TECHPROED

PROFESSIONAL TECHNOLOGY EDUCATION

İrfan Canan

JAVA

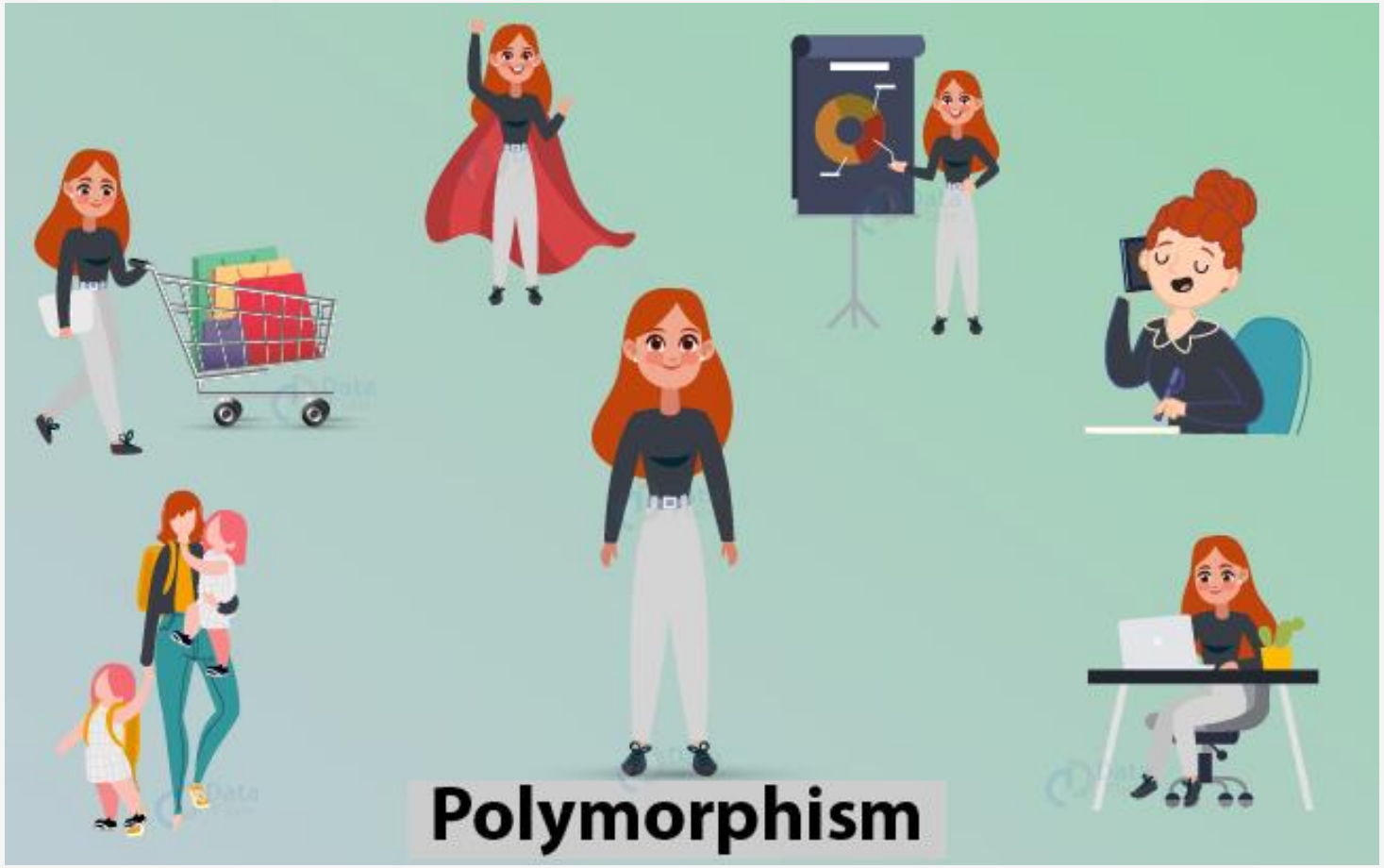
POLYMORPHISM

JAVA POLYMORPHISM

1. • Polymorphism nedir?
   - Gerçek hayatta
   - Lafız olarak manası
   - OOP olarak ve Java'da Polymorphism
2. • İlgili olduğu konular
   - Öncesinde hangi konuları bilmeli? Niçin?
   - Sonrasında hangi konuları bilmeye yardımcı olur? Niçin?
   - Zorluklar
3. • Çeşitleri
   - Nasıl yapılır?
   - Hatırlatma: Biz derste neler görmüştük?
   - Örnekler :
4. • Interview Soruları

POLYMORPHISM NEDİR?

