

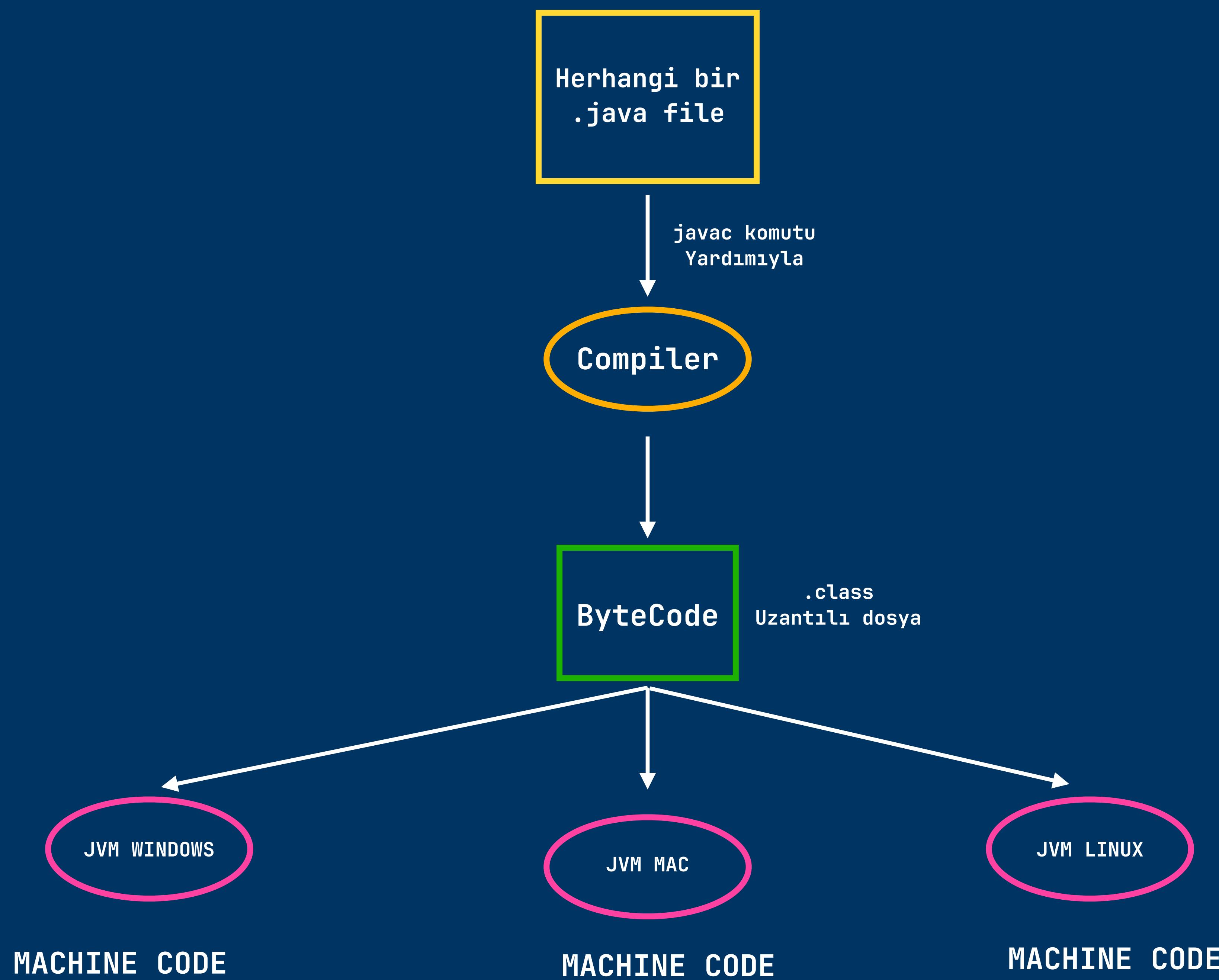
JAVA MÜLAKATLARI

JUNIOR ADAYLAR İÇİN

Furkan Şahin KULAKSIZ
Senior Software Developer

Onur AKTAŞ
Software Developer

JAVA NASIL ÇALIŞIR.?



JDK - JRE - JVM NEDİR.?

