Q1: Why **java** platform is **independent**?

A1: Because of its bytecodes which can run on any system.

S1: Java platformu neden **bağımsızdır**?

C1: Herhangi bir sistemde çalışabilen bayt kodlarından dolayı.

Q2: Why java is **not 100% object oriented**?

A2: Because it makes use of 8 primitive data types which are not object.

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

C2: Çünkü nesne olmayan 8 primitive data türünü kullanır.

Q3: Explain **public static void main (String [] args)** in Java. A3: This is the **entry point** for any Java program.

**public**: it is an access modifier which is used to specify who can access this method. it means this method will be accessible by any class.

**static**: it is a keyword in java which identifies it is class based. we use it for main () is made static in java so that it can be accessed without creating the instance of a class.

**void**: it is the return type of the method.

**main** (): it is the name of the method which is searched by JVM. JVM: java virtual machine.

**String args**: it is the parameter passed to the main method.

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.

Q4: What are the **wrapper classes** in java?

A4: Wrapper classes convert the java primitives into the reference types or object of that class.

S4: **Wrapper Class’lar** nelerdir?

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

Q5: What is **constructor** in Java?

A5: it refers to a block of code which is used to initialize an object. it must have the same name as the name of the class.

S5: Java da **Constructor** nedir?

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

Q6: How many types of the constructers are there in java? A6: There are **two types**;

1. **Default Constructor**: it doesn’t take any input.
2. **Parameterized Constructor**: it is capable of initializing the instance variables with the provided values.

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.

Q7: How can we make a class **singleton**? A7: By making its **constructer private**.

S7: Bir Class i nasıl singleton yapabiliriz?

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

Q8: What is the difference between **ArrayList** and **Vector** in Java?

A8: **ArrayList** is not synchronized so that it is fast but **vector** is synchronized so that it is slow as it is thread safe.

S8: Java da ArrayList ve Vector arasındaki fark nedir?

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

Q9: What is the **difference** between **equals ()** and **==** in Java?

A9: **equals ()** operator for using to compare primitives and objects.

**==**, it uses to compare two objects.

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.

Q10: What are the **differences** between **heap** and **stack memory** in java?

A10: **Heap memory** is used by all the parts of the application and objects stored in the heap are globally accessible. **Stack memory** is used only by one thread of execution and it cannot be accessed by others.

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.

Q11: What is a **package** in java?

A11: It is a collection of related classes.

S11: Java da package nedir?

C11: İlgili Class’ların toplandığı yerdir.

Q12: What are the **advantages of the packages**? A12: 1 avoiding name clashes.

1. easier access control on the code
2. it can also contain hidden classes which are not visible to the outer classes.

S12: Package’ların avantajları nelerdir? C12: 1. İsim çatışmasını önler.

1. Kodlara kolay ulaşım sağlar.
2. Gizli Class’ları barındırır ki Class dışından görülemesin.

Q13: Why **pointers** are not use in Java?

A13: Because they are unsafe and increases the complexity of the program.

S13: Neden pointer lar Java da kullanılmazlar?

C13: Çünkü güvenli değillerdir ve program içindeki karmaşıklığı artırırlar.

Q14: What is **JIT compiler** in java?

A14: Just in Time compiler is a program that helps in converting the java bytecodes into the instructions.

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

C14: JIT derleyici yani tam zamanlı derleyici, Java bayt kodlarını talimatlara dönüştürmeye yardımcı olan bir programdır.

Q15: What are the **access modifiers** in java? A15: Default, Private, Public and Protected.

Q16: Explain the access modifiers in java?

A16: **Default**: accessible same packages and same class **Public**: accessible any packages and any class **Private**: accessible only same class

**Protected**: not accessible only different package non-subclass.

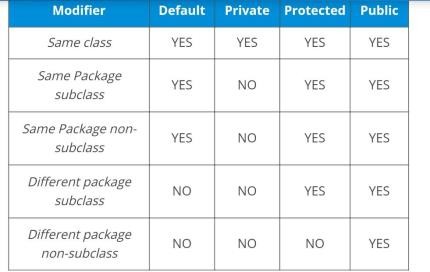
S15: Java da access modifier lar nelerdir? C15: Default, Private, Public ve Protected.

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.)



\*Tablo burada daha net anlaşılabilir. same => ayni;

different => farklı;

subclass - alt class => alt kumesi gibi;

non-subclass => alt class i olmayan demek.

