### **1. Java programlama dilinin özellikleri nelerdir?**

* **Nesne yönelimli:** Java, nesne-yönelimli bir programlama dilidir.
* **Platformdan bağımsız:** Java, bir kez yazılır ve farklı platformda çalışabilir. Birçok farklı işletim sistemini destekler.
* **Otomatik bellek yönetimi:** Java, bellek yönetimini otomatik olarak gerçekleştirerek, programcının bellek yönetimiyle ilgilenmesini ortadan kaldırır.
* **Güvenli:** Java, oldukça güvenlidir. Kodun güvenliğini sağlar.
* **Açık kaynak:** Java, açık kaynak bir programlama dilidir. İsteyen herkes Java’yı kullanabilir. Kullanıcıların kodlarını değiştirmesine veya kullanmasına izin verir.
* **Multithreading:** Java, birden fazla iş parçacığını (thread) destekler ve aynı anda birden fazla işlemi gerçekleştirmek için kullanır.
* **Dinamik:** Java, dinamik olarak bağlantı kurar.
* **Geniş kapsamlı:** Java, geniş kapsamlı bir programlama dilidir. Birçok farklı alanda kullanılabilir ve farklı türde yazılım geliştirmek için kullanılır.
* **Büyük kullanıcı topluluğu:** Java’nın geniş bir kullanıcı ekosistemi bulunur. Ayrıca, büyük kullanıcı topluluğu vardır. Bu topluluk ile Java geliştiricileri birbirlerinden yardım alabilir.

### **2. Java Sanal Makinesi nedir?**

✔️ **Java Sanal Makinesi (JVM)**, Java programlarının çalıştırılması için gerekli olan bir yazılımdır. Bytecode adı verilen kodu yorumlar ve istenen çıktıyı üretir.

✔️ Java kodunun **işletim sistemi** tarafından anlaşılmasını sağlar.

✔️ JVM, aynı zamanda **bellek yönetimini**, güvenliği ve **hata ayıklamayı** gerçekleştirir.

### **3. JVM, JRE ve JDK arasındaki farklar nelerdir?**

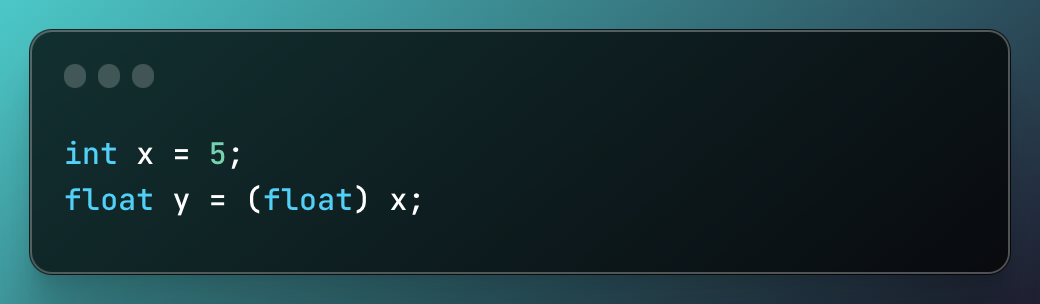
JVM, JRE ve JDK, Java’nın üç temel kavramıdır. JVM Java kodlarını çalıştırır, JRE Java programlarını çalıştırmak için gerekli olan ortamı sağlar ve JDK Java programlarının geliştirilmesi için gerekli olan araçları sağlar. Daha detaylı bakacak olursak:

* **🎯 JVM** tam açılımıyla Java Virtual Machine, Java kodunu platformdan bağımsız hale getirir ve tüm işletim sistemlerinde ve platformlarda çalışmasını sağlar.
* **🎯 JRE** tam açılımıyla Java Runtime Environment, Java programlarının çalıştırılması için gerekli olan bir yazılımdır. JRE, JVM dahil olmak üzere Java ile ilgili gerekli tüm dosyaları içerir.
* **🎯 JDK** tam açılımıyla Java Development Kit, Java programlarının geliştirilmesi için gerekli olan bir yazılımdır. JDK, Java programlarının geliştirilmesi için gereken tüm araçları, kütüphaneleri, derleyicileri ve debugglerları içerir.