JVM - JAVA VIRTUAL MACHINE

JVM ByteCode'u çalıştırırmak için kullanılan bir sanal makine.
ByteCode'ları doğrudan bağımlı olduğu makine koduna çevirir.

JRE - JAVA RUNTIME ENVIRONMENT

JRE, JVM'nin yanı sıra Java uygulamalarının çalıştırılmasını sağlamak için gerekli olan core sınıfların, yardımcı dosyaların ve diğer kütüphanelerin bir araya getirilmiş Halidir.

JDK - JAVA DEVELOPMENT KIT

JDK, JRE'yi içinde barındıran, java uygulaması geliştirmek için gerekli olan bir yapıdır.

PRIMITIVE TYPES vs NON-PRIMITIVE TYPES

WRAPPER CLASSLAR

PRIMITIVE TYPES

byte	-	short	-	int	-	float	-	long	-	double	-	boolean	-	char
8 bit		16 bit		32 bit		32 bit		64 bit		64 bit		1 Bit		16 bit
-2^7		-2^{15}		-2^{31}		-2^{-149}		-2^{63}		-2^{-1074}		0		2^{16-1}
2^7-1		$2^{15}-1$		$2^{31}-1$		$(2-2^{-23}).2^{127}$		2^{63}		$(2-2^{-52}).2^{1023}$				
						Ondalıklı sayılar				Ondalıklı sayılar				

Primitive Type'lar stack'te saklanır.

Primitive Type'lara ulaşmak, Non-Primitive type'lara göre daha hızlıdır.

Primitive Type'lar null değerini alamazlar.

Primitive Type'ların değerleri sabittir.

NON-PRIMITIVE TYPES

Class - Arrays - String - Instancealar

Non-Primitive Type'ların Bellek Adresleri stack'te saklanır. İçerikleri Heap'de saklanır

Primitive Type'lara ulaşmak, Non-Primitive type'lara göre daha hızlıdır.

Non-Primitive Type'lar null değerini alabilirler.

Non-Primitive Type'ların bellekte tuttuğu alanlar değişkenlik gösterebilir.

Non-Primitive Type'ların kullanılmayanları Garbage Collector tarafından toplanır.

METHOD OVERLOADING - METHOD OVERRIDING

METHOD OVERLOADING

Aynı isimdeki methodların farklı parametrelerce kullanılması

METHOD OVERRIDING

Üst class'taki bir methodun, alt class'ta methodun dönüş tipini ve parametrelerini Değiştirmeden kendine göre değiştirmesi/ezmesi

