**JAVA INTERVIEW SORULARI**

**S1: Java platformu neden bağımsızdır?**

C1: Herhangi bir sistemde çalışabilen bayt kodlarından dolayı.

**S2: Java neden %100 nesne yönelimli değil?**

C2: Çünkü nesne olmayan 8 primitive data türünü kullanır.

**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.

**S4: Wrapper Class’lar nelerdir?**

C4: Wrapper Class’lar Java primitive datalarını o sınıfın başvuru türlerine veya o

class’ın nesnelerine dönüştürürler.

**S5: Java da Constructor nedir?**

C5: Bir nesneyi başlatmak için kullanılan bir kod bloğunu ifade eder.

Sınıfın adıyla aynı ada sahip olmalıdır.

**S6: Kaç çeşit Constructor vardır?**

C6: 2 çeşit Constructor vardır.

**1) Default Constructor**: herhangi bir girdi ya da değer almaz.

**2) Parameterized Constructor**: durum değişkenlerini sağlanan değerlerle başlatabilir.

**S7: Bir Class i nasıl singleton yapabiliriz?**

C7: O Class’ta ki Constructor i **private** yaparak.

**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.

**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.

**S10: 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.

**S11: Java da package nedir?**

C11: İlgili Class’ların toplandığı yerdir.

**S12: Package’ların avantajları nelerdir?**

C12: 1. İsim çatışmasını önler.

2. Kodlara kolay ulaşım sağlar.

3. Gizli Class’ları barındırır ki Class dışından görülemesin.

**S13: Neden pointer lar Java da kullanılmazlar?**

C13: Çünkü güvenli değillerdir ve program içindeki karmaşıklığı artırırlar.

**S14: Java da JIT derleyici (compiler) nedir?**

C14: JIT derleyici yani tam zamanlı derleyici, Java bayt kodlarını talimatlara

dönüştürmeye yardımcı olan bir programdır.

**S15: Java da access modifier lar nelerdir?**

C15: Default, Private, Public ve Protected.

**S16: Java’da ki access modifier lari açıklayınız?**

C16: Default: ayni package ve class tan ulaşılabilir.

Public: tüm package lar ve class tan ulaşılabilir.

Private: sadece ayni class tan ulaşılabilir.

Protected: sadece farklı pakette ama alt olmayan class tan ulaşılamaz. Diğer şekillerde ulaşılabilir.

(Uyarı: Protected a dikkat edin lütfen yani ayni package ya da farklı package olması önemli değil sadece farklı package in alt class in da olmayan dosyadan erişim sağlanamıyor.)

**S17: Java da class i tarif ediniz?**

C17: Class, datalarımızı barındıran içinde field variable’ları ve method’ların olduğu bizim çalışma alanımızdır.

**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.

**S20: Java da OOPs nin ana konsepti nelerdir?**

C20: İnheritance, Encapsulation, Abstraction, Polymorphism.

**S21: Bunları (OOPs nin ana konseptleri) açıklayınız?**

C21:

**Inheritance** (*kalıtım-miras*): Bir sınıfın başka bir sınıfın özelliklerini elde ettiği miras aldığı bir süreçtir.

**Encapsulation**(*kapsülleme*): Verileri sarmalama mekanizmasıdır.

**Abstraction**(*soyutlama*): Kullanıcıdan var olan detayları gizleme metodolojisidir.

**Polymorphism**: Bir değişkenin birden çok form alma yeteneğidir.

**S22: Local variables ile instance variables arasındaki fark nedir?**

C22: Local variables: tipik olarak bir method, constructor veya block içinde kullanılır ve yalnızca yerel kapsamı vardır. Instance variables: objesinin kendisine bağlı bir değişkendir. Bir class içinde ancak method dışında bildirilirler.

**S23: Java da Constructors ve Methods arasındaki farklar nelerdir?**

C23: **1. Constructors**: Bir objenin durumunu başlatmak için kullanılır.

**Methods**: Bir objenin davranışını temsil etmek için kullanılır.

**2. Constructors**: Herhangi bir dönüş türü (return type) yoktur.

**Methods**: Bir (return type) dönüş türüne sahip olmalıdır.

**3. Constructors**: (Örtülü) İmplicitly olarak çağrılır

**Methods**: (Açıkça) Explicitly çağrılması gerekiyor

**4. Constructors**: Class adıyla aynı isimde olmak zorundadır.

**Methods**: Olabilir veya olmayabilir de.

**S24: Java’da final keyword unu açıklayınız?**

C24: Final Java özel bir keyword dur ve non-access modifier (erişim olmayan

değiştirici) olarak kullanılır.

**S25: Final keyword un kullanımını açıklayınız?**

C25: Final keywordunu bir variable ile kullandığımızda onun value yani değeri bir kez atandığı için değiştirilemez.

Final Methodlar override edilemez. Final Class’lar extends edilemez.

**S26: Break ve Continue yapıları arasındaki farklar nelerdir?**

C26:

1. Break Switch ve Loop’larda kullanılır.

Continue sadece Loop’larda kullanılır.