➢Gerçek hayatta:



Real life example of polymorphism: A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behaviour in different situations. This is called polymorphism.

Polymorphism is considered as one of the important features of Object Oriented Programming. Polymorphism allows us to perform a single action in different ways. In other words, polymorphism allows you to define one interface and have multiple implementations.

POLYMORPHISM NEDİR?

➢Lafız olarak : Yunanca Poly (Çok) + Morphism

(şekillilik/çeşitlilik) - Polimorfizm "birçok şekil" anlamına gelir

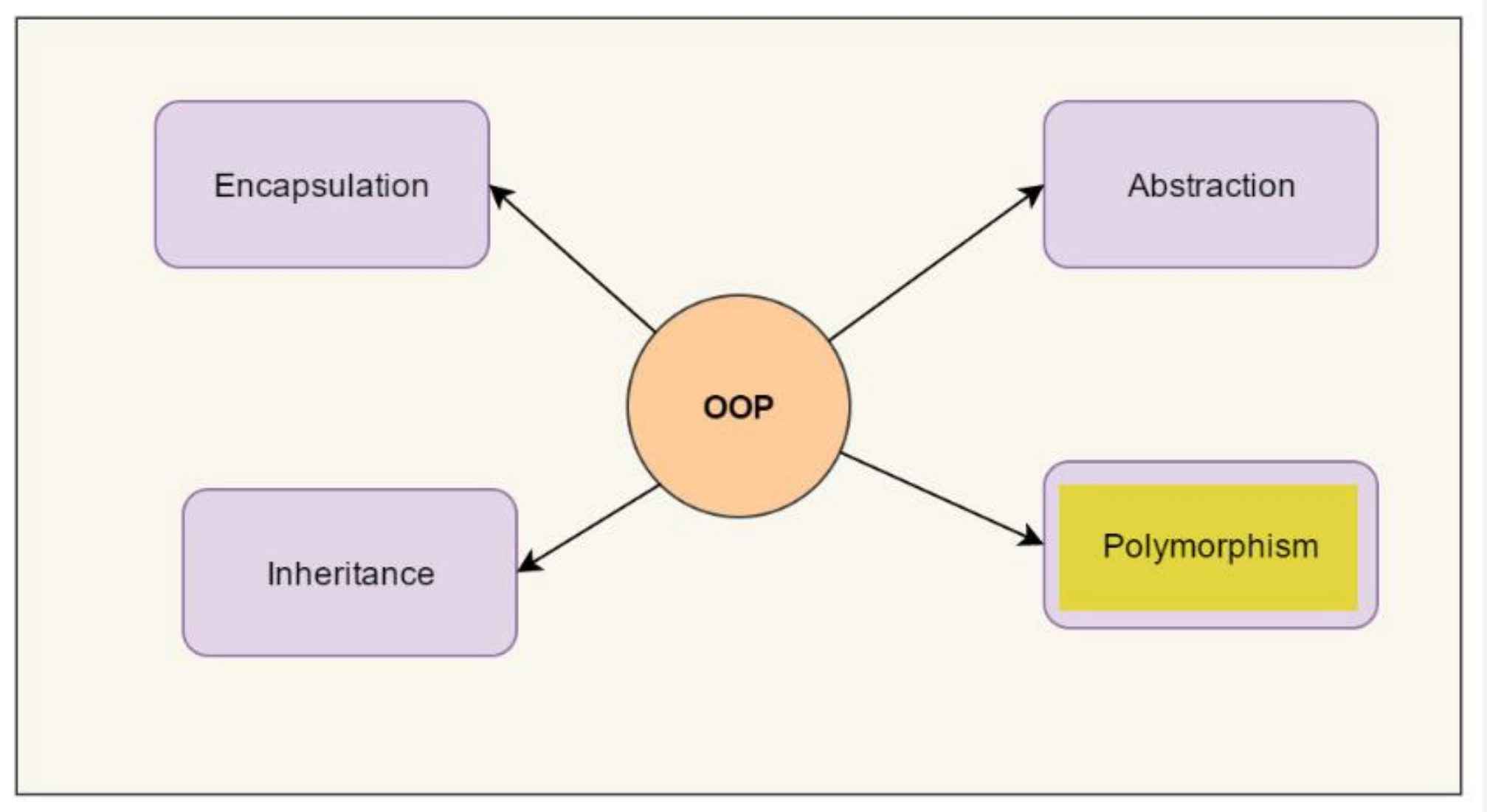
➢OOP/Java'da :