JAVA

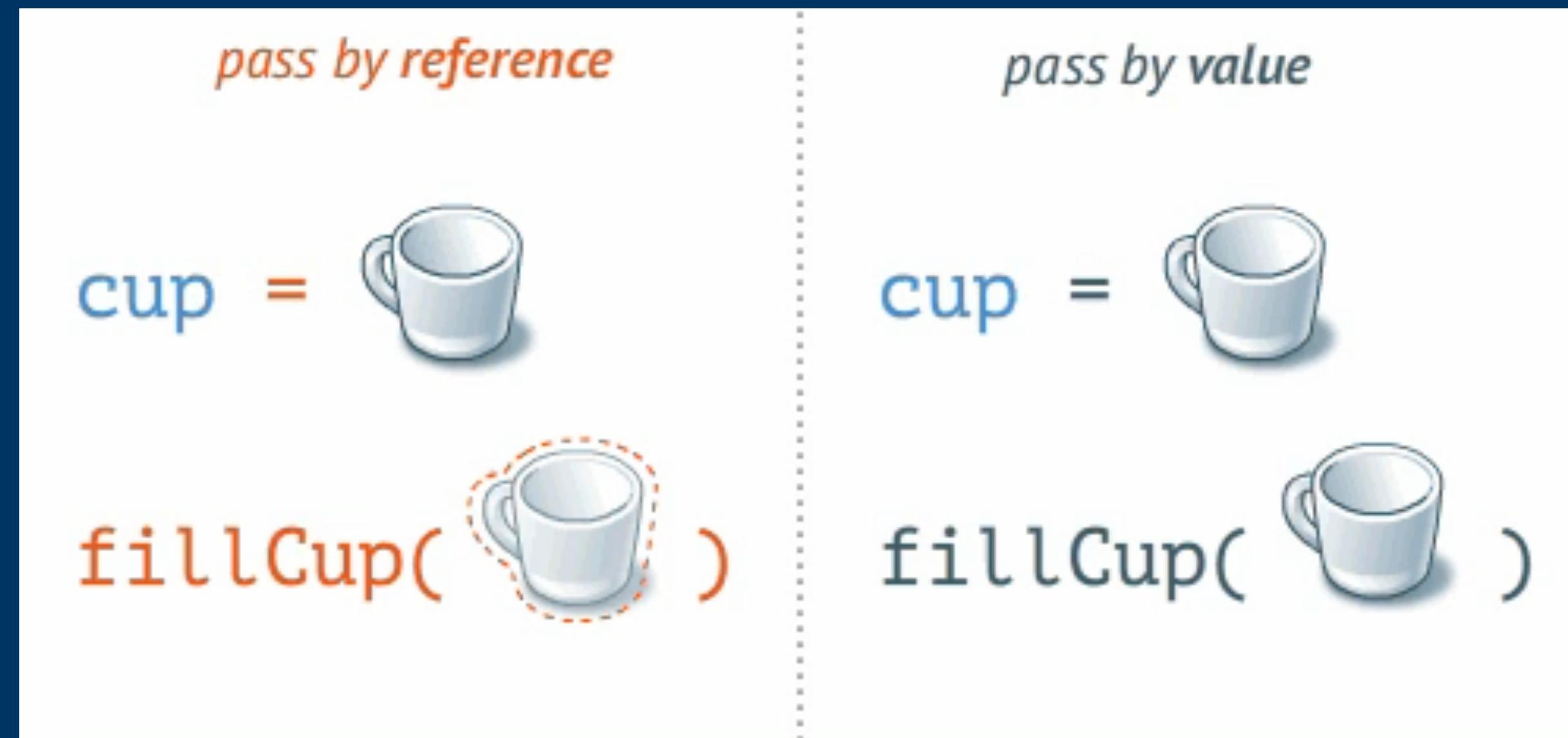
PASS-BY-VALUE

mudur

PASS-BY-REFERENCE

mıdır.?

Java dili pass-by-value'dur. Çünkü bir methoda, parametre olarak bir değişkenin ya da bir instance değişkeninin referansı parametre olarak verilmez. Kendisi parametre olarak verilir.



STATIC KEYWORDU

Static keywordü sınıfı ait değişkenler için kullanılır.

Yani static olan bir method'a ya da bir field'a Class adıyla doğrudan erişilebilir.

Nesne adıyla da ulaşılabilir ama best practice değildir.

Static methodlar override edilmezler.

JVM her ayağa kalktığında static değişkenler, tetiklensin ya da tetiklenmesin uygulama contextinin içerisine yüklenir.

JVM'in heap alanındaki perm gen/metaspace alanına yerleşir.

Class'lar static keywordü ile imlenemezler. Yani Class'lar static olmazlar.

FINAL KEYWORDU

Final olan bir şey finaldir. Yani değiştirilemez.

Final olan Class'lar extend edilemezler.

Final olan methodlar override edilemezler.

Final olan bir field için ya başlangıç değeri verilmeli, ya da constructorlarda ilk değer ataması yapılmalıdır.

`=` vs `equals()`

`=`, iki değişkenin değerlerini kontrol ederken, `equals()` iki değişkenin içeriklerine bakar.

`= primitive type'larda kullanılmalıdır.`

`equals()` non-primitive type'larda kullanılmalıdır.

Constructor Methodlar nedir.?

Nesne oluşturulduğunda çağrılan methodlara constructor methodları denir.

Genellikle publictir.

Class adıyla Constructor method adı aynı olmak zorunda.

Geriye değer döndürmezler.

Parametre alabilirler.

Bir Class'ta birden fazla constructor bulunabilir.

Bir Class oluşturduğumuzda constructor methodu tanımlamadıysak, arka tarafta Bir tane parametre almayan constructor bizim için oluşturulur.

ENCAPSULATION Nedir.?

Class içerisindeki methodların ya da fieldların erişimlerini belirleme ilkesi.

Access Modifiers ->	private	Default/no-access	protected	public
Inside class	Y	Y	Y	Y
Same Package Class	N	Y	Y	Y
Same Package Sub-Class	N	Y	Y	Y
Other Package Class	N	N	N	Y
Other Package Sub-Class	N	N	Y	Y

Polymorphism Nedir.?