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.

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

C28: **This()** aynı class in varsayılan Constructor’larını çağırmak için kullanılır, ancak **super()** onları ebeveyn / temel class tan çağırmak için kullanılır.

Not: this () ve super () (anahtar sözcükleri) keywords bir bloğun ilk satırında yer almalıdırlar.

**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.

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

C30:

1. **Static method**: Static anahtar sözcüğü method adından önce kullanılmalıdır.

Ancak **non-static** method anahtar kelimesini statik kullanmaya gerek yoktur.

**2. Static method**: Class kullanılarak çağrılır

Ancak **non-static** methodlar genel methodlar gibi çağrılabilir.

**3. Static method**: Statik olmayan değişkenlere (variables) erişilemez.

Ancak **non-static** method bunlara erişebilir.

**S31. String’ler ve StringBuilder’lar arasındaki farklar?**

C31:

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 methodlarinda

**S32: Constructor’lar inherited edilebilir mi?**

C32: Hayır edilemez.

**S33: ClassLoader nedir?**

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

**S34: Kaç tane ClassLoader vardır ve nelerdir?**

C34: Java 3 tane ClassLoader destekler

*1. Bootstrap*

*2. Extension*

*3. System / Application*

**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.

**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.

**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

**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.

**S39: Collection Class’lar neler içerirler açıklayınız?**

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

**S40: Polymorphism nedir?**

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

**Q41: Kaç çeşit Polymorphism vardır?**

C41: 2 çeşittir;

1- Compile time Polymorphism method overloading dir.

2- Run Time Polymorphism dir inheritance and interface tarafından yapılır.

**S42: Java’da Abstraction (soyutlama) nedir?**

C42: Temelde ayrıntıları gizlemek ve önemli olanları kullanıcıya göstermekle

ilgilenir. İki yol olabilir soyutlama sınıfları (Abstraction class) ve ara yüzleri (Interface)

**S43: Java da Interface ne anlam ifade eder?**

C43: Abstraction method ve statik sabitlerin birleşimidir. Herhangi bir Constructor’a sahip değildir. Boş gövdeli birbiriyle alakalı methodlar dan oluşan bir gruptur.

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

C44:

**1.Abstract Class**: Overridden yapılmasını sağlayabilir.

**Interface**: sadece imzada yer alır, herhangi bir kod sağlayamaz.

**2. Abstract Class**: bir class yalnızca bir soyut Abstraction class i genişletebilir (extend edebilir)

**Interface**: bir class birkaç Interface uygulayabilir

**3. Abstract Class**: soyut olmayan method sahip olabilir

**Interface**: Tüm (ara yüz) Interface methodlar (abstract tir) soyuttur

**4. Abstract Class**: örnek değişkeni (instance variable) olabilir

**Interface**: örnek değişkenlere (instance variables) sahip olamaz

**5. Abstract Class**: herkes tarafından ulaşılan görünürlüğe sahip olabilir public protected

private

**Interface**: sadece public olmalı ya da olmamalı

**6. Abstract Class**: Constructor içerebilir

**Interface**: Constructor içeremez

**7. Abstract Class**: hızlı

**Interface**: yavaş , gerçek class a uygun methodu bulmak için ekstra zamana ihtiyaci var.

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

A45: Bu bir kavramdır ve kodun yeniden kullanılmasına yardımcı olur. Farklı class lar arasında bir ilişki kurar.

İsimleri ebeveyn class ı ve alt class ıdır.

**S46: Java da alt class (child) nedir?**

A46: Özellikleri miras alan bir class, alt class olarak tanınır.

**S47: Inheritance çeşitleri nelerdir?**

C47: **tek** (single), **çok düzeyli(**multilevel**), hiyerarşik** (hierarchical) ve **karma** (hybrid)

**S48: Açıklayınız?**

C48:

**Single**: Bir ebeveyn bir çocuk

**Multilevel**: : Daha fazla ebeveyn bir çocuk

**Hierarchical**: Bir ebeveyn daha fazla alt sınıf

**Hybrid**: İki veya daha fazla kalıtım türünün bir kombinasyonudur.

**S49: Java da method Overloading nedir?**

C49: Methodun davranışına daha fazlasını eklemek veya genişletmektir.

Bu bir derleme zamanı (compile time) polimorfizmidir.

Methodların farklı imzaları olmalıdır.

Aynı Class’ta ki methodlar aynı adı paylaşır ancak her methodun farklı sayıda parametresi olmalıdır. Veya farklı tür ve sıraya sahip parametreler.

Class’ta mirasa inheritance ihtiyaç duyabilir veya gerekmeyebilir.

**S50: Java da method Overridden nedir?**

A50: Methodun mevcut davranışını "değiştirmektir".

Bu bir çalışma zamanı polimorfizmidir.

Methods aynı imzaya sahip olmalıdır.

Alt class, aynı ada sahip aynı methoda sahiptir.