### **4. Typecasting'i açıklayın ve kod örneği gösterin.**

Typecasting, bir veri tipindeki değişkeni başka bir veri tipindeki değişkene atama/dönüştürme işlemine denir. Örtülü ve açık olmak üzere iki tip veri türü vardır.

Typecasting işlemi, genellikle programcı tarafından yapılır. Örneğin, Java'da, bir int değeri bir float değerine dönüştürmek için (int) ifadesi kullanılır.

Kod şöyle gösterilir 👇:  


### **5. Nesne nedir?**

Bir sınıf tanımlandıktan sonra yeni bir veri tipi oluşturulmuş olunur. Java’da oluşturulan bu yeni **veri tipine ait değişkene** nesne (object) adı verilir.

Bir Java nesnesinin iki önemli özelliği davranış ve durumdur. JVM yeni anahtar sözcükle karşılaştığı anda bir nesne oluşturur.

**6. Static metod, variable, code bloğu nedir**

### **7. Java constructors nedir?**

**Java constructors** bir diğer adıyla **yapıcı**, nesneleri başlatmak için kullanılan kod bloğunu ifade eder. Constructors, sınıfın adı ile aynı olmalıdır. Bir nesne veya sınıf oluşturulduğunda otomatik olarak oluşturulur. "**new**" anahtar kelimesiyle birlikte çalışır.

Bir constructors tanımlarken şu kurallara dikkat etmek önemlidir:

* ✅ Oluşturucunun adı sınıfın adı ile aynı olmalıdır.
* ✅ Oluşturucu bir dönüş türüne sahip olmamalıdır.
* ✅ Eğer sınıfta farklı oluşturucu tanımlanıyorsa, parametrelerin sayısı ve tipi farklı olmalıdır.

### 

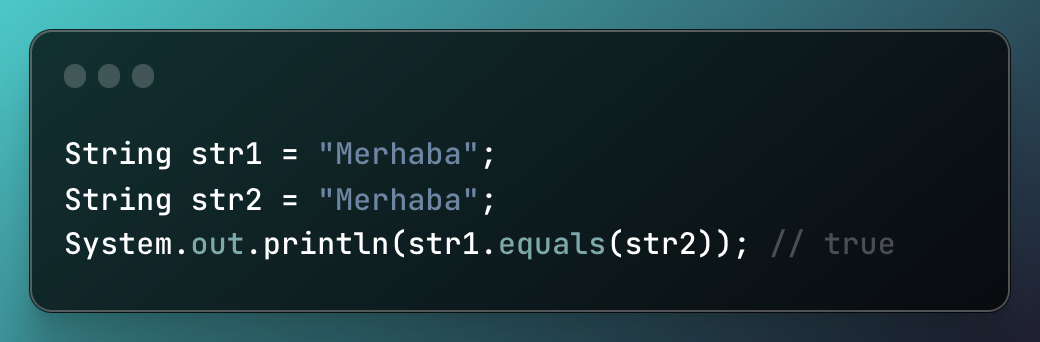
### 

### **8. Java'da equals() ve == nedir? Aralarındaki fark nedir?**

Java’da nesnelerin eşitliklerini karşılaştırmak için **equals()** ve **== operatörleri** kullanılır.

**1️⃣ Equals() operatörü**, iş mantığı tarafından tanımlanan iki nesnenin eşitliğini kontrol etmek için kullanılır. Equals() operatorü, Object sınıfında tanımlanır.

**String** sınıfı ve bazı diğer sınıflar, içerik karşılaştırması için "equals()" metodunu geçersiz kılar.



2️⃣ Java'daki varsayılan uygulama, iki nesneyi karşılaştırmak için **== operatörünü** de kullanabilir. "==" operatörü, nesnelerin referanslarını (bellek adreslerini), iki değişkenin aynı bellek bölgesini işaret edip etmediğini kontrol etmeye yarayan operatördür.