**Alt sınıfı ait bir nesnenin üst sınıfı ait bir nesne olarak gösterilmesi
Üst sınıf değişkeninin alt sınıf nesnelerini referans edebilmesi.**

Polymorphism

Runtime'da

mi çalışır

Compiler Time'da

mi çalışır?

Polymorphism runtime'da çalışır.

Dynamic Binding-Late Binding / Static Binding-Early Binding

Polymorphis’de
upcasting / downcasting
nedir, nasıl yapılır?

Abstract vs Interface

Abstract class'lar birbirleriyle benzer sınıfları bir çatı altında toplamak için kullanılır.
Interfaceler birbirleriyle benzemeyen, farklı sınıfları bir çatı altında toplamak için kullanılır

Abstract class'lardan ve interface'lerden nesne üretilemezler.

Abstract class'ların kendi constructorları olabilir ama, interface'lerin kendi constructorları olamaz.

Abstract class'ların kendi fieldları olabilir, interface'lerin de kendi fieldları olabilir ama final olmak zorundadır ve başlangıç değerleri olmalıdır. Bu fieldlar aynı zamanda default olarak olarak public ve static'tir.

Bir class, bir interface'i ya da bir abstract class'ı implements/extend ettiğinde ilgili abstract class'ın ya da interface'in içi boş methodlarını override etmek zorundadır.

Bir class, bir abstract class'ı extend edebilir ama birden fazla interface'i implements edebilir.

Bir interface başka bir interface'i extend eder. Ve bir interface birden fazla interface'i extend edebilir.

Bir abstract class'ın içerisinde birden fazla abstract method olabilir / hiç abstract method olmayabilir.
Bir interface'in içerisinde de abstract method olabilir / hiç abstract method olmayabilir

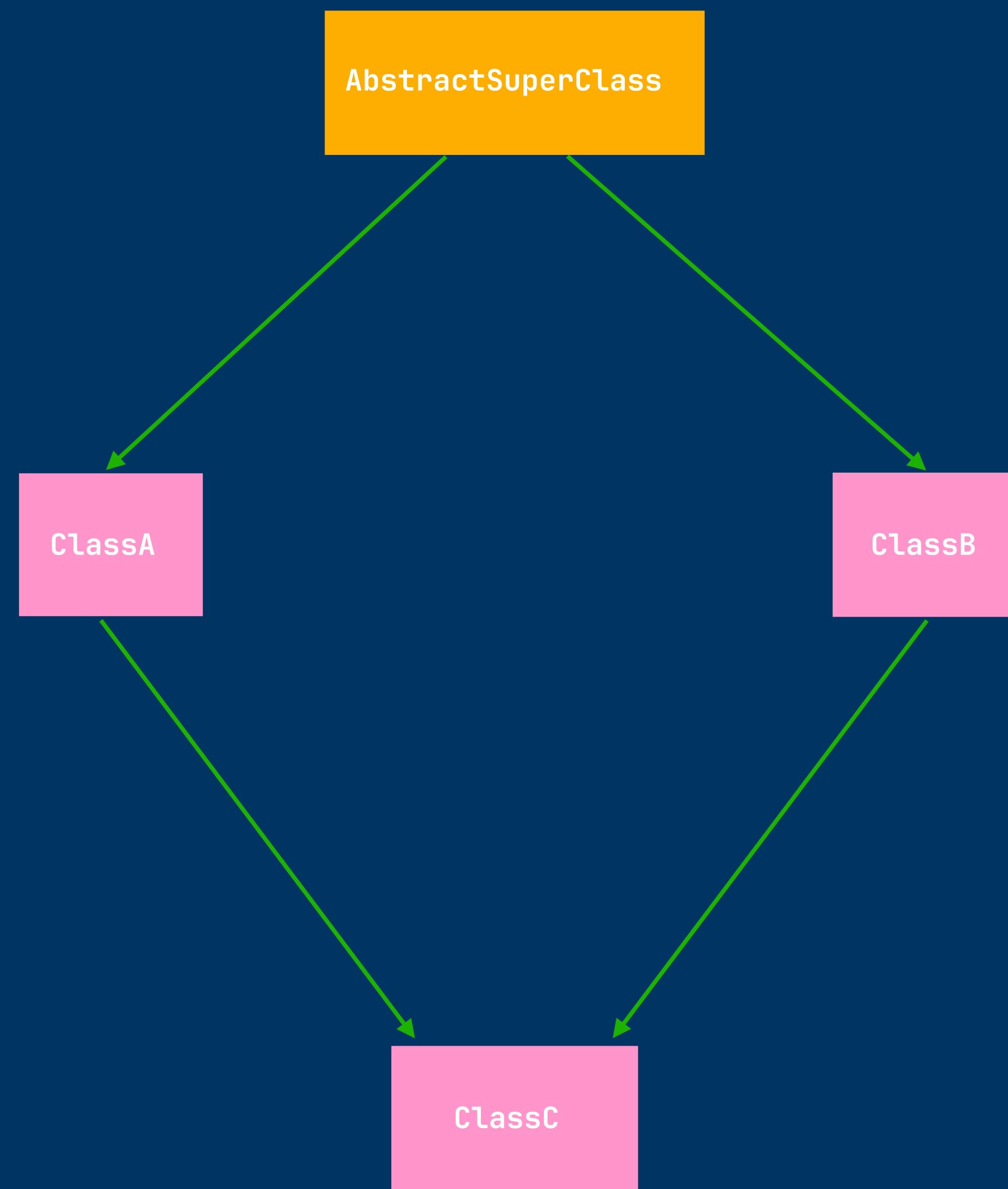
Java8'den sonra private-static-default anahtar kelimeleriyle bir interface içerişine içi dolu method yazılabilir.