Çok biçimlilik : bir türün bir başka tür gibi davranabilme ve bu tür gibi kullanılabilme özelliği.

- Nesne yönelimli programlama dillerinde çok biçimlilik özelliği ise;
  - aynı temel sınıftan türetilmiş olan sınıflarda paylaşılan(overload yapılan) aynı metodun bu sınıflarda farklı şekillerde uyarlanabilmesidir.
  - Nesnenin davranışı çalışma anında belirlendiği için programcılar, çok biçimlilik özelliği sayesinde nesnelerin türünü önceden bilmek zorunda kalmaz.

- Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So the word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

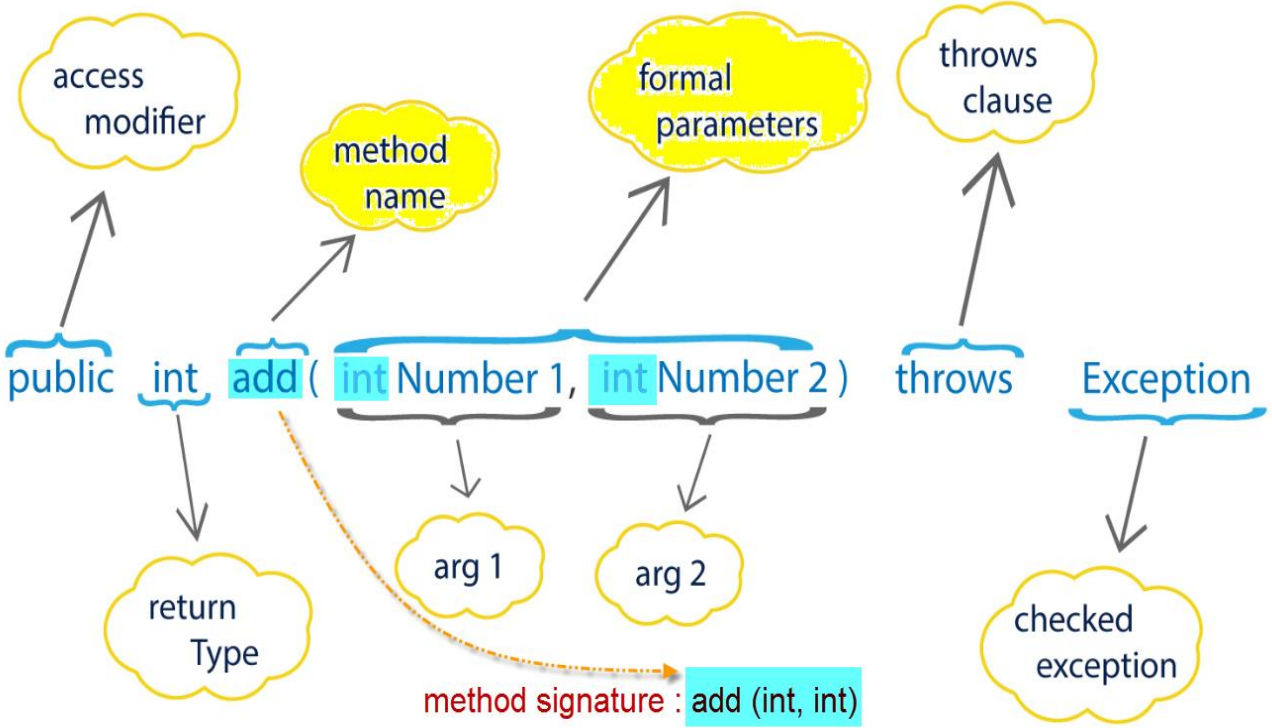
➢OOP/Java'da olmazsa olmazlar :