### **9. Java süper anahtar kelime nedir? Ne zaman kullanılır?**

Java'da süper anahtar kelime, doğrudan bir üst sınıf nesnesine atıfta bulunan bir **referans değişkenidir**. Bir alt sınıf örneği oluşturduğunuzda, süper anahtar kelimesi tarafından başvurulan üst sınıfın bir örneğini de yaratmış olursunuz.

Java **süper anahtar** kelimesinin kullanımları şunlardır:

1️⃣Bir üst sınıf örnek değişkenine atıfta bulunmak istendiğinde kullanılır.

2️⃣ Süper anahtar kelimesi, bir ana sınıfın yöntemini çağırmak için kullanılır.

3️⃣ Super(), ana sınıfın yapıcısını çağırmak için kullanılır.



**Çıktı:**

Black

White

### **10. Java’da HashSet, TreeSet ve HashLinkedSet nedir?**

HashSet, LinkedHashSet ve TreeSet, Java’da Set (küme) türleridir.

1️⃣ **HashSet**, set arabirimini uygulayan bir sınıftır ve elemanları tekrar etmeyen bir yapıda saklar. HashSet arayüzü verilerini düzensiz olarak sıralar. Performans olarak tasarruf etmenizi sağlar.

**2️⃣ HashLinkedSet** ise verileri eklenen sıraya göre düzenli tutar. HashSet'in bir alt sınıfıdır. Eğer verilerin sırası önemli değilse, HashLinkedSet ekstra maliyetli olacaktır.

**3️⃣ TreeSet**, NavigableSet arabirimini uygulayan bir sınıftır ve elemanları sıralı bir şekilde saklar. Bu sınıf, elemanları kısmi sıralama veya doğal sıralama düzenine göre düzenler.

### **11. HashSet ve TreeSet arasındaki fark nedir?**

| **HashSet** | **TreeSet** |
| --- | --- |
| Bir hash tablosu aracılığıyla uygulanır. | TreeSet, verileri depolamak için ağaçları kullanan SortedSet arayüzünü kullanır. |
| Boş nesneye izin verir. | Boş nesneye izin vermez. |
| Arama, ekleme ve silme işlemleri için TreeSet'ten daha hızlıdır. | Arama, ekleme ve silme için HashSet'ten daha yavaştır. |
| Öğeleri düzenli bir şekilde tutmaz. | Öğeleri sıralı bir şekilde tutar. |

### **12. HashSet, LinkedHashSet ve TreeSet ne zaman kullanılır?**

* **💻 HashSet'i kullanın:** Öğeleri herhangi bir sıraya göre tutmaya gerek olmadığında.
* **💻 LinkedHashSet'i kullanın:** Benzersiz öğeler grubuna ihtiyaç duyulduğunda ve öğelerin ekleme sırası da önemli olduğunda.
* **💻 TreeSet kullanın:** Benzersiz öğeler/nesneler grubu gerektiğinde ve bazı comparator’lere göre öğelerin sıralanması gerektiğinde.

### **13. Java’da Break nedir?**

Java'da break ifadesi, döngü içerisinde koşul sağlanmadan döngüyü bitirmek, **tekrarlı yapıları kontrol etmek** için kullanılan bir ifadedir.

“for, while, do while” gibi **witch** ve **loop** ifadelerinde kullanılır. Çalıştırıldığı anda switch ve loop ifadelerinin sonlanmasını sağlar.

### **14. Java’da Continue nedir?**

Java'da "continue" ifadesi, döngülerde veya switch-case yapılarında kullanılarak, belirli bir durumu atlayarak döngünün veya yapı bloklarının devamını sağlayan bir **kontrol ifadesidir.** Yalnızca döngü ifadeleriyle kullanılabilir.

### **15. Java’da OOP (Nesne Yönelimli Programlama) Nedir?**

Nesne yönelimli programlama veya OOP, programların mantık ve işlevler yerine nesneler etrafında düzenlendiği bir programlama modeli veya yaklaşımıdır.