Q17: Define a **class** in Java?

A17: A class is our working area which includes all our data. It contains fields variables and methods.

S17: Java da class i tarif ediniz?

C17: Class, datalarımızı barındıran içinde field variable’ları ve method’ların olduğu

bizim çalışma alanımızdır.

Q18: What is an **Object** in java and how is it created?

A18: Object is an instance of the class having instance variables. An object is created using by the **new** keyword in java.

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.

Q19: What is **OOP** (**Object Oriented Programming**)?

A19: It is a programming model or approach is ideal for the programs large and complex codes and needs to be actively updated and maintained. The programs are organized around objects rather than logic and functions.

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.

Q20: What are the main concepts of OOPs in JAVA?

A20: **Inheritance**, **Encapsulation**, **Abstraction**, **Polymorphism**.

Q21: Explain them? A21:

**Inheritance**: it is a process where once class acquires the properties of another.

**Encapsulation**: it is a mechanism of wrapping up data.

**Abstraction**: it is a methodology of hidding implementation details from the user.

**Polymorphism**: it is the ability of a variable to take multiple forms.

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.

Q22: What is the difference between **local variables** and **instance variables**?

A22: A **local variables**: It is a typically used inside a method, constructor or a block and it has only local scope. An **instance variable**: It is a variable which bounded to its object itself. They are declared within a class but outside of the method.

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.

Q23: What are the differences between **constructors** and **methods** in java? A23:

1. Constructors: it is used to initialized the state of an object. Methods: it is used to represent the behavior of an object.
2. Constructors: it does not have any return type. Methods: it must have a return type.
3. Constructors: it is invoked implicitly Methods: It needs to be invoked explicitly
4. Constructors: it must be same name as the name of class. Methods: it may be or not.

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.

Q24: What is the **final keyword** in java?

A24: Final is a special keyword in java that is used as a **non-access modifier**.

Q25: Explain using **final keyword**?

A25: When the final keyword is used with a variable then its value cannot be changed once assigned.

**final method**: it cannot be overridden.

**final class**: it cannot be extends.

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.

Q26: What are the differences between **break** and **continue statements**? A26:

1. Break can be used in switch and loops. Continue is just used in loops.
2. Break terminates the moments it is executed.

Continue it does not terminate just jumps to the next step.

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.

Q27: What is an **infinite loop** in java?

A27: It is an instruction sequence in java that loops endlessly when a functional exist is not met.

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

C27: Java’da bir işlevsellik karşılanmadığında, bir talimat dizisi tarafından sonsuz

döngüye girilir.

Q28: What is the difference between **this()** and **super()** keywords in java?

A28: This () is used to call the default constructors of the same class but super () used to call them of the parent/base class.

Note: This () and super () keywords must be the first line of a block.

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.

Q29: What is a Java **String Pool**?

A29: It refers a collection of Strings which are stored in heap memory. Whenever a new object is created, it first checks the object is already in the pool or not.

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.

Q30: What are the differences between **static** and **non-static methods**? A30:

1. **Static Methods:** The static keyword must be used before the method name but

**Non-static Methods:** no need to use static the keyword.

1. **Static Methods:** it is called using the class but **non-static Methods:** it is can be called like general methods.
2. **Static Methods:** it cannot be access non static variables but **non-static Methods:** it can access them.

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.

1. Static method: Class kullanılarak çağrılır

Ancak non-static methodlar genel methodlar gibi çağrılabilir.

1. Static method: Statik olmayan değişkenlere (variables) erişilemez

Ancak non-static method bunlara erişebilir.

Q31: What are the differences between **Strings** and **StringBuilder**? A31: 1. **storage area;** Strings in StringPool

StringBuilder in Heap area

* 1. **Thread safe;** Strings yes

StringBuilder no

* 1. **form;** Strings immutable StringBuilder mutable
  2. **Performance** Strings Fast

StringBuilder more efficient, especially setter and getter methods.

S31. **String**’ler ve **StringBuilder**’lar arasındaki farklar? A31: 1. **saklama alanı;** String’ler StringPool'da

StringBuilder Heap hafızada

1. **İş parçacığı güvenliği;** String’ler evet

StringBuilder no

1. **form;** String’ler Değişmez

StringBuilder değiştirilebilir

1. **Performans**; String’ler Hızlı

StringBuilder daha verimli, özellikle setter ve getter

methodlarinda.

