap yapısında ise sıra belirli değildir.

http://www.baskent.edu.tr/~tkaracay/etudio/ders/prg/dataStructures/Collections/InterfaceMap.pdf

https://medium.com/@recepsirin/javada-map-aray%C3%BCz%C3%BC-hashmap-ve-treemap-828696533acf

S38: Java da Collection Class nedir?

C38: Collection bir mimar gibi bir grup nesne objeyi depolamak ve işlemek için hareket eden bir çerçeveden framework tan oluşur.

Java Collections (Koleksiyonlar), nesne grubunu depolamak veya işlem yapma gibi işlemleri sağlayan bir türdür. Şöyle düşünelim, program geliştirirken bir değişken içerisinde birden fazla eleman tutmak isteyebiliriz. Bunun gibi durumlarda **Java Collections (Koleksiyonlar)** sınıfını kullanmak bizim için en verimli seçenek olacaktır.

https://www.kodkampusu.com/java-collections-koleksiyonlar-nedir/

https://emrecelen.com.tr/java-collections-nedir/

https://medium.com/gokhanyavas/java-collections-koleksiyonlar-83cf8bba100c

https://www.baskent.edu.tr/~tkaracay/etudio/ders/prg/dataStructures/Collections/Collections01.pdf

S39: Collection Class'lar neler içerirler açıklayınız?

C39: Interface, class ve methodlar içerirler. List, queue ve set en önemli parçalarıdır Collection Class'ların.

Collection, Interface olup içinde benzer türden nesneleri belirli şekilde tutacak, nesnelere ait temel davranışları belirler. Collection yapısı Java'da bu ihtiyaçları karşılamak için:

add,

addAll,

remove,

clear,

isEmpty gibi metodlar sunar.

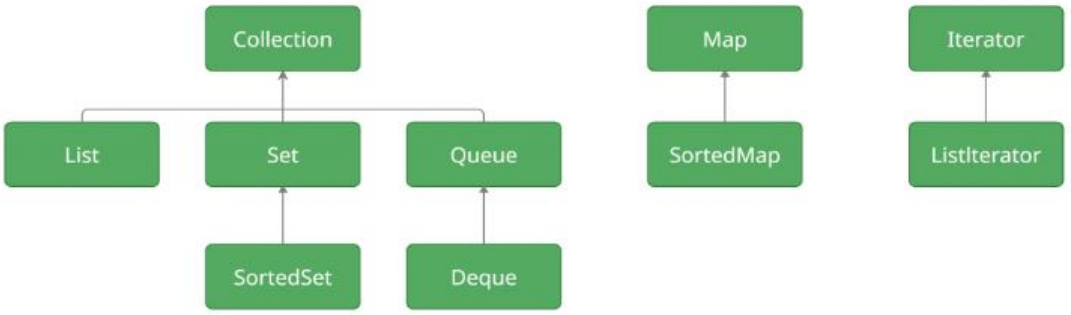
Collection<E> interface'inin altında ise 3 farklı interface vardır :

Set<E>

List<E>

Queue<E>

Bu üç interface collection karakteristik özelliklerinin belirlendiği interface'lerdir.



Collection: Nesne grupları ile çalışmamız yardımcı olur hiyeraşik olarak en tepede bulunmaktadır.

List: Koleksiyonu genişletir sıralı öge koleksiyonlarını barındırır. Liste yapısını örnek alır. İçerisinden bir elemandan fazla eleman bulundurulmasına olanak tanır. Elemanlarını sıralı bir biçimde barındırır

Set: Kopyalanan ögeleri içermez. Matematiksel küme soyutlamasını modeller. Yalnızca miras alınan metotları içerir.

SortedSet: ögelerini artan düzende tutan bir türdür. Sıralamadan yararlanmak için birkaç ek işlem sağlanmaktadır. Sıralanmış kümeler veya kelime listeleri gibi genel olarak sıralı kümeler için kullanılır.

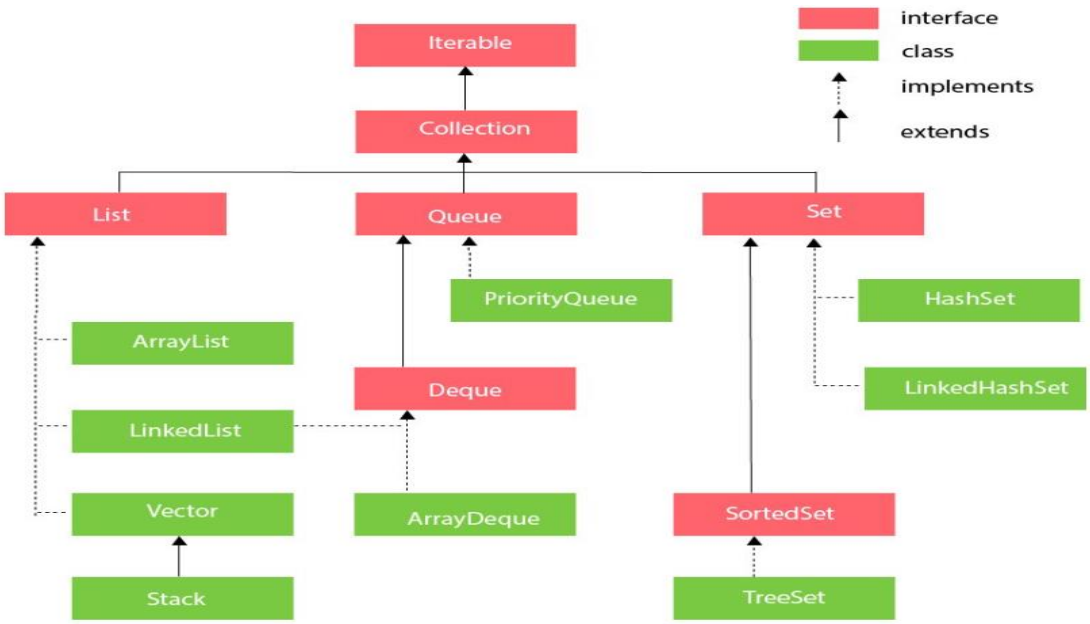
Queue: Kuyruk veri yapısı işlemi sağlar. Yani kuyruk yapısındaki ekleme, silme gibi işlemleri kolaylaştıran metotlara sahiptir.