İLGİLİ OLDUĞU KONULAR

➢Class/Method :

- ➢ Bu kavramlar iyi bilinmelidir.
- ➢ Method Signature kavramını bilmek gerekir.



The reason for the emphasis on just the method name and parameter list is because of overloading. It's the ability to write methods that have the same name but accept different parameters.

Method Signature Examples

```
public void setMapReference(int xCoordinate, int yCoordinate)
{
//method code
}
```

The method signature : *setMapReference(int, int)*

```
public void setMapReference(Point position)
{
//method code
}
```

The method signature : *setMapReference(Point)*

```
public double calculateAnswer(double wingSpan, int
numberOfEngines, double length, double grossTons)
{
   //method code
}
```

The method signature : *calculateAnswer(double, int, double, double)*

İLGİLİ OLDUĞU KONULAR

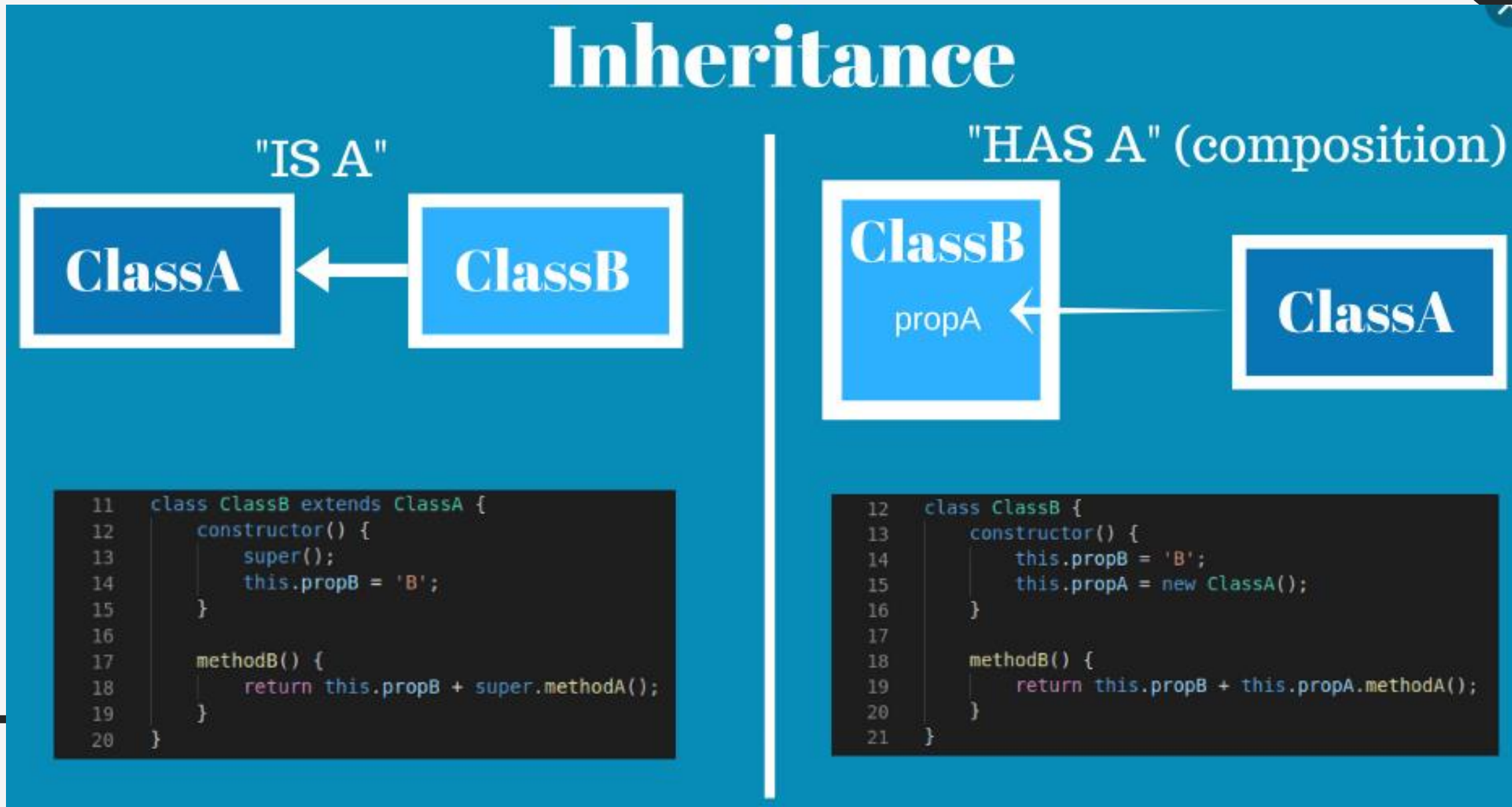
➢ Inheritance:

- Miras (inheritance) yoluyla birbiriyle ilişkili birçok sınıfımız olduğunda ortaya çıkar.
- Aynı temel sınıftan türetilmenin - class hiyerarşisinin- olduğu yerlerde polymorphism'den söz edilir. Yani **polymorphism'in olabilmesi için inheritance olması gerekir.**
  - **Inheritance ile => *Super (parent) Class'taki tanımlamaları extends ile Sub-Class'ta (child) tekrar kullanıyor/tanımlıyoruz.***
- **Inheritance olması için polymorphism olması gerekmez.**
  - A sınıfından türetilmiş bir object, B sınıfının (A'nın atalarından biri) bir object'i gibi davranır.

Inheritance lets us inherit attributes and methods from another class. Polymorphism uses those methods to perform different tasks. This allows us to perform a single action in different ways.

İLGİLİ OLDUĞU KONULAR

- Inheritance:




```
11  class ClassB extends ClassA {
12      constructor() {
13          super();
14          this.propB = 'B';
15      }
16
17      methodB() {
18          return this.propB + super.methodA();
19      }
20  }
```

```
12  class ClassB {
13      constructor() {
14          this.propB = 'B';
15          this.propA = new ClassA();
16      }
17
18      methodB() {
19          return this.propB + this.propA.methodA();
20      }
21  }
```

İLGİLİ OLDUĞU KONULAR

➢ Abstract Class/ Methodlar :

- Abstract Method kavramı iyi bilinmelidir.
- Sub-Class, Abstract Classtaki Abstract Methodlara sahip olmak zorunda.
- Polymorhism iyi bilinirse bu methodların uygulaması kolaylaşır.

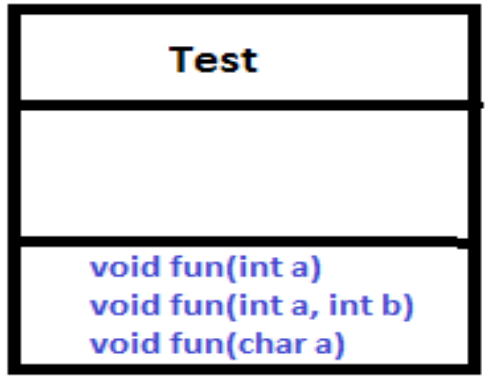
İLGİLİ OLDUĞU KONULAR

Interface :