Q32: Is constructor **inherited**? A32: No, it is not inherited.

S32: Constructor’lar inherited edilebilir mi? C32: Hayır edilemez.

Q33: What is a **ClassLoader** in java?

A33: It is a subset of JVM that is responsible for loading the class file.

Q34: How many ClassLoader java **provides** and what are they? A34: Java provides **3 ClassLoader**.

1. Bootstrap
2. Extension
3. System/ Application.

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

Q35: Why java Strings are **immutable** in nature?

A35: Simply means once String object is created its state cannot be modified. It enhances security caching, synchronization and performance of the 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.

Q36: What is the difference between **Array** and **ArrayList**? A:36

1. Array cannot contain values of different data types. ArrayList can contain them.
2. Array's size must be defined.

Size of ArrayList can be dynamically changed.

1. Array need to specify the index in order to add data. ArrayList does not need that.
2. Arrays can contain Primitive data and objects. ArrayList can contain only objects.

S36: Array ve ArrayList arasındaki farklar nelerdir? C36:

1. Array, farklı veri türlerinin değerlerini içeremez.

ArrayList bunları içerebilir.

1. Array size’ı tanımlanmalıdır.

ArrayList size’ı dinamik olarak değiştirilebilir.

1. Veri eklemek için Array’in indeksi belirtmesi gerekir. ArrayList buna ihtiyacı yok.
2. Array’ler primitive data ve nesneler (Objects) içerebilir.

ArrayList yalnızca objeleri içerebilir.

Q37: What is a **Map** in java?

A37: Map is an interface of Util Package which maps unique keys to values.

Does not contain duplicate keys

Each key can map at maximum one value.

Q38: What is **Collection class** in Java?

A38: The collection is a framework that acts as an architecture for storing and manipulating a group of objects.

Q39: Explain about the **collection classes** what they include? A39: Includes: **interfaces**, **classes**, **methods**.

**List**, **queue** and **set** important part of collections.

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.

Q40: What is **Polymorphism**?

A40: A best example of this question is a mobile phone which is used like a camera or a calculate or a mp3 player or a remote.

Q41: How many types of **polymorphism** are there in java? A41: There are two types;

**Compile Time polymorphism** is method **overloading**.

**Run Time polymorphism** is done using **inheritance** and **interface**.

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.

Q42: What is the **abstraction** in java?

A42: It basically deals with hiding the details and showing the essential things to the user. it can be two ways abstraction classes and interfaces.

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)

Q43: What do you mean **interface** in Java?

A43: It is a collection of abstract methods and static constants. It does not have any constructions. It is a group of related methods with empty bodies.

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.

Q44: What are the differences between **abstract classes** and **interfaces**? A44: 1. Abstract Classes: it can provide that have to be overridden.

Interfaces: it cannot provide any code at all just signature.

1. Abstract Classes: a class may extend only one abstract class Interfaces: a class may implement several interfaces
2. Abstract Classes: it can have non-abstract methods Interfaces: All methods of interface are abstract
3. Abstract Classes: it can have instance variable Interfaces: it cannot have instance variables
4. Abstract Classes: it can have any visibility public protected private Interfaces: it must be public or none
5. Abstract Classes: it can contain constructions Interfaces: it cannot contain
6. Abstract Classes: fast

Interfaces: slow need to extra time to find corresponding method to the actual

class.

S44: Abstract Class ile Interface arasındaki farklar nelerdir? (**Bu soru olmazsa olmazlardan**) A44: 1. Abstract Class: Overridden yapılmasını sağlayabilir.

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

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

Interface: bir class birkaç Interface uygulayabilir

1. Abstract Class: soyut olmayan method sahip olabilir

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

1. Abstract Class: örnek değişkeni (instance variable) olabilir

Interface: örnek değişkenlere (instance variables) sahip olamaz

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

private

Interface: sadece public olmalı ya da olmamalı

1. Abstract Class: Constructor içerebilir

Interface: Constructor içeremez

1. Abstract Class: hızlı

Interface: yavaş , gerçek class a uygun methodu bulmak için ekstra zamana

ihtiyacı var.

Q45: What is **inheritance** in java?