Map: Benzersiz anahtarları değerleri ile eşleştiren arayüzdür. Örneğin her sipariş numarasının ayrı bir numaraya sahip olması gibi. Collection arayüzünü kullanmazlar.

SortedMap: Map arayüzünün özellikllerini taşır. Anahtar değerleri aratan bir sırada saklanır.

Iterator: Koleksiyon ögelerini tekrarlamak için kullanılır.

ListIterator: Elemanlarımız arasında gezinmemizi sağlar. En önemli özelliği collection elemanlarını remove edebilmesidir.



https://mbilgil0.medium.com/java-collections-dc42832c8af0

S40: Polymorphism nedir?

C40: Bu soruya en iyi örnek; kamera, hesap makinesi, mp3 çalar ya da uzaktan kumanda gibi kullanılan bir cep telefonudur.

Java'da çok biçimlilik (Polymorphism) kavramı birçok türe sahip olmak anlamına gelmektedir. Gerçek hayattan örnek vermemiz gerekirse; bir kişinin birden fazla görevi yapması, misal erkek olan biri, bir koca, bir çalışan veya bir baba olabilir. Yani aynı kişi farklı davranışlara sahip olabilir ve farklı görevlere sahip olabilir. **Java'da çok biçimlilik (Polymorphism)**, bir nesnenin farklı nesneler gibi davranabilmesini sağlayabiliriz, farklı **sınıflardan** oluşturulmuş nesneleri tanımlayabilme gibi özellikler mevcuttur.

https://www.mobilhanem.com/java-polymorphism/

https://emrecelen.com.tr/java-polymorphism-nedir/#close

Q41: Kaç çeşit Polymorphism vardır?

C41:

Polymorphism: Oluşturulan nesnelerin gerektiğinde kılıktan kılığa girip başka bir nesneymiş gibi

davranabilmesine polymorphism diyebiliriz. Bunlar program kodlarının yeniden kullanılabilmesi veya

var olan kodun geliştirilebilmesi açısından çok önemlidir.

Polymorphism 2 çeşittir;

1- Compile time Polymorphism method overloading dir. Java hangi methodu kullanacagina compile time da karar verir. Method Overloading sayesinde ayni isme, ayni body'e, **farkli parametrelere** sahip bir cok method uretip kullanabiliriz

2-Run time polymorphhism method overriding 'dir. Method Overriding sayesinde ayni isme, ayni parametrelere ,farkli body'e sahip bir cok method üretip kullanabiliriz.

S42: Java'da Abstraction (soyutlama) nedir?

C42: Java'da soyutlama (Abstraction)**, nesne yönelimli programlamanın yapı taşlarından biridir.** Java'da soyutlama**, gereksiz ayrıntıları göz ardı ederek bir nesnenin yalnızca gerekli özelliklerini belirleme işlevi olarak da nitelendirilebilir. Bir bilgisayarı kullanan bir kişiyi düşünün, bilgisayarı kullanan kişi yalnızca bilgisayarı kullanır ancak arka planda ne gibi işlemlerin döndüğünü bilemez.** Soyutlama **aslında budur.** Java'da soyutlama işlemini **yapmak için** soyut sınıf (abstract class) **ve** arayüzler (interface) **yapılarını kullanırız.**

S43: Java da Interface ne anlam ifade eder?

C43: **Java** programlama dilinde bir **Interface**, bir sınıfın davranışını belirtmek için kullanılan soyut bir tür olarak tanımlanır. Java'daki bir **Interface**, bir sınıfın planıdır. **Interface** soyut methodlar içerir(Soyut yöntemlerin bir koleksiyonudur.). Bir class bir arayuzu uygular, böylece Interface'in soyut methodlari devralır.

Interface, Java'da bir referans türüdür. Sınıfa benzer. Interface uygulayan class soyut değilse, arayuzun tüm yöntemlerinin sınıfta tanımlanması gerekir.

Interface içinde sadece kendisinden türeyen sınıfların içini doldurmak zorunda olduğu, body'si olmayan method'larin olusturuldugu bir yapıdır.

Kısacası kendisini inherit eden class'lar için, yerine getirmeleri gereken metodları belirten "tum alanlari doldurulmak zorunda olan bir sablon" gibidir.

Interface'ler somutlastirilamaz yani Interface'de obje olusturulamaz.

Interface'ler nasil yapilacagina degil ne yapilacagina odaklanir.

Interface'e sadece abstract public method'lar konabilir

Neden interface

Java çoklu mirası desteklemez.Çoklu miras alabilmek içinde interface çözümlerden biridir.

S44: Interface'in Abstract Class ile arasındaki farklar nelerdir? (Bu soru olmazsa olmazlardan)

A44:

Abstract Class

- Methodlarin gövdeleri olabilir.
- Bir class yalnızca bir soyut Abstract class i genişletebilir (extend edebilir) Birden fazla abstract class'a direkt
- Child olunmaz.
- Soyut olmayan(concrete) methodlara sahip olabilir.
- Ornek değişkeni (instance variable) olabilir.