**OOP**, mantık yerine nesnelere odaklanır. Bu yaklaşım, büyük ve karmaşık kodların daha düzenli ve kolay anlaşılır bir şekilde geliştirilmesini sağlar .

### **16. Java'daki OOP'lerin prensipleri nelerdir?**

1. **Kalıtım:** Kalıtım, bir sınıfın diğerinin özelliklerini almasıdır.
2. **Kapsülleme:** Java'da kapsülleme, verileri ve kodu tek bir birim olarak bir araya getirmektir.
3. **Soyutlama:** Soyutlama, uygulama ayrıntılarını kullanıcıdan gizleme ve yalnızca kullanıcılara işlevsellik sağlama metodolojisidir.
4. **Polimorfizm:** Polimorfizm, bir değişkenin, işlevin veya nesnenin birden çok biçim alma yeteneğidir.

### **17. Java erişim belirleyiciler nedir?**

Erişim belirleyiciler, sınıfların diğer farklı sınıflardaki metotlara erişim sağlamasını veya bir sınıftaki özel bir metoda başka bir sınıfın erişim sağlamasını engellemesini sağlar.

Java’da 4 farklı erişim belirleyici bulunur:

**1️⃣ public:** Herhangi bir yerden erişim sağlanabilir.

2️⃣ **protected:** Sınıf, paket ve altsınıflar tarafından erişim sağlayabilir.

**3️⃣ private:** Sadece sınıf içerisinden erişilebilir.

**4️⃣ default(erişim belirleyici yok):** Bir erişim belirleyici kullanmadan sınıf ve paket içerisinden erişilebilir.

### **18. Java’da this() ve super() nedir? Aralarındaki farklar nedir?**

Bunlar, sınıf yapıcıları arasında geçiş yapmak için kullanılır. Her iki metod da yapıcı metotların içinde ilk satır olarak yazılması gerekmektedir. Aksi halde hata verecektir.

**👉 "this()":** Bu metod, sınıfın kendisinde bulunan yapıcı metotlar arasında geçiş yapmak ve metodları çağırmak için kullanılabilir.

**👉 "super()":** Bir sınıfın kalıtım yoluyla erişebildiği üst sınıfın yapıcı metotları arasında geçiş yapmak ve çağırmak için kullanılabilir.

### **19. Overloading ve override nedir?**

**👉 Overloading:** Java'da, aynı sınıf içinde aynı isimli birden fazla metodun olmasına denir. Bu metodlar, aynı isimlerine rağmen farklı parametreler alırlar. Böylece, kod okunaklılığı arttırılabilir.

**👉 Overriding:** Java'da, bir sınıfın kalıtım yoluyla erişebildiği üst sınıftaki aynı isimli metodun yeniden tanımlanmasına denir. Böylece, üst sınıftaki metodun işlevselliği değiştirilebilir ve kod tekrarı önlenebilir.

# **20. Java’da Veri Tipleri Nedir ve Türleri Nelerdir?**

Java’da veri tipleri iki ana kategoriye ayrılır: ilkel (primitive) veri tipleri ve referans (reference) veri tipleri. İlkel veri tipleri arasında int, char, boolean gibi türler bulunur. Referans veri tipleri ise sınıflar, diziler ve arayüzler gibi nesneleri temsil eder.

**Örnek Yanıt:** “Java’da ilkel veri tipleri doğrudan değeri saklar ve sabit bellek alanları kullanır. Referans veri tipleri ise nesneleri işaret eder ve dinamik bellek yönetimi sağlar.”

**21. Java’da ‘final’ Anahtar Kelimesi Ne İşe Yarar?**

final anahtar kelimesi, değiştirilemezliği ifade eder. final değişkenlerin değeri bir kez atandıktan sonra değiştirilemez, final metotlar override edilemez, final sınıflar ise türetilemez.