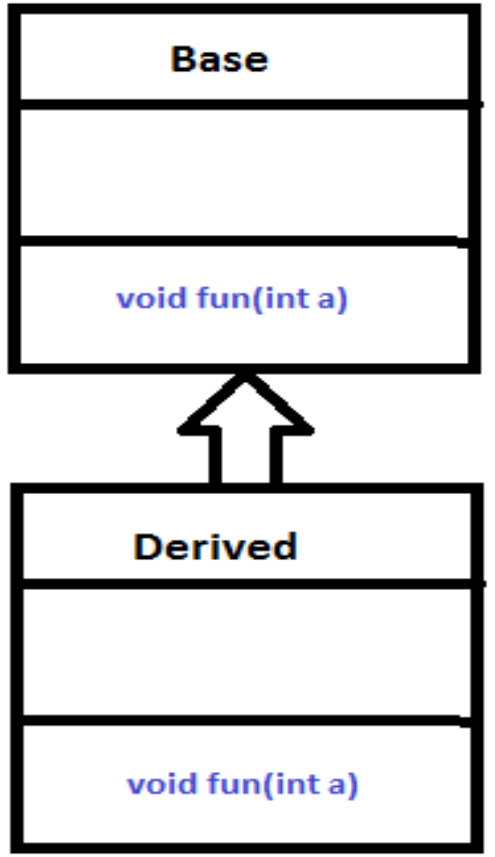
- Interface kavramı iyi bilinmelidir.
- Interface'te abstract methodlar olduğu için polymorphism ile alttaki class'lar bunları uygularlar. İçini doldururlar.

POLYMORPHISM ÇEŞİTLERİ?

- Compile time (*static*) Polymorphism : (Method Overloading)
- Runtime (*dynamic*) Polymorphism: (Method Overriding)



Overloading



Overriding

POLYMORPHISM ÇEŞİTLERİ?

➢Compile time (*static*) Polymorphism : (Method Overloading)

- When there are multiple methods with same name but different parameters then these functions are said to be **overloaded**.
- **Methods can be overloaded by change in number of arguments or/and change in type of arguments**

Example-1 : By using different types of arguments

```
class MultiplyFun {

 // Method with 2 parameter
 static int Multiply(int a, int b)
 {
     return a * b;
 }

 // Method with the same name but 2 double
parameter
 static double Multiply(double a, double b)
 {
     return a * b;
 }
}
```

```
class Main {
 public static void main(String[] args)
 {

     System.out.println(MultiplyFun.Multiply(2,
4));
     System.out.println(MultiplyFun.Multiply(5.5,
6.3));
 }
}
```

POLYMORPHISM ÇEŞİTLERİ?

➢Compile time (*static*) Polymorphism : (Method Overloading)

Example-2 : By using different numbers of arguments

```
//Java program for Method overloading

class MultiplyFun {

 // Method with 2 parameter
 static int Multiply(int a, int b)
 {
     return a * b;
 }

 // Method with the same name but 3 parameter
 static int Multiply(int a, int b, int c)
 {
     return a * b * c;
 }
}
```

```
class Main {
 public static void main(String[] args)
 {
     System.out.println(MultiplyFun.Multiply(2, 4));

     System.out.println(MultiplyFun.Multiply(2, 7, 3));
 }
}
```

POLYMORPHISM ÇEŞİTLERİ?

➢Compile time (*static*) Polymorphism : (Method Overloading)

Example-3 : By using different types & numbers of arguments

Here the method demo() is overloaded 3 times: first method has 1 int parameter, second method has 2 int parameters and third one is having double parameter. Which method is to be called is determined by the arguments we pass while calling methods. This happens at runtime compile time so this type of polymorphism is known as compile time polymorphism.

```
class Overload
{
    void demo (int a)
    {
        System.out.println ("a: " + a);
    }
    void demo (int a, int b)
    {
        System.out.println ("a and b: " + a + "," + b);
    }
    double demo(double a) {
        System.out.println("double a: " + a);
        return a*a;
    }
}
```

```
class MethodOverloading
{
    public static void main (String args [])
    {
        Overload nesne = new Overload();
        double result;
        nesne .demo(10);
        nesne .demo(10, 20);
        result = nesne .demo(5.5);
        System.out.println("O/P : " + result);
    }
}
```

POLYMORPHISM ÇEŞİTLERİ?