- Constructor vardir
- hızlı
- ne yapmasi gerektiğini belirlemek ile beraber nasil yapmasi gerektiğini de belirleyebilir.
- Kismi abstraction sağlar.
- herkes tarafından ulaşılan görünürlüğe sahip olabilir public protected

Interface

- methodlarin sadece imzalari olur
- Bir class birkaç Interface uygulayabilir. Coklu inheritance'a uygundur.
- Tüm (ara yüz) Interface methodlar (abstract tir) soyuttur
- Ornek değişkenlere (instance variables) sahip olamaz.
- Constructor içeremez
- yavaş , gerçek class a uygun methodu bulmak için ekstra zamana ihtiyac vardir
- nesnenin ne yapmasi gerektiğini belirler ama nasil yapmasi gerektiğini belirlemez.
- Tam abstraction sağlar
- sadece public olmalı ya da olmamalı

S45: Java da Inheritance (kalıtım-miras) nedir?

A45: Kalıtım(Inheritance), **bir sınıfın kendisine ait özellikleri ve işlevleri bir başka sınıfa aynen aktarması ya da bazı özellik ve işlevlerini diğer sınıfların kullanmasına izin vermesi** şeklinde oluşmaktadır.

Java'da Inheritance'in arkasındaki fikir, daha once'den olusturulmus Class'larin üzerine yeni Class'lar oluşturabilmemizdir.

Inheritance sayesinde yeni olusturdugumuz bir class'in var olan bir class'in tum methodlarini ve variable'larini kullanmasini saglayabiliriz.

Inheritance sayesinde child class, parent class'daki public veya protected primitive datalari, objectleri, veya metodlari problemsiz bir sekilde kullanabilir.

NOT 1: Child classlar public ve protected data'lari problemsiz bir sekilde inherit edebilir.

NOT 2: Private data'lar inherit edilemez.

NOT 3: Default data'lar child ve parent ayni package'da olduklari zaman inherit edilebilirler.

NOT 4: Static Methods veya variable'lar inherit edilemezler.

Üst sınıf (Süper sınıf) özelliklerini alt sınıflarda kullanmak için extends keywordunu kullanılırız.

Kalitımın Faydaları

=>Tekrardan kurtulmak

=>Daza az kod yazarak islemleri yapabiliriz.

=>Kolaylikla update yapılabilir.

=>Application'in bakimi ve surdurulebilirligi kolaylaşır.

S46: Java da alt class (child) nedir?

A46: Bir sınıf başka bir sınıftan türüyorsa yani başka bir sınıfın değişkenlerini ve metotlarını miras alıyorsa **Child** olarak adlandırılır. Alt sınıflar ayrıca türetilmiş sınıflar, alt sınıflar veya genişletilmiş sınıflar olarak da bilinebilir.

Peki alt sınıf, üst sınıfın sahip olduğu tüm özellikler miras almak zorunda mı?

Kesinlikle hayır. Gelecekteki mirasımız gibi birçok Constrcutor modelinde hangi özelliklerden ayrılacağımızı planlıyoruz. Alt sınıf, olan Constructor'ı kullanarak sadece belirli özellikler miras alabilir.

S47: Inheritance çeşitleri nelerdir?

C47:

1)Tek (single)

2)Cok düzeyli(multilevel)

3)Hiyerarşik (hierarchical)

4)Karma (hybrid)

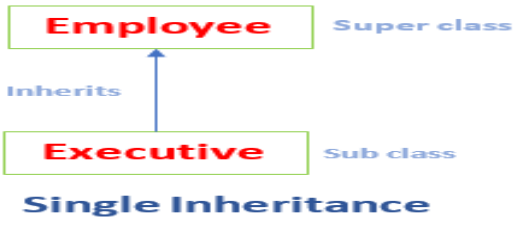
S48: Inheritance çeşitlerini açıklayınız?

C48:

1) Single:

Bir ebeveyn bir çocuk.

Bir child class'in sadece bir parent class'i olabilir.



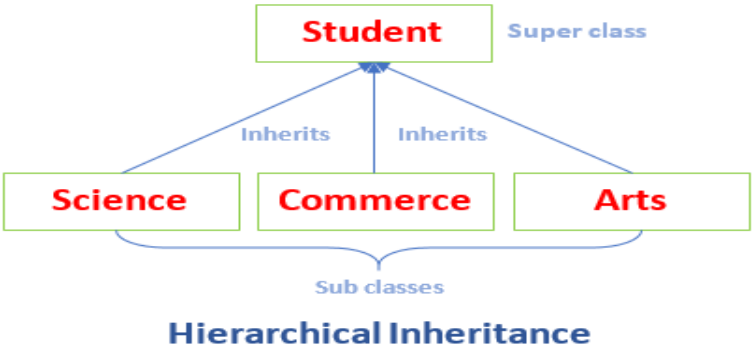
2)Multilevel Inheritance

Insanlardaki soy agaci gibi, child class'in parent'i ve grand parent'leri olabilir.

