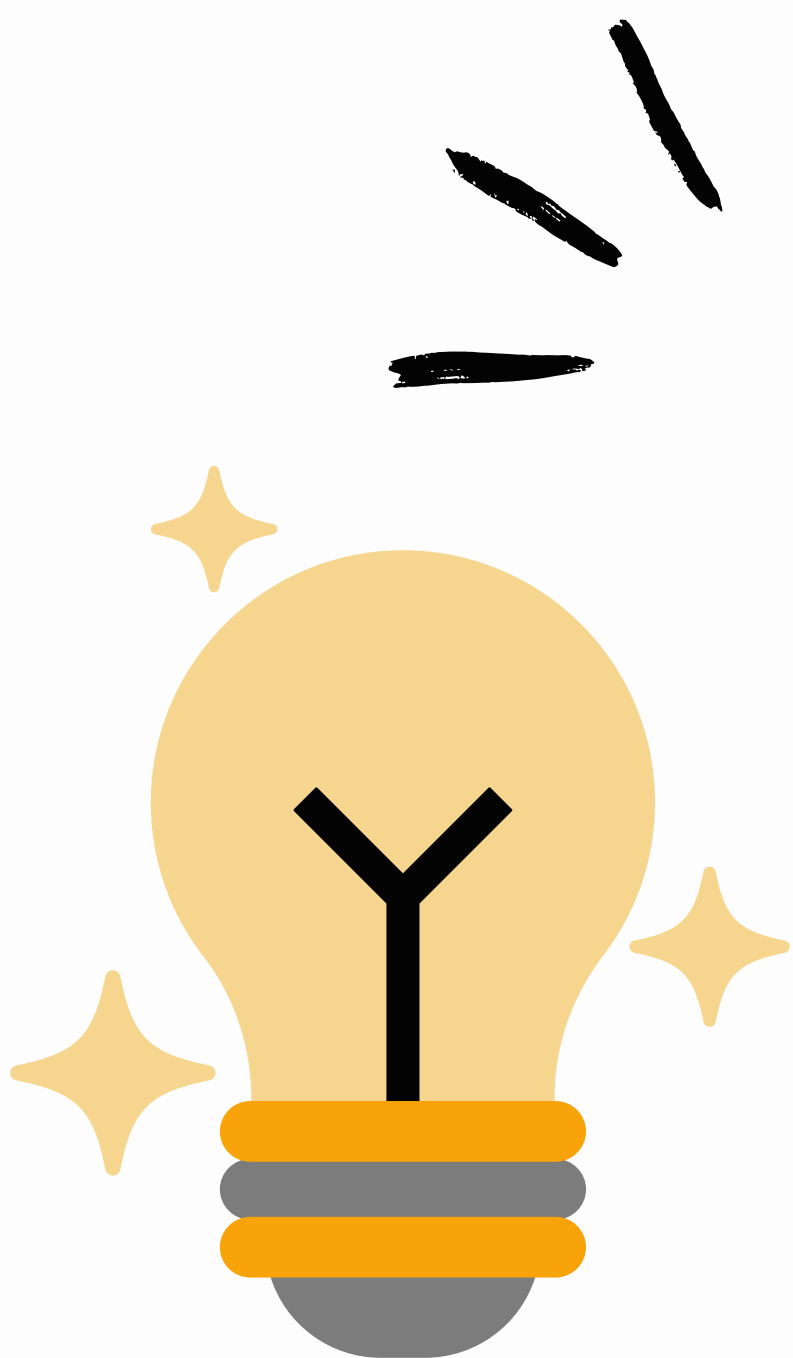