A45: it is a concept and helps to reuse the code.

it establishes a relationship among different classes. their names are parent class and child class

Q46: What is **child class** in java?

A46: A class which inherits the properties is known child class. Q47: What are the different types of inheritance?

A47: **Single**, **multilevel**, **hierarchical** and **hybrid**. Q48: Explain them?

A48: **single**: one parent one child

**multilevel**: more parents one child

**hierarchical**: one parent more child classes

**hybrid**: it is a combination of two or more types of inheritances.

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.

Q49: What is method **overloading** in java?

A49: It is to add or extend more to the method's behavior. It is a compile time polymorphism.

Methods must have different signature.

Methods of the same class shares the same name but each method must have a different number of parameters.

Or parameters having different types and order. It may or may not need inheritance in class.

Q50: What is method **overriding** in java?

A50: It is to "change" existing behavior of the method. It is a run time polymorphism.

The methods must have the same signature.

The sub-class has the same method with the same name.

Exactly the same number and type of parameters and same return type as a super class.

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.

Q51: Can you override a private or static method in java?

A51: No, you cannot because it is not accessible there and you need to do inheritance in class.

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.

Q52: Is **multiple inheritance** supported by java?

A52: Java doesn’t support multilevel inheritance. Because it is difficult for the compiler to decide. Which method to execute. If the same name of method in different parent classes.

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)

Q53: What is **encapsulation** in java?

A53: The data is hidden from the outer world and can be accessed only via current class methods. Providing public methods to modify and view the values of the variables.

A53: 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

Q54: What is an **association** in java?

A54: It is a relationship where all objects have their own lifecycle and there is no owner. These relationships can be one to one, one to many, many to one and many to many.

Q55: What do you mean by **aggregation** in java?

A55: It is a specialized form of association where all objects have their own lifecycle but there is an ownership and child object cannot belong to another parent object.

Q56: What is a **composition** in java?

A56: It is again a specialized form of aggregation and we can call this as a death relationship.

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.

Q57: What is a **marker interface**?

A57: It can be defined as the interface having no data member and member functions.

S57: İşaretleyici Interface nedir?

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

Q58: What is **object cloning**?

A58: It is the process of creating an exact copy of an object. But object clone is a protected method, thus you need to override it.

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.

Q59: What is a **constructor overloading**?

A59: It is a technique of adding any number of constructors to a class each having a different parameter list.

S59: Constructor Overloading (yapıcı aşırı yükleme) nedir?

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

Q60: How can you **handle** java **exceptions**?

A60: There are five keywords. **Try, catch, finally, throw** and **throws** using them. Q61: What is difference between **error** and **exception**?

A61: **Error**: it is an irrecoverable condition occurring at the run time.

**Exception**: it is condition that occur because of bad input or human error etc. in most of the cases it is possible to recover it.

Q62: What are the differences between **checked** and **unchecked exceptions**? A62: Checked Exceptions: They are checked at compile time.

Example; IO exception SQL exception.

Unchecked Exceptions: they are not checked at compile time. Example;

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.)

Q63: What purpose do the keywords **final**, **finally** and **finalize**? A63:

**Final**: It is used to apply restrictions on class method and variable Final class cannot be inheritance

Final method cannot be overridden Final variable cannot be changed

**Finally**: It is used to place important code; it will be executed.

Whether exception is handled or not.

**Finalize**: It is used to perform clean up processing just before The object is garbage collector.

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)

Q64: What are the differences between **throw** and **throws**? A64:

1. **throw** is used to explicitly throw an exception,

**throws** is used to declare an exception.

1. **throw** is followed by an instance

**throws** is followed by class

1. **throw** is used within method

**throws** is used within the method signature

1. we cannot **throw** multiple exceptions

we can **throws** declare multiple exceptions.