3)Hierarchical:

Bir ebeveyn daha fazla alt sınıf

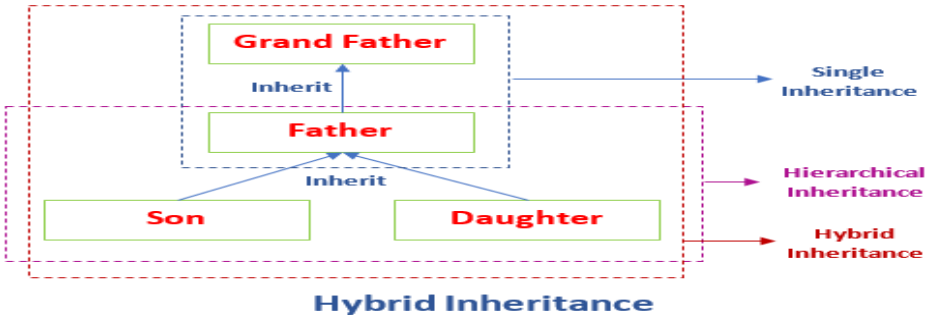
Birden fazla class ayni class'i parent olarak kullanabilir.



4) Hybrid:

İki veya daha fazla kalıtım türünün bir kombinasyonudur.

Hibrit ortalamalar birden fazla oluşur. Hibrit kalıtım, iki veya daha fazla kalıtım türünün birleşimidir.



S49: Java da method Overloading nedir?

C49: Java, aynı isme sahip birden fazla metodlar tanımlamamıza izin vermektedir. Bu şekilde birden fazla aynı metodun yazılmasına overloading, aşırı yükleme adı verilir.

Java Overloading, projelerimizde oluşturmuş olduğumuz metotların yapısal olarak yeniden yazılmasına, esneklik sahibi olmasına ve daha fazla işlevsel bir hal almasına olanak sağlayarak aynı isimde birden fazla metodumuzun oluşmasına imkan verir.

Methodun davranışına daha fazlasını eklemek veya genişletmektir.

Bu bir derleme zamanı (compile time) polimorfizmidir.

Overloading Yaparken Uymamız Gereken Kurallar

Overload yapacağımız metotların isimlerinin aynı olması gerekmektedir.

Metotlarımızın sahip oldukları parametreler birbirinden farklı yada siralamalari farkli olmalıdır. **Örneğin:** metot(**parametre 1**) ve metot (**parametre 1**, **parametre 2**) şeklinde olmalıdır.

Java'da Overloading Kullanımının Avantajları

Aynı isimlere sahip ama farklı parametrelerle birçok metot oluşturma yöntemidir.

Temiz ve sade bir kod düzeni oluşturur.

Kodun okunabilirliğini arttırır.

Overloading, programcılara farklı veri türleri için benzer bir metodu çağırma esnekliğini sağlamaktadır.

Peki biz metodları neden aşırı yüklüyoruz örnekle inceleyelim?

– topla(); (int için)

– ltopla(); (long için)

– ftopla(); (float, double için)

Tek bir işlem için üç farklı isimde metod kullanmanız gerektiğinden gereksiz bir karışıklık yaşanıyor. Tabii bu örnek için herhangi bir karışıklık görünmüyor olabilir fakat devasa bir Java projesini düşündüğümüz zaman benzer işlemleri gerçekleştirmek için farklı farklı metotlar yazmanın nasıl bir karışıklığa yol açacağını tahmin etmek zor değil.

Oysa overloading işlemi sayesinde metodumuzu aşırı yükleyerek tek bir isimle mutlak değer işlemini gerçekleştirmemiz mümkün. Sadece metodumuz 3 farklı parametre ile 3 defa aşırı yüklenecek.

S50: Java da method Overriding nedir?

A50:

Parent class'da varolan bir methodu method signature'ini degistirmeden, method body'sini

degistirerek kullanmaya Method Overriding denir.

Overriding'in amaci methodun mevcut davranışını "değiştirmektir".

Overriding kullanarak, child class'in parent class'daki methodu kendine uyarlayarak (implement) kullanmasini saglamis oluruz.

Overriding yapildiginda parent class'daki methoda Overridden Method, child class'daki methoda Overriding Method denir. Bu bir run time polimorfizmidir.

Overriding Kurallari:

1)Methods aynı Signature'a (imzaya)sahip olmalıdır. (isim ve parametrelere)

2)Alt class, aynı ada sahip aynı methoda sahiptir.

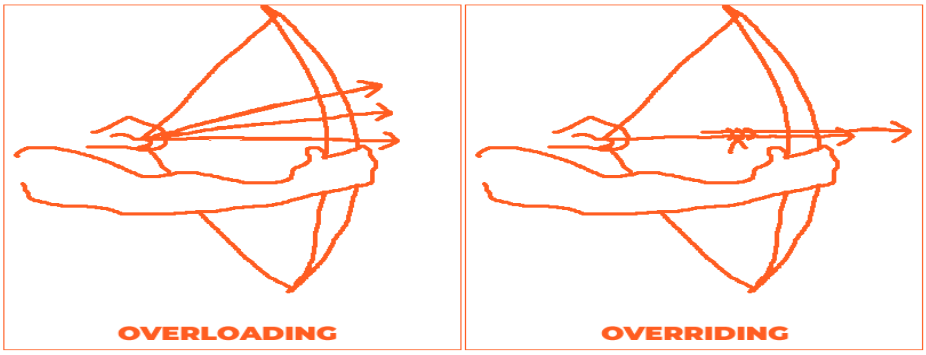
3)Child class'daki method'un (overriding method) Access Modifier'i parent class'daki method'un (overridden) Access modifier'indan daha dar olamaz.

4) private, static and final method'lar overriding yapilamazlar.

Java Overriding Kullanımının Avantajları

Temiz, sade ve anlaşılır kod satırları oluşturmamıza imkan sağlar.

Bir sınıfın hangi yapıda duruş sergileyeceğini ve metotların işlevlerinin nasıl uygulanacağını tanımlamamıza imkan sağlar.



S51: Özel ve statik methodu override edebilir misiniz?