Örnek Yanıt: “final anahtar kelimesi, değişkenlerin, metotların ve sınıfların değiştirilemez olmasını sağlar. Bu, kodun güvenliğini ve tutarlılığını artırır."

**22. Java Koleksiyonları (Collections) Nedir?**

Java Koleksiyonları, veri kümelerini depolamak ve yönetmek için kullanılan bir yapı sağlar. List, Set, Map gibi çeşitli veri yapıları içerir.

Örnek Yanıt: “Koleksiyonlar, verileri gruplar halinde saklamak için kullanılan yapılar sunar. List, Set, ve Map gibi veri yapıları, veri yönetimini kolaylaştırır."

**23. Java’da İstisna Yönetimi (Exception Handling) Nedir?**

İstisna yönetimi, hataları yakalamak ve bu hatalarla başa çıkmak için kullanılan bir mekanizmadır. try-catch blokları ile uygulanır.

Örnek Yanıt: “İstisna yönetimi, çalışma sırasında oluşabilecek hataları yakalamak ve işlemek için kullanılan bir sistemdir. Bu, uygulamanın beklenmedik çökmesini önler.”

**24. Java’da Interface ve Abstract Sınıf Arasındaki Farklar Nelerdir?**

Interface, sadece soyut metot imzalarını içerir ve bir sınıf tarafından implement edilir. Abstract sınıf, hem soyut hem de somut metotlar içerebilir.

Örnek Yanıt: “Interface’ler sadece metot imzalarını içerir ve bir sınıf tarafından uygulanabilirken, abstract sınıflar hem soyut hem de somut metotlar içerir. Abstract sınıflar kalıtım için, interface’ler ise implementasyon için kullanılır.”

**25. Java’da Equals ve HashCode Metotları Arasındaki Fark Nedir?**

equals metodu, iki nesnenin mantıksal olarak eşit olup olmadığını belirler. hashCode metodu, nesnenin bir hash kodunu döndürür ve HashMap, HashSet gibi yapılarla çalışmak için kullanılır.

Örnek Yanıt: “equals metodu, iki nesnenin mantıksal olarak eşit olup olmadığını kontrol eder. hashCode metodu ise bu nesnelerin hash kodunu döndürür ve hash tabanlı koleksiyonlarla çalışmayı sağlar."

**26. Spring Boot’ta RESTful Web Servisleri Nasıl Geliştirilir?**

Spring Boot, RESTful web servisleri oluşturmak için kolay bir yapı sunar. @RestController anotasyonu ile HTTP isteğine yanıt veren metotlar tanımlanabilir.

Örnek Yanıt: “Spring Boot, @RestController anotasyonu kullanarak RESTful servisler geliştirmenizi sağlar. Bu anotasyon ile HTTP isteklerini karşılayan metotlar tanımlayabilirsiniz."

**27. Java’da Streams API Nedir?**

Streams API, Java 8 ile birlikte gelen ve veri işlemlerini fonksiyonel bir şekilde yapmanızı sağlayan bir araçtır. Akış (stream) üzerinden veriyi filtreleme, sıralama ve dönüştürme işlemleri yapılabilir.

Örnek Yanıt: “Streams API, Java 8 ile gelen ve veri işlemlerini fonksiyonel bir yaklaşımla yapmamızı sağlayan bir araçtır. Akışlar üzerinden veri üzerinde işlemler gerçekleştirebilirsiniz.”

**28. Java’da Garbage Collection Nasıl Çalışır?**

Garbage Collection, JVM’nin kullanılmayan nesneleri otomatik olarak bellekten temizlediği bir süreçtir. Bu işlem, bellek sızıntılarını önler ve bellek yönetimini kolaylaştırır.

Örnek Yanıt: “Garbage Collection, JVM’nin kullanılmayan nesneleri otomatik olarak bellekten temizlediği bir süreçtir. Bu, belleğin etkin bir şekilde kullanılmasını sağlar ve bellek sızıntılarını önler.”