Bir interface, bir class'ı ya da bir abstract class'ı extend alamaz. Ama bir abstract class, bir Class'ı extend edebilir.

Eğer bir interface'in sadece bir tane içi boş methodu varsa, bu interfacelere Functional Interface'ler denir. Eğer bir interface'in hiç bir içi boş methodu yok ise, bu interfacelere Marker Interface'ler denir.

Java'da Neden Çoklu Kalıtım yoktur.?

DIAMOND PROBLEM

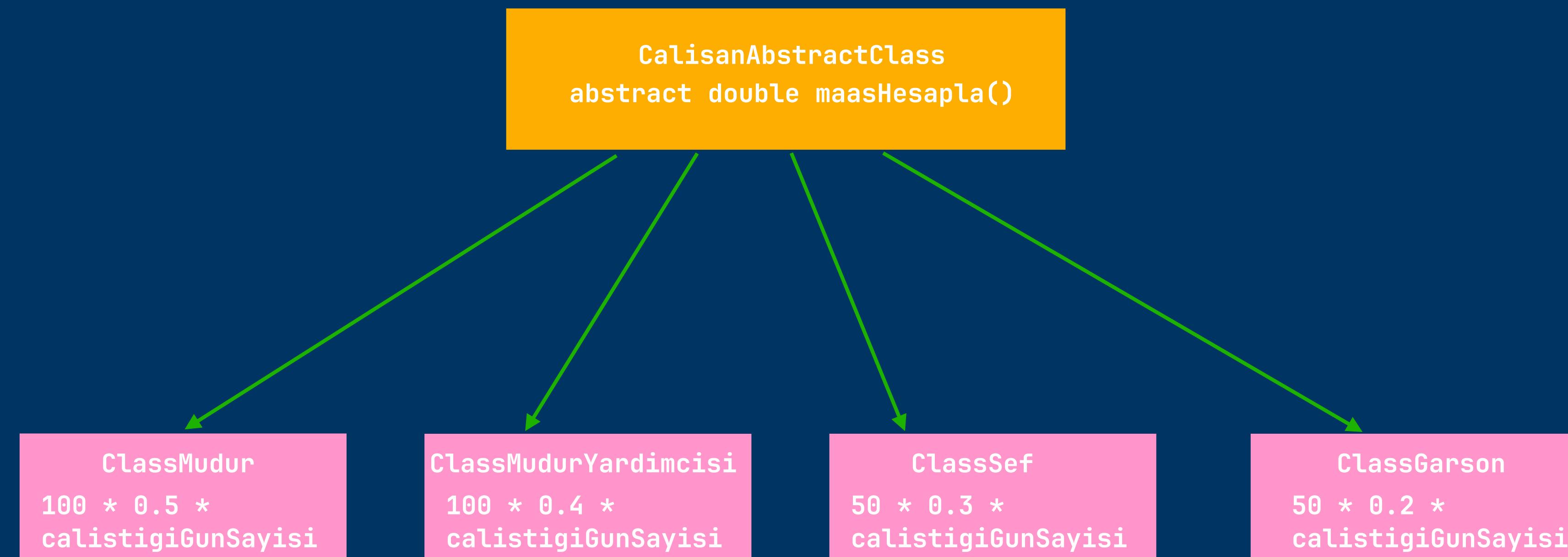


Elimde bir abstract class var.