➢Compile time (*static*) Polymorphism : (Method Overloading)

Operator Overloading: Java also provide option to overload operators. For example, we can make the operator (‘+’) for string class to concatenate two strings. We know that this is the addition operator whose task is to add two operands. So a single operator ‘+’ when placed between integer operands, adds them and when placed between string operands, concatenates them.
In java, Only “+” operator can be overloaded:

- To add integers
- To concatenate strings

Example:

```
class Main {
 public static void main(String[] args)
 {
     OperatorOVERDDN obj = new OperatorOVERDDN();
     obj.operator(2, 3);
     obj.operator("joe", "now");
 }
}
```

```
//Java program for Operator overloading

class OperatorOVERDDN {

 void operator(String str1, String str2)
 {
   String s = str1 + str2;
   System.out.println("Concatinated String-" +s);
 }

 void operator(int a, int b)
 {
     int c = a + b;
     System.out.println("Sum = " + c);
 }
}
```

POLYMORPHISM ÇEŞİTLERİ?

➢Runtime (*dynamic*) Polymorphism: (Method Overriding)

It is also known as *Dynamic Method Dispatch*. It is a process in which a method call to the overridden method is resolved at Runtime.

Example:

```
//Java program for Method overridding

class Parent {

 void Print()
 {
     System.out.println("parent
class");
 }
}
```

```
class subclass1 extends Parent {
@Override
 void Print()
 {
     System.out.println("subclass1");
 }
}

class subclass2 extends Parent {

@Override
 void Print()
 {
     System.out.println("subclass2");
 }
}
```

```
class TestPolymorphism3 {
 public static void main(String[]
args)
 {

     Parent a;

     a = new subclass1();
     a.Print();

     a = new subclass2();
     a.Print();
 }
}
```

- Method overriding, on the other hand, occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be **overridden**.

POLYMORPHISM ÇEŞİTLERİ?

➢Kısaca

overriding - üzerine yazma/ezme

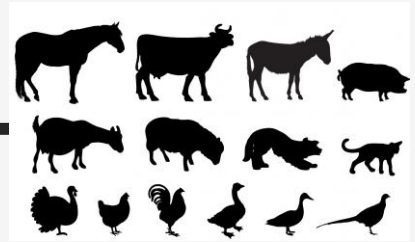
- aynı imza ile yazınca body'si base'i ezer
- bir class'ın bir metodu siz aksini belirtmediğiniz sürece overridebale (onu inherit edip extend yapan bir class o class'ın metodunu ezebilir)
- Kullandırmamak için metodu private etmek akla gelebilir ama o zaman da başka classta kullanılamaz.
- final ifadesi kullanılarak ezmenin önüne geçilir. Olduğu gibi kullan dersek, overrideble olmaz
- en önce alttaki classta var ise o method, onu çalıştırır, üzttekini ezer
- alttaki classtaki methodu silersek program çalışır. Üstteki method çalışır

overloading – methodu aşırı yükleme

- methoda yeni bir özellik eklemek istiyoruz

ÖRNEKLER

- Hayvan => Kedi, Köpek,..vb.
- Araba => Honda, Toyota, ...
- Asker => Er, Yüzbaşı, Binbaşı..



1. **Log**lama (kayıt) yapma yöntemi

   - **Dosyaya**, **Veritabanına**, **Email**'e kayıt etme/gönderme (kritik durumda

2. Bir okulun **mesaj** sistemi

   - **Öğretmene**, **Öğrenciye**, **Veliye**, **Çalışan**lara...

3. Bir okulun **indirim** sistemi

   - **Başarı Bursu miktarı**na göre, **Öğretmen çocuğu**na, **Şehit Çocuğu**, **Kardeş indirimi** durumu,...

4.

- Polymorphism => Coklu yapi

- Overloading ve Overriding ile bir method polymorphism haline getirilir.

- Polymorphism'i Overloading ile yaparsaniz "Compile Time Polymorphism" denir. Diger adi static polymorphim'dir.

- Polymorphism'i Overriding ile yaparsaniz "Run Time Polymorphism" denir. Diger adi dinamic polymorphism'dir.

DERSTE NELER GÖRMÜŞTÜK

- Methodun body kismini degistirmek yeterli olmadi

- Overloading yapma yollari:

- **1)** Parametrelerin sayisini degistirerek ayni isimli method'lar olusturabilirsiniz.