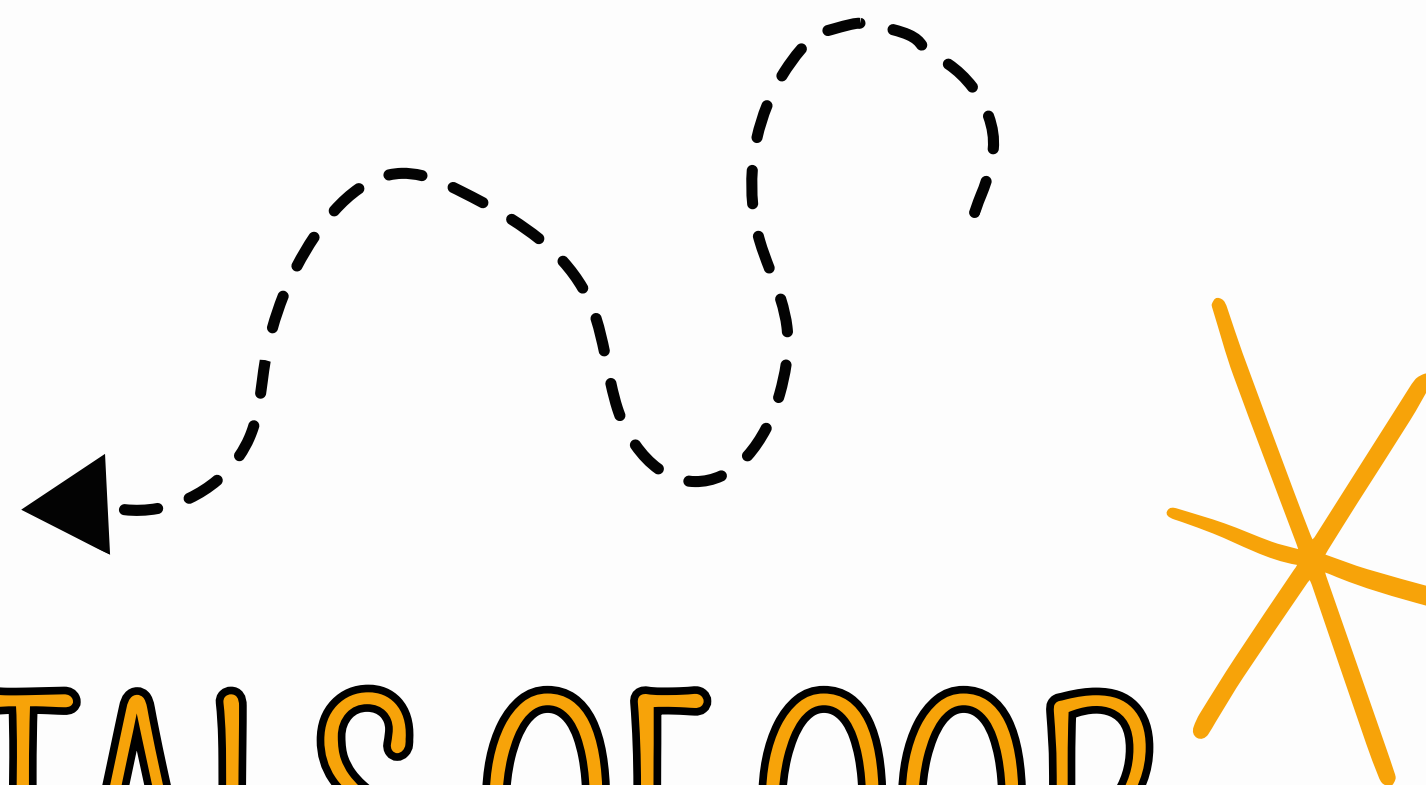