C51: Hayır yapamazsınız, önce class içinde **inheritance** yapmanız lazım.

Override (ezmek/geçersiz kılmak) mekanizmasının temel amacı sınıfların aynı metodlara farklı cevap vermesini sağlayabilmek. Yani elimizde bir adın ne sorusuna baba sınıf "benim adım X" diye cevap verirken oğul sınıfın "ben Y" diye cevap vermesi. Buradaki fark X ve Y değişkenlerinin dışında kalan kısmın (mesajın)değişmesi.

Bu şekilde düşünürsek adınNe sorusu aslında direk olarak bir sınıfın hafızadaki örneği (instance) ile ilişkilendiriliyor. Static metodlar bir sınıfın örneğine üzerinden değil sınıfın meta bilgisi üzerinden çalıştığı için override etmek mantıklı görünmüyor.

Kısa cevap hayır java izin vermez.

S52: Java multiple Inheritance i destekler mi?

C52: *Java bunu desteklemez*. Çünkü farklı ebeveyn sınıflarında aynı method adı var ise hangi methodun çalıştırılacağını derleyicinin compiler karar vermesi zordur. [Buradaki multilevel demek birden fazla parent class demektir, yoksa Java hybrid ve hierarchical gibi çoklu Inheritance'lari destekliyor]

(iki babalı ya da iki anneli bi çocuk olmaz. C++ da bunu destekliyor ama Java'da desteklemiyor) C++'da var da neden Java'da bu iş yok diye düşünebilirsiniz ama bu tamamen dilin dizayn aşaması ile ilgili bir durum. Çok kullanılmaması ve yanlış sonuçlara yol açabilmesi, ayrıca tam anlamıyla hiçbir zaman çoklu kalıtım diye birşeyin olmayacak olması Java'yı geliştirenleri böyle bir duruma yöneltmiş.

S53: Java da Encapsulation (kapsülleme) nedir?

C53: Encapsulation bir sınıfın içersindeki metot ve değişkenleri korumasıdır. Bünyesindeki metot veya değişkenlere erişmenin sakıncalı olduğu durumlarda **Kapsülleme** mekanizması kullanılır. Kapsülleme sayesinde sınıf üyeleri dış dünyadan gizlenirler. Bu gizliliğin dereceleri vardır ve bu dereceyi Erişim belirleyiciler (Access Modifiers) tanımlar. (public-private-protected- default)

S54: Java da association nedir?

C54: Association, sınıflar arasındaki bağlantının zayıf biçimine verilen addır.

Bu bağlantı oldukça gevşektir. Yani sınıflar kendi aralarında ilişkilidir lakin birbirlerinden de bağımsızdırlar! Parça bütün ilişkisi yoktur

Örneğin; bir otobüsteki yolcular ile otobüs arasındaki ilişki Association'dır. Nihayetinde hepsi ayni yöne gitmektedir ancak bir yolcu indiğinde bu durum otobüsün ve diğer yolcuların durumunu değiştirmez.

https://www.youtube.com/watch?v=9-xHO-MKIew

S55: Java da aggregation ile neyi kastediyorsun?

C55 : Aggregation ise nesneleri birleştirip daha büyük bir nesne yapmaya verilen addır. Aggregation ilişkisinde association'da olmayan bir sahiplik ve parça-bütün ilişkisi vardır. Bunu fabrika üretimi olan nesneler üzerinden gözlemleyebiliriz. Çünkü fabrika üretimi nesneler genellikle aggregation ilişkisini temsil ederler. Örneğin fabrika çıkışlı bir mikrodalga nesnesi bir kabinden, bir kapıdan, gösterge panelinden, butonlardan, bir motordan, bir magnetron'dan falan oluşur. Dikkat ederseniz tüm bu nesneler toplana toplana bir mikrodalga nesnesini oluşturmaktadır. İşte büyük nesne küçük nesnelere *"sahiptir".* Bu yüzden büyük nesnenin küçük bileşenleri ile olan ilişkisine aggregation denir. Fakat association'daki gevşekliğe nazaran aggregation'da yakın bir bağlılık ilişkisi söz konusudur. Çünkü düşünün ki mikrodalganın motorunu çıkardık. Bu durumda kalan parçalar bir mikrodalga özelliği gösteremeyecektir. Halbuki association örneğimizde bir yolcu indiğinde kalan parçalar halen işlevsel durumda devam edebiliyorlardı.

Association ve Agregation arasındaki farkı bulmak için '*Nesnelerin birini aralarından çıkarırsam ne olur?*' sorusunu sorabilirsiniz. Bu sorunun cevabı eğer sorun yok ise association, eğer sorun var ise agregation'dır.

https://www.includekarabuk.com/kategoriler/genel/Association-vs.- Aggregation-vs.-Composition.php

S56: Java da composition nedir?

C56: Composition sahip olunan nesnenin sahip olan nesneden bağımsız bir şekilde var olamamasına denir. Örneğin; araba ve tekerlek nesnelerini düşünelim. Bir araba tekerleğe "sahiptir". Tekerleksiz araba düşünülemez. Bu yüzden bu nesneler arasında sıkı bir ilişki vardır. Böylece association'ı elemiş olduk. Fakat bir tekerlek arabasız olabilir. Yani sonuçta raflarda satılacak tekerlekler arabayla yer almaz. Dolayısıyla böyle de bir ilişkide esneklik vardır. İşte bu örnekte sahip olunan tekerlek nesnesi kendi başına ayrı olarak var olabildiği için bu iki nesne arasındaki ilişkiye agregation'dır deriz. Yani sahip olunan nesnenin sahip olan nesneden bağımsız bir şekilde var olabilmesine agregation denir.