Bu abstract class'ın bütün methodlarını public ve abstract yaparsam ne olur.?

Teknik olarak başıma ne gelir.?

Teknik olarak böyle bir şey yapmak mümkündür. Bir abstract class, teknik olarak bu şekilde tanımlanabilir. Ama bir class sadece bir kere extend yapabileceği için bu esneklik kaybolmuş olur.



AbstractClass
Class

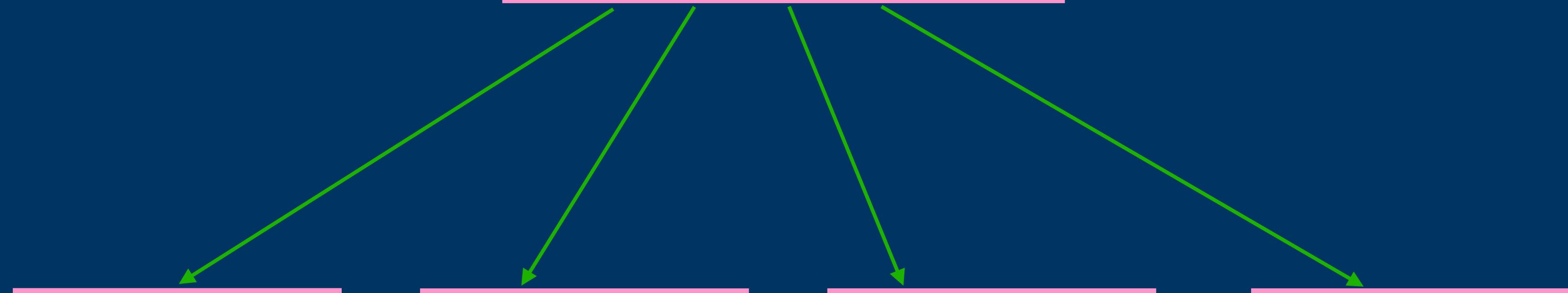
CalisanClass
double maasHesapla() {return 0;}

ClassMudur
 $100 * 0.5 *$
calistigiGunSayisi

ClassMudurYardimcisi
 $100 * 0.4 *$
calistigiGunSayisi

ClassSef
 $50 * 0.3 *$
calistigiGunSayisi

ClassGarson
 $50 * 0.2 *$
calistigiGunSayisi



Eğer Calisan class'ı abstract olmaz ise, Calisan class'ini extend eden class'larda Calisan class'ındaki methodları override etme zorunluluğu ortadan kalkar.

Öyle bir yapı tasarlayın ki, interface’ı implemente eden class ilgili interface’ın methodunu override etmek zorunda kalmasın.

Araya bir abstract class koyulur ve bu abstract class, interface'i implemente eder.
Interface'in içerisindeki methodu gerçekleştirir.
Sonrasında Class, ilgili abstract class'ı extend edip içerisindeki methodu kullanmaz.

Immutable nedir.?

Bir class nasıl immutable yapılır.?

Stringler neden immutable'dır.?

Immutable, bir nesnenin değişmezliği anlamına gelir.
Yani herhangi bir şekilde değiştirilemez.

Bir Class'ın immutable olması için,
final olarak imlenmesi lazım.
Set methodlarının olmaması lazım.
Deep Copy / Shallow Copy uygulanması lazım.

ArrayList vs LinkedList

ArrayList index bazlı çalışırken **LinkedList** index bazlı çalışmaz.

LinkedList'te verinin kendisiyle beraber bir önceki ve bir sonraki verinin adres bilgisi de tutulur. Bu yüzden Ram'de daha fazla yer kaplar.