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

1. **throw** ardından bir örnek gelir

**throws** ardından class gelir

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

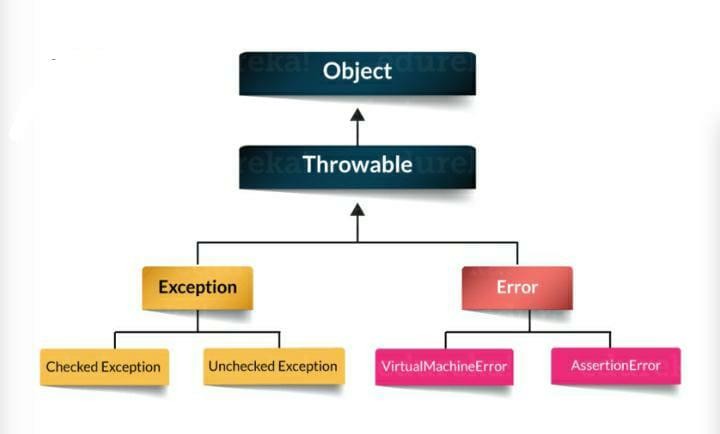
**throws**, method un imzası içinde kullanılır

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

**throws** ile birden çok exception bildirebiliriz.

Q65: What is **exception hierarchy** in java?

S65: Java da exception hiyerarşik yapısı nasıldır?



Q66: How to create a **custom exception**?

A66: To create your own exception extend the exception class or any of its subclasses.

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.

Q67: What are the differences between **processes** and **threads**? A67:

* 1. Run in

process: separate memory spaces Threads: shared memory spaces

* 1. Control

process: by operating system threads: a programmer in a program

* 1. Forms

process: independent threads: dependent

S67: **Process** ve **threads** arasındaki farkları açıklayınız? C67:

1. Çalışma:

Süreç (prosess): ayrı bellek alanları

Konular, diziler (threads): paylaşılan bellek alanları

1. Kontrol

Process: işletim sistemine göre

Threads: bir programdaki programcı tarafından

1. Biçimler

processs: bağımsız

threads: bağımlı

Q68: What is a **finally block**?

A68: It is a block which always executes a set of statements.

It is always associated with a try block.

Q69: Is there a case when **finally**, will **not execute**? A69: Yes, there is. if the program exits by calling.

System.exit() or causing a fatal error.

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.

Q70: what is **synchronization**? A:70:

-A synchronized block of code can be executed by only one thread at a time.

-As Java supports execution of multiple threads, two or more threads may access the same fields or objects.

-Synchronization is a process which keeps all concurrent threads in execution to be in sync.

-Synchronization avoids memory consistency errors caused due to inconsistent view of shared memory.

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)

Q71: Can we have **multiple catch blocks** under **single try block**?

A71: Yes, we can have multiple catch blocks under single try block but the approach should be from specific to general.

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

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

Q72: What is “**Out of Memory Error**”?

A72: “Out of Memory Error” is the subclass of java-lang. Error which generally occurs when our JVM runs out of memory.

S72: “OutOfMemoryError” (bellek hatası) nedir?

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

Q73: What are the **important methods** of **Java Exception Class**?

A73:

1. String getMessage()
2. public StackTraceElement[] getStackTrace()
3. Synchronized Throwable getCause()
4. String toString()
5. void printStackTrace()

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()

Q74: What is a **Thread**?

A74: A thread is the smallest piece of programmed instructions which can be executed independently by a scheduler.

In Java, all the programs will have at least one thread, which is known as the main

thread.

Q75: What are the ways to create a **thread**?

A75: In Java, threads can be created in the following two ways; By implementing the Runnable interface.

By extending the Thread.

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.

Q76: What is the **garbage collection**?

A76: Garbage collection in Java a program which helps in implicit memory management. Since in Java, using the new keyword you can create objects dynamically, which once created will consume some memory.

Q77: What are the different types of **garbage collectors** in Java? A77: 4 types;

**Serial** Garbage Collector **Parallel** Garbage Collector **CMS** Garbage Collector **G1** Garbage Collector

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

Q78: Is **delete next, main exit** or **null** keyword in java? A78: No, we don’t use them as a keyword in java.

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

C78: Hayır onları keyword olarak kullanamayız.

Q79: How many types of **memory areas** are allocated by JVM?

A79:

* 1. Class(method) area
  2. Heap
  3. Stack
  4. Program counter register
  5. Native Method Stack

S79: JVM tarafından kaç tür bellek alanı bulunur?

C79: 5 tanedir;

1. Class(method) area
2. Heap
3. Stack
4. Program counter register
5. Native Method Stack

Q80: What is the **default value** of the **local variables**?

A80: The local variables are not initialized to any default value.

S80: Yerel değişkenler (local variables)’in varsayılan değeri (default value) nedir? C80: Yerel değişkenler (local variables), herhangi bir varsayılan değerle (default

value) başlatılmaz.