Composition ise sahip olunan nesnenin tek başına var olamayacağı durumu ifade etmek için kullanılan bir ilişkidir. Mesela vücut ve kan hücresi nesnelerini ele alalım. Sizce normal şartlar altında vücudun olmadığı yerde kan hücresi olur mu? Olmaz. İşte bu daha sıkı ilişkiye composition adı verilir. Yani sahip olunan nesne olan kan hücresi tekerleğin aksine tek başına var olamayacağı için bu ilişkiye composition denir. Fakat aklınıza şu soru gelmiş olabilir:

Hastanelerdeki şişelerde hastalardan alınan kanlar varken bir kan hücresi vücuddan ayrı bir şekilde var olamaz mı? Evet, olur. Fakat burada ilişkinin agregation mı composition mı olduğunu *iş sahası* belirliyor. Eğer bir hastane için yazılım geliştiriyorsanız vücut ile kan hücresi agregation ilişkisine tekabül eder. Çünkü kan hücresi vücuttan ayrıyeten var olabiliyor. Eğer bir fabrika için yazılım geliştiriyorsanız vücut ile kan hücresi composition ilişkisine tekabül eder. Çünkü bir kan hücresi personelin vücudundan ayrıyeten olamaz. Yani bu ilişki, iş sahasına göre görecelidir.

Sonuç olarak bir ilişkinin agregation mı composition mı olduğunu anlamak için ilişkideki sahip olunan nesnenin tek başına var olup olamayacağını saptamak yeterlidir. Tabi iş sahasını göz önünde bulundurarak...

S57: İşaretleyici Interface nedir?

C57: Marker Interface Pattern;(İşaretleyici Interface) kod yazma süreçlerinde derleyicinin nesneler hakkında ek bilgilere sahip olabilmesini ve böylece ilgili nesnenin kullanılacağı noktaları derleme sürecinde kurallar eşliğinde belirleyerek, kodu runtime'da olası hatalardan arındırmamızı sağlayan bir pattern'dır.

Özünde, belirli görevlere istinaden belirli arayüzlerle ya da class'larla işaretlenmiş nesneleri devreye sokabilmemizi sağlar ve böylece bu işaretlenmiş nesnelerin dışında o görevlerde farklı türlerde nesnelerin kullanılmasına müsaade etmeyerek kodu daha kontrollü hale getirebilme imkanı tanıyaran bir stratejik manevra oluşturur.

https://www.gencayyildiz.com/blog/marker-interface-patternisaretleyici-arayuz- tasarimi/

S58: Object klonlaması nedir?

C58: Klonlama aynı zamanda bir nesne yaratmanın bir yoludur, ancak genel olarak klonlama sadece yeni bir nesne yaratmakla ilgili değildir. Klonlama, halihazırda var olan bir nesneden yeni bir nesne oluşturmak ve verilen nesnenin tüm verilerini o yeni nesneye kopyalamak anlamına gelir.

Fakat klonlanmış objenin methodu Protected olur, bu yüzden bu methodun **override** edilmesi gerekir.

S59: Constructor Overloading (yapıcı aşırı yükleme) nedir?

C59: Java, aynı isme sahip birden fazla metodlar tanımlamamıza izin vermektedir. Bu şekilde birden fazla aynı metodun yazılmasına overloading, aşırı yükleme adı verilir.

Bunun için belirli şartlar gerekiyor. Overloading yapabilmek için aynı isimdeki metodlarımıza farklı parametreler göndermemiz gerekiyor. Burada parametrelerimiz türü ve/veya sayısının farklı olması yeterli.

Java için bu işlem oldukça basit. Aşırı yükleme yaptığınızda ne kadar aşırı yüklerseniz yükleyin Java hangi metotun çalışacağını biliyor. Metotların içeriği, geri dönüş değerleri hiç önemli değil. Aşırı yükleme yaparken tek dikkat etmemiz gereken nokta parametreler. Programınız çalışırken Java metodunuza göndermek istediğiniz değere bakıyor. Diyelim ki **String** türünde bir değer kullanmak istiyoruz. Java bunu gördüğü anda aşırı yüklenmiş metodumuzun String parametresi alan bir durumu var mı kontrol ediyor. Eğer varsa çalıştırıyor.

http://kod5.org/javada-metodlari-asiri-yukleme-overloading/

S60: Java da Exception in nasıl üstesinden geliriz?

C60: 5 tane keywords var. Try, catch , finally, throw ve throws ile.

S61: Error ve Exception arasındaki fark nedir?

C61: Hata (error): Çalışma zamanında ortaya çıkan düzeltilemez bir durumdur.

İstisna (exception): Hatalı girdi nedeniyle ortaya çıkan durumdur veya insan hatası vb. Çoğu durumda kurtarmak mümkündür.

Ek: Error :OutOfMemoryError gibi hande edilemeyen hatalardır. Fakat exception tipindeki hataları try-catch ile ele alabiliriz.

S62: Checked ve unchecked exceptions arasındaki fark nedir?

C62: Checked Exceptions: Derleme zamanında kontrol edilirler.

Ör: IO istisnası SQL istisnası

Unchecked Exceptions: Derleme sırasında kontrol edilmezler.

Ör: aritmetik istisna, nullpointer istisnası. (bu soru interviewlerde çokça soruluyor dikkat edelim.)

Ek:

Fonksiyonel olarak aralarında hiç bir fark yoktur.

Checked exceptionları try-catch bloğunda handle etmezsek derleme sırasında hata verecektir.

Unchecked exceptionlarda böyle bir zorunluluk yoktur fakat biz yine de handle ederek hata sonucunda bir aksiyon aldırmayı tercih ediyoruz.