Her ikisinde de veriler, girildiği sırayla eklenir.

LinkedList'e veri eklemek/silmek daha performanslıdır. Ama, veri okumak **ArrayList**'te daha performanslıdır.

Her ikisi de thread safe'tır.

Map Yapıları

Map yapıları Collection hiyerarşisi içindedir ama Collection interface'ine dahil değildir.

HashMap - LinkedHashMap - TreeMap - EnumMap - WeakHashMap

HashMap'te key değerleri Set olarak tutulur. Value değerleri List olarak tutulur.

HashMap'te key değeri null olabilir.

Key-Value pairlerinin her birisine Entry denir.

HashMap'ın çalışma mantığı

<https://www.youtube.com/watch?v=xsjE0QNcsxc&t=11095s>

Finally Block Nedir.?

Bir try-catch-finally blockunda çalışan bir kod, finally blockuna belli istisnalar haricinde kesinlikle düşer.

Throw vs Throws

`Throw` genellikle hem kontrollü istisnalar, hem de kontrolsüz istisnalar için kullanılır.

`Throws` genellikle kontrolsüz istisnalar için kullanılır.

`Throws`, methodun hemen yanına yazılırken; `throw`, yeni bir exception fırlatıldığında catch blocku içerisinde kullanılır.

Exception vs Error

Error JVM tarafından atılan hatalar olduğu için handle edilemez.
Ama exception handle edilebilir.
Error durumunda program sonlandırılır.
Ama exception durumunda eğer exception handle edilirse program çalışmaya devam eder.

Garbage Collector nedir.?

Java'da unreachable objelerin belirli kısıtlara göre JVM tarafından silinmesi.

<https://www.youtube.com/watch?v=IRuKb1IoAPc>

Tricky Sorular



```
public static void main(String []args){  
  
    Integer a = 50;  
    Integer b = 50;  
    System.out.println("a == b? " + (a == b));  
  
    Integer c = 500;  
    Integer d = 500;  
    System.out.println("c == d? " + (c == d));  
  
}
```

CEVAP

JVM parametrelerinin default değerleri korunduğu varsayılarak
-127 +128 değerleri arasını cacheler ve bu değerler arasına
yazılan iki farklı sayıyı aynı referans noktalarına atar.

Bundan dolayı;
Birinci ifade **true**,
İkinci ifade **false**
çıktısını verir.





```
public static void main(String args[]) {  
    System.out.println(0.1 * 3 == 0.3);  
    System.out.println(0.1 * 2 == 0.2);  
}
```



Float değişkenlerin binary formatında nasıl saklandığı ile alakalı olan bu soruda, 0.2 tam olarak binary sisteme çevrilebilirken 0.3 tam olarak binary sisteme çevrilemez ve bundan dolayı;

false
true
çiktısını alırız.



```
public static void main(String[] args) {  
    int intValue = Integer.MAX_VALUE;  
    intValue++;  
  
    System.out.println(intValue);  
}
```

`Integer.MIN_VALUE`
`-2147483648`



```
class A {  
    static void staticMethod( ) {  
        System.out.println("Static Method");  
    }  
}  
  
public class MainClass {  
    public static void main(String[ ] args) {  
  
        A a = null;  
        a.staticMethod( );  
  
    }  
}
```

static methodlar nesne üzerinden değil sınıf üzerinden çağrıldıkları için örneği yaratılmayan bir sınıfa ait static method da çağrıldığı zaman çalışacaktır.



```
public static void main(String[] args) {  
    String[] array = {"Turkey", "Java"};  
    List<String> v = Arrays.asList(array);  
    v.set(0, "Community");  
    System.out.println(v.contains("Community"));  
}
```

Sonuç "true" olacaktır.
asList'den dönen özelleştirilmiş listede
elemanlarının eklenmesine veya çıkartılmasına
izin verilmmez ancak set operasyonunu
gerçekleştirebiliyoruz.

```
public interface InterfaceA {  
    static void main(String... args) {  
        System.out.println("This is interface")  
    }  
}  
  
public class ClassA implements InterfaceA {  
    public static void main(String[] args) {  
        InterfaceA.main(new String[10]);  
        System.out.println("This is class");  
    }  
}
```

This is interface
This is Class

https://twitter.com/turkiyejavacom/status/1485904339243880448/retweets/with_comments