- **2)** Parametre sayisini degistirmeden herhangi bir parametrenin data type'ini degistirerek de ayni isimli methodlar olusturabilirsiniz.

- **Note:** toplama(2,3); cagrisi toplama(int num1, int num2) var oldugu surece toplama(int num1, int num2) yi kullanir, cunku argument data type ile parametre data type birebir ortusuyor.

- Fakat birebir ortusen data type yoksa auto widening'e gore Java kullanacagi method'u secer. Auto widening'e gore kullanabilecegi bir method yoksa Java hata verir.

- **3)** FARKLI data type'indeki parametrelerin yerlerini degistirerek de ayni isimli methodlar olusturabilirsiniz.

DERSTE NELER GÖRMÜŞTÜK

- **Note 1:** Return type'i degistirmek overloading icin yeterli degildir.

- Java methodlarin farkli olup olmadiklarini anlamak icin iki seye bakar.

- **a)** Method ismine **b)**parametrelere bakar . Method ismi ve parametreler Java icin method'larin imzasi gibidir.Java da method imzasi yerine "*method signature*" denir.

- **Note 2:** Access Modifier'lari degistirmek de method'lari Java acisindan farkli hale getirmez.Cunku access modifier'lar method signature'a dahil degildir.

- **Note 3:** Method'larin body'sini degistirmek de method'lari Java acisindan farkli hale getirmez. Cunku body'ler method signature'a dahil degildir.

- **Note 4:** Siz daha kodu yazarken yani kodu calistirmadan once Java sizi yaptiginiz hatalardan dolayi uyarirsa bu tip hatalara "Compile Time Error" denir.

- **Note 5:** Kodu yazarken herhangi bir hata yok fakat kodu calistirdiktan sonra console'da kirmizi hata mesajlari alirsaniz bu hatalara "*Run Time Error*" denir.

- **Note 6:** Overloading "*Compile Time Error*" verir.

DERSTE NELER GÖRMÜŞTÜK

TECHPROED

PROFESSIONAL TECHNOLOGY EDUCATION

- BURADA
- Asker => Er, Yüzbaşı,. örneğini Eclipse'te çözdük.

ÖRNEK

INTERVIEW (MÜLAKAT) NOTLARI

1) Java'da polymorphism'in çeşitleri nelerdir? (What are different types of polymorphism in Java)

2) Overloading ve overriding arasında ne fark vardır? (What is difference between overloading and overriding)

3) Compile-time polymorphism ve Run-time polymorphism nedir?

4) Access modifier'ı overloaded methodta veya overidden methodta değiştirmem doğru olur mu?

(Is it Ok to change access modifier in case of overloaded method or overridden method?

5) Return type'ı overloaded methodta veya overidden methodta değiştirmem doğru olur mu? Veya bu metodların parametrelerini değiştirmem doğru olur mu?

(Is it Ok to change my return type for my overloaded method or overridden method? Or is it OK to change the parameters of these methods?)

6) How about the exception part? Can I throw you exceptions? Can I throw the order of exceptions in case of overloading and overriding these methods?

7) Method verilir ve valid mi diye sorulur? Overloading ve overriding doğru yapılmış mı ? Are they valid? Are they properly overloaded / overridden?

Faydalı Linkler

https://www.w3schools.com/java/java_polymorphism.asp	Animal – Cats, Dogs, Birds örneği
https://tr.wikipedia.org/wiki/%C3%87ok_bi%C3%A7imlilik	IHayvan, TemelHayvanSinifi, Kedi, Köpek örneği (Interface kullanmış)
https://www.geeksforgeeks.org/polymorphism-in-java/	Sınıflandırması güzel
https://data-flair.training/blogs/polymorphism-in-java/	
https://www.learntek.org/blog/polymorphism-in-java/	Örnekleri incelenmeli
https://www.javatpoint.com/runtime-polymorphism-in-java	Şema (UML) incelenmeli
https://www.tutorialspoint.com/java/java_polymorphism.htm	Employee (çalışan) – Salary (maaş) örneği incelenmeli
https://stackify.com/oop-concept-polymorphism/	Kahve makinesi örneği