Checked exceptionlara örnek olarak IOException, SqlException exceptionlarını verebiliriz.

Unchecked exceptionlara örnek olarak NullPointerException, OutOfMemoryError ve ArithmeticException gibi exceptionları verebiliriz.

S63: Final, Finally ve finalize keywordlerin kullanım amaçları nelerdir?

C63:

Final: Class methodu ve değişken variable üzerinde kısıtlamalar uygulamak için kullanılır.

Final class (miras) inheritance olamaz

Final method geçersiz kılınamaz (override edilemez)

Final variable değeri değiştirilemez

Finally: Önemli kodu yerleştirmek için kullanılır, çalıştırılır.

İstisnanın (exception) işlenip işlenmediği önemli değil.

Finalize: Temizleme işleminin hemen öncesinde gerçekleştirilmesi için kullanılır

Nesne çöp toplayıcı (garbage collector) çalışmasından önce. (önemli interview sorularindan)

Ek:

final :

sınıfların başına geldiği zaman sınıfın extend edilememesi (kalıtlamaya kapalı olması)

değişkenlerin başına geldiği zaman değişkenin değerinin değiştirilmeye kapalı olması

metodların başına geldiği zaman metodun override (ezilme) edilememesi

anlamına gelir.

finally :

try-catch-finally veya try-finally şeklinde kullanılır. Bir hata çıksın veya çıkmasın try bloğuna giren kodun sonuçta kesin çalıştırması gereken işlerin yapılması için kullanılır.

```
try {
  potansiyelHata()
  ......calismayacakKod()
}catch(Exception e) {....}
finally {
   kesinCalisacakKod()
}
```

potansiyelHata fonksiyonu bir hata(exception) fırlattığı zaman calismayacakKod() fonksiyonu hiçbir zaman çalışmadan geçecek. Ama her halükarda kesinCalisacakKod fonksiyonunun çalışması garanti altına alınmış olur.

finalize:

Java'da bildiğimiz gibi yıkıcı (desctuctor) fonksyion yok, garbage collector var. Garbage collector çalışmadan önce sınıfın son yapacağı işlemler finalize metodunda yapılır. Bir nevi sınıf ölmeden önce yapacağı şeyleri yapmasına olanak tanır.

S64: throw ve throws arasındaki farklar nelerdir?

C64:

1. **throw** (fırlatma), açıkça bir istisna exception atmak için kullanılır

throws (atar) ise, bir istisna exception bildirmek için kullanılır

2. **throw** ardından bir örnek gelir

throws ardından class gelir

3. method içinde methodla beraber **throw** kullanılır

throws, method un imzası içinde kullanılır

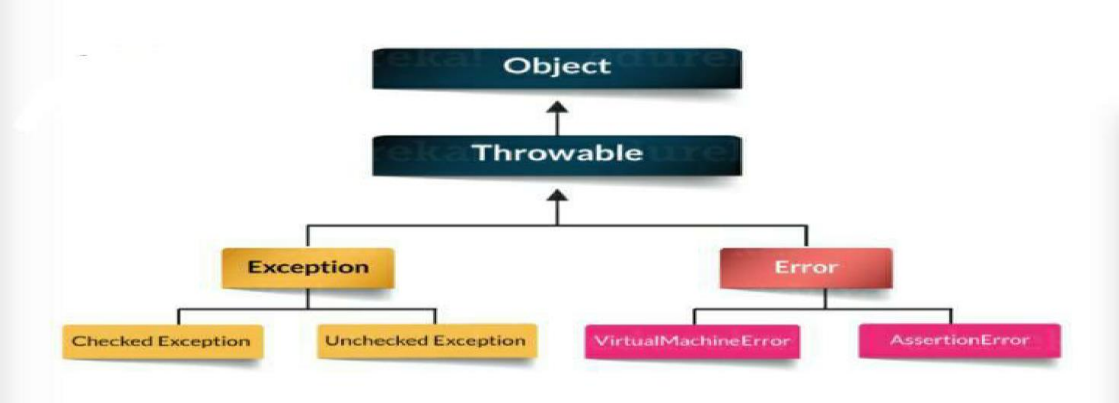
4. **throw** ile birden fazla exception yapamayız,

throws ile birden çok exception bildirebiliriz.

Ek :

https://ekremcandemir.wordpress.com/2015/06/23/23-throw-ve-throws/

S65: Java da exception hiyerarşik yapısı nasıldır?



S66: Özel bir istisna exception nasıl yapılır?

C66: Kendi Exception'ınızı oluşturmak için exception class a extends edin veya onun alt class'larından herhangi birinden extends edin.

Ek:

http://www.ilkaygunel.com/java-exception-handling-best-practices/

S67: Process ve threads arasındaki farkları açıklayınız?

C67:

1. Çalışma:

Süreç (prosess): ayrı bellek alanları

Konular, diziler (threads): paylaşılan bellek alanları

2. Kontrol:

Process: işletim sistemine göre

Threads: bir programdaki programcı tarafından

3. Biçimler:

processs: bağımsız

threads: bağımlı

Ek:

https://celeyrum.wordpress.com/2018/02/06/process-thread-nedir-ve-arasindaki-farklar-nelerdir/

S68: **Finally block** nedir?

C68: Her zaman bir dizi ifadeyi yürüten bir bloktur.

Her zaman bir try bloğu ile ilişkilidir.