**29. Java’da Optional Sınıfı Nedir?**

Optional, Java 8 ile gelen ve null kontrolü yaparken daha güvenli bir kod yazmanıza yardımcı olan bir sınıftır. Bir değerin mevcut olup olmadığını kontrol etmek için kullanılır.

Örnek Yanıt: “Optional, null değerleri güvenli bir şekilde kontrol etmenizi sağlar ve null pointer istisnalarını önlemeye yardımcı olur. Bu, kodunuzu daha güvenli ve okunabilir hale getirir.”

**30. @SpringBootApplication Anotasyonu Nedir?**

@SpringBootApplication, Spring Boot uygulaması için gereken üç temel anotasyonu bir araya getirir: @Configuration, @EnableAutoConfiguration, ve @ComponentScan.

Örnek Yanıt: “@SpringBootApplication, Spring Boot uygulaması için gerekli temel ayarları içeren birleştirilmiş bir anotasyondur ve uygulamanın çalışması için gereken yapılandırmaları sağlar."

**31. @GetMapping ile @PostMapping arasındaki fark?**

**38. @PathVariable ve @RequestParam Anotasyonları Nedir?**

@PathVariable, bir URL yolundaki değişkenleri metot parametrelerine bağlarken, @RequestParam HTTP isteğindeki parametreleri metot parametrelerine bağlar.

Örnek Yanıt: “@PathVariable, URL yolundaki değişkenleri metot parametrelerine bağlar. @RequestParam ise istek parametrelerini metot parametrelerine bağlar."

**39. @Component, @Service ve @Repository Anotasyonları Nedir?**

@Component, bir sınıfın Spring tarafından yönetilen bir bileşen olduğunu belirtir. @Service, iş mantığı içeren sınıflar için kullanılır. @Repository ise veri erişim katmanı için kullanılır.

Örnek Yanıt: “@Component, bir sınıfın Spring bileşeni olduğunu belirtir. @Service, iş mantığı sınıflarını belirtir ve @Repository, veri erişim sınıflarını belirtir."

**40. @Qualifier Anotasyonu Nedir?**

@Qualifier, birden fazla bean adayı olduğunda hangi bean'in enjekte edileceğini belirtir.

Örnek Yanıt: “@Qualifier, birden fazla uygun bean olduğunda belirli bir bean'i seçmek için kullanılır."

**41. @Primary Anotasyonu Nedir?**

@Primary, aynı türde birden fazla bean varsa, varsayılan olarak kullanılacak bean'i belirtir.

Örnek Yanıt: “@Primary, birden fazla uygun bean olduğunda varsayılan olarak kullanılacak olanı belirtir."

**42. @Scheduled Anotasyonu Nedir?**

@Scheduled, bir metodun belirli aralıklarla çalıştırılmasını sağlar.

Örnek Yanıt: “@Scheduled, bir metodun belirli zaman aralıklarında çalışmasını sağlar ve zamanlanmış görevleri yönetmek için kullanılır."

**43. Kafka nedir ve RabbitMQ nedir?**

**44. Lombok nedir. Dto nedir, Entity nedir?**

**45. NoArgs nedir AllArgs nedir RequiredArgs nedir?**

**Hocanın slayttan**

**46. AOP nedir?**

**47. Swagger nedir**

**48. Logging nedir nasıl yapılır ve neden yapılır.**

**49. Spring Boot Starter nedir**

**50. SOLID nedir**

**51. Heap ile Stack farkı nedir**

**52. Clean Code ile alakalı neler söyleyebilirsin**

**53. Methodları nasıl gizleriz**

**54. Mocklama nedir**

**55. unit testlerdeki code coverage nedir**

**56. @Query nedir**

**57. @JPQL nedir.**

**58. Design pattern -> singleton factory, factory, builder**

**59. DI nedir ve neden yapılır. IOC**

**60. Mikroservis nedir, neden kullanılır ve bileşenleri nelerdir.**

**61. Temel sql sorguları.**

**62. Veri tabanı optimizasyon.**

**63. GIT**

**64. Spring boot ile Spring farkı.**