Tam olarak aynı sayıda ve türde parametreler ve süper class olarak aynı return type olmalıdır.

**S51: Özel ve statik methodu override edebilir misiniz?**

C51: Hayır yapmazsınız, önce class içinde inheritance yapmanız lazım. Buda imkânsız özel ve static olduğu için method.

**S52: Java multiple Inheritance i destekler mi?**

C52: Java bunu desteklemez. Çünkü derleyicinin compiler karar vermesi zor, hangi

methodun çalıştırılacağını, aynı method adı var ise farklı ebeveyn sınıflarında. (52. soruda sunu kaçırmayalım burada ki multilevel demek bir den fazla parent class olayı yoksa Java hybrid ve

hierarchical gibi çoklu Inheritance lari destekliyor) (Derste burada iki babalı ya da iki anneli bi çocuk olmaz demişti Süleyman Bey. C++ da bunu destekliyor ama Java’da desteklemiyor)

**S53: Java da Encapsulation (kapsülleme) nedir?**

C53: Veriler dış dünyadan gizlidir ve sadece mevcut class methodları aracılığıyla erişilebilir. Değişkenlerin değerlerini (value of the variables) değiştirmek ve görüntülemek için public method sağlanır.

**S54: Java da association nedir?**

C54: Tüm objelerin sahip olduğu bir ilişkidir. Kendi yaşam tarzları vardır ve sahipleri

yok. Bu ilişkiler bire bir veya çok olabilir, çoktan bire ve çoklu ilişkilerde mevcuttur.

**S55: Java da aggregation ile neyi kastediyorsun?**

C55: Özel bir association biçimidir. Objelerin kendi yaşam döngüleri vardır, ancak

burada bir sahiplik vardır ve alt obje başka bir ana objeye ait olamaz.

**S56: Java da composition nedir?**

A56: Yine özel bir aggregation biçimidir ve buna ölü bir ilişkisi diyebiliriz.

**S57: İşaretleyici Interface nedir?**

C57: Veri üyesi ve üye işlemi olmayan Interface i (marker) işaretleyici olarak tanımlarız.

**S58: Object klonlaması nedir?**

A58: Bu açık olarak bir objenin kopyasının yapılması prosedürüne denir. Fakat klonlanmış objenin methodu Protected olur, bu yüzden bu methodun override edilmesi gerekir.

**S59: Constructor Overloading (yapıcı aşırı yükleme) nedir?**

C59: Her biri farklı olan bir Class’a herhangi bir sayıda ekleme tekniğidir.

**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.

**S62: Checked ve unchecked exceptions arasındaki fark nedir?**

A62: **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.)

**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)*

**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.

**S65: Java da exception hiyerarşik yapısı nasıldır?**

C-65:

**OBJECT==>THROWABLE===>**

A - ) EXCEPTION

A1-)CHECKED EXCEPTION

A2-)UNCHECKED EXCEPTION

B - ) ERROR

B1-)VirtualMachineError

B2-)AssertionError

**S66: Özel bir istisna exception nasıl yapılır?**

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

**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

**S68: Finally block nedir?**

C68: Her zaman bir dizi ifadeyi yürüten bir bloktur.

Her zaman bir try bloğu ile ilişkilidir.

**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?**

C71: Evet uygulayabiliriz. Ancak yaklaşım özelden genele doğru olmalıdır.

**S72: “OutOfMemoryError” (bellek hatası) nedir?**

C72: OutOfMemoryError, java-lang'ın alt klasıdır. Genellikle JVM'mizin belleği dolduğunda oluşan hata.

**S73: Java exception class in en onemli methodlari nelerdir?**

C73:

1. String getMessage()

2. public StackTraceElement[] getStackTrace()

3. Synchronized Throwable getCause()

4. String toString()

5. void printStackTrace()

**S74: (İş parçacığı) Thread nedir?**

C74: 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.

**S75: İş parçacığı oluşturmanın yolları nelerdir?**

C75: Java'da iş parçacıkları aşağıdaki iki yolla oluşturulabilir;

Runnable ara yüzünü uygulayarak. Thread’i genişleterek.

**S76: Garbage collection nedir?**

C76: Java'da çöp toplama, örtük implicit olarak hafıza yönetimine yardımcı olan bir programdır. Java'da, yeni anahtar sözcüğü kullanarak dinamik olarak objeler oluşturabilirsiniz, herhangi bir kez oluşturulduktan sonra biraz hafıza tüketir.

**S77: Java da kaç farklı garbage collector çeşidi vardır?**

C77: 4 cesit;

*Serial Garbage Collector*

*Parallel Garbage Collector*

*CMS Garbage Collector*

*G1 Garbage Collector*

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

C78: Hayır onları keyword olarak kullanamayız.

**S79: JVM tarafından kaç tür bellek alanı bulunur?**

C79: 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?**

C80: Yerel değişkenler (local variables), herhangi bir varsayılan değerle (default value) başlatılmaz.