Ek:

Finally Bloğu:

Gelelim Finally kısmına. Bu bloklarda Finally eklenmesi durumunda her ne olursa olsun, her durumda ve her koşulda Finally bloğu içindeki işlemlerin yapılması beklenir. Yani hata olmadı diyelim; try içindekiler ve finally içindekiler, hata oldu diyelim; catch içindekiler ve finally içindeki kodlar çalışır. Gelelim bunu örneklemeye yine yukarıdaki örnekten gidiyorum, Finally bloğunda bir mesaj verdirelim.

```
private void btnCarp_Click(object sender, EventArgs e)
        {
            try
            {

                int sayi1, sayi2, carpim;

                sayi1 = Convert.ToInt32(txtSayi1.Text);
                sayi2 = Convert.ToInt32(txtSayi2.Text);
                carpim = sayi1 * sayi2;
                MessageBox.Show("Sonuç: " + carpim.ToString());
            }

            catch (Exception hata)
            {
                MessageBox.Show(hata.ToString());
            }

            finally
            {
                MessageBox.Show("Ne yaparsan yap, bu mesajdan kaçış yok:)");
            }
        }
```

Hata alsakta, almasakta ne yaparsak yapalım, finally içindeki koddan kaçamayacağız.

S69: Finally block un çalışmayacağı bir durum var mı?

C69: Evet var.

Program System.exit () ile çıkarsa veya önemli bir hatayla çıkış yaparsa çalışmaz.

S70: Senkronizasyon nedir?

C70:

-Senkronize edilmiş bir kod bloğu, bir seferde yalnızca bir iş parçacığı ile calisir.

-Java birden fazla iş parçacığının yürütülmesini desteklediğinden, iki veya daha fazla iş parçacığı aynı alanlara veya objelere erişebilir.

-Senkronizasyon, tüm eşzamanlı iş parçacıklarını tutan bir işlemdir.

-Senkronizasyon, paylaşılan hafızanın kararsız görünümü nedeniyle bellek kararsızlık hatalarını önler.

(NOT: Genelde mülakatlarda bu açıklamanın pesinden Selenium'da senkronize meselesi soruluyor)

S71: Birden fazla catch'i tek bir try blok altında uygulayabilir miyiz?

Evet uygulayabiliriz ancak yaklaşım özelden genele doğru olmalıdır.

S72: "OutOfMemoryError" (bellek hatası) nedir?

OutOfMemoryError, java-lang'in alt classıdır. Genellikle JVM belleği dolduğunda oluşan hatadır. Yetersiz bellek, programlar veya işletim sistemi tarafından kullanılmak üzere ek belleğin ayrılamadığı, genellikle istenmeyen bir bilgisayar çalışması durumudur.

S73: Java Exception Class'ın en önemli methodları nelerdir?

1. String getMessage()

2. public StackTraceElement[] getStackTrace()

3. Synchronized Throwable getCause()

4. String toString()

5. void printStackTrace()

S74: Thread (İş Parçacığı) nedir?

-İş parçacığı, programlanmış talimatların en küçük parçasıdır. Bir programlayıcı tarafından bağımsız olarak yürütülebilir.

Java'da, tüm programların en az bir iş parçacığı olacak ana iş parçacığı olarak bilinir.

Bir process'in birden fazla işi aynı anda yapmasını sağlayan yapılara denir.

Bir process bünyesinde bir ya da birden fazla **thread** barındırabilir. **Thread**'ler aynı anda sadece tek bir iş yapabilir. Kısaca N adet **thread** N adet iş yapabilir diyebiliriz.

S75: İş parçacığı oluşturmanın yolları nelerdir?

-Java'da iş parçacıkları aşağıdaki iki yolla oluşturulabilir;

Runnable ara yüzünü uygulayarak

Thread'i genişleterek.

* Runnable: Bir çalıştırılabilir temel olarak, *iş parçacığının ne yapması gerektiğini açıklayan bir iş parçacığına eklenebilecek bir sınıf türüdür* (Çalıştırılabilir Bir Arayüz).

S76: Garbage Collection nedir?

-Java'da çöp toplama, örtük implicit olarak hafıza yönetimine yardımcı olan bir programdır. Java'da yeni anahtar sözcüğü kullanarak dinamik olarak objeler oluşturulabilir, herhangi bir kez oluşturulduktan sonra biraz hafıza tüketir.

Garbage Collection, *otomatik bellek yönetimi mekanizmasıdır.* Bu işlem heap belleğe bakıp, kullanılan objelerin tespit edilmesi ve referans edilmeyenlerin silinmesi üzerine kuruludur. Kullanılmayan/referans edilmeyen nesnelerin kapladığı alan bellekte boşa çıkarılır ve bellekte boş yer açılmış olur.

S77: Java'da kaç farklı Garbage Collector çeşidi vardır?

-4 çeşittir;

1. Serial Garbage Collector
2. Parallel Garbage Collector
3. CMS Garbage Collector
4. G1 Garbage Collector

S78: Java da delete next, main exit veya null keyword olarak kullanılır mı?

-Hayır, onları keyword olarak kullanamayız.

S79: JVM tarafından kaç tür bellek alanı bulunur?

-5 tanedir;

1. Class (Method) Area

2. Heap

3. Stack

4. Program Counter Register

5. NativeMethodStack

S80: Yerel değişkenler (local variables)'in varsayılan değeri (default value) nedir?

-Yerel değişkenler (local variables), herhangi bir varsayılan değerle (default value) başlatılmaz.

Hazırlayanlar:

(1-10 : Akın Alkan) (11-20 : Emre Sulukan) (21-30 : Enes Türkmen) (31-40 : Zeliha Öznük) (41-50 : Büşra Türkoğlu)

(51-60 : Yasemen Sönmez) (61-70 : Deniz Taşkıran) (71-80 : Sümeyye Geçici)