By : Shahd Tarek



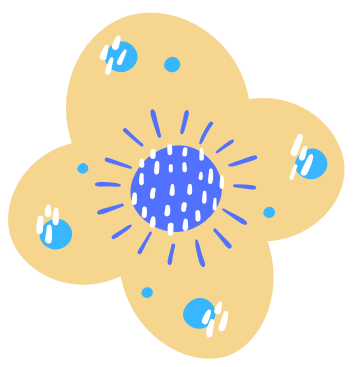
# FUNDAMENTALS OF OOP



# ABOUT ME

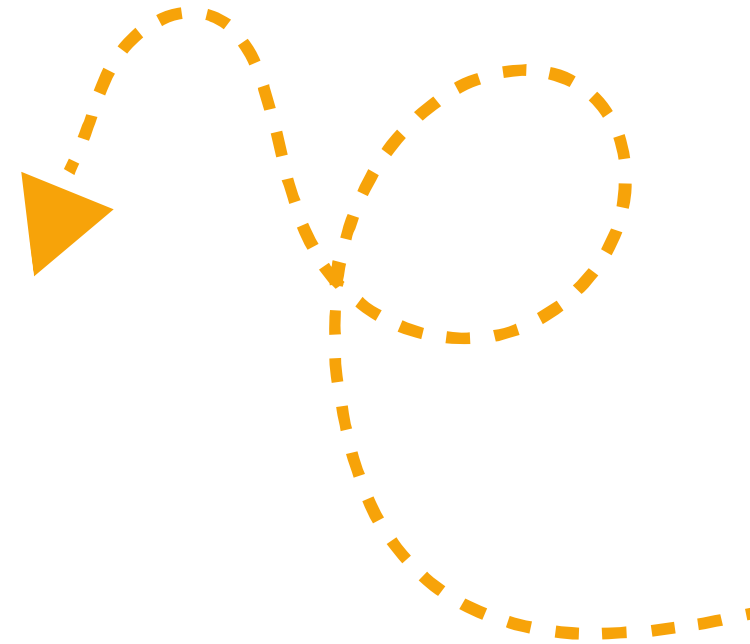
- Shahd Tarek
- BIS Student At MET Academy
- Level 2
- **Linkedin:** Shahd Tarek
- **GitHub:** shahd-tareq



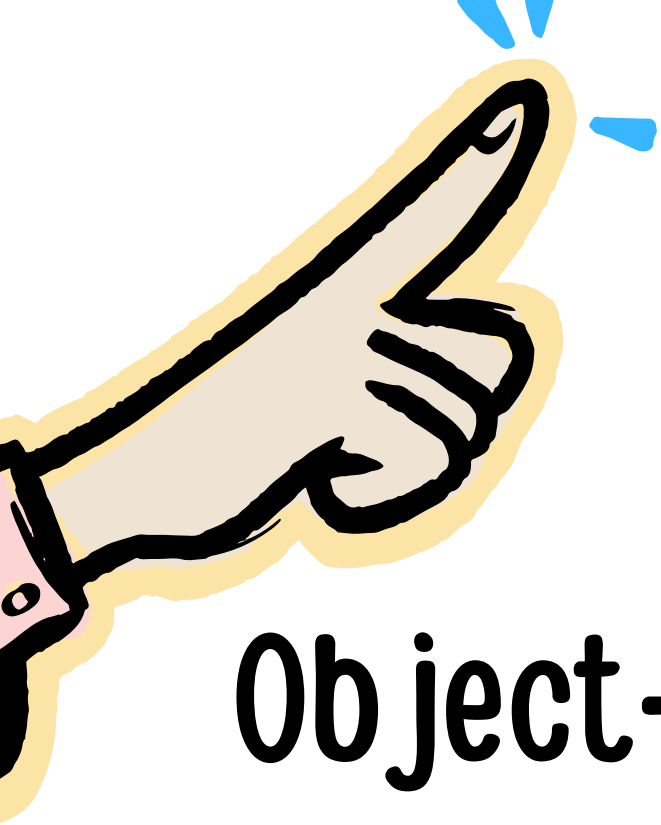


# Presentation content:

- Introduction in OOP
- Abstraction
- Polymorphism
- Inheritance
- Encapsulation
- Examples



# Introduction in OOP



**Object-Oriented Programming (OOP) is a coding paradigm designed to organize programs in a way that makes them easier to understand and maintain. OOP is based on objects, which are units that contain data (Attributes) and functions (Methods) that define their behavior.**



( Object - Oriented  
Programming System)



# 1-Abstraction

هشرحها بطريقة بسيطة اوي  
تخيل انت بتسوق عربيه  
وبتستخدم الفرامل وعداد السرعة  
مثلا والمحرك والدواسه انت اه  
بتستخدمهم بس مش مهتم  
تعرف هم اتعملو ازاي هو ده ال  
Abstraction بيخفي ما وراء  
الكواليس عشان يبسط التعامل  
مع الكود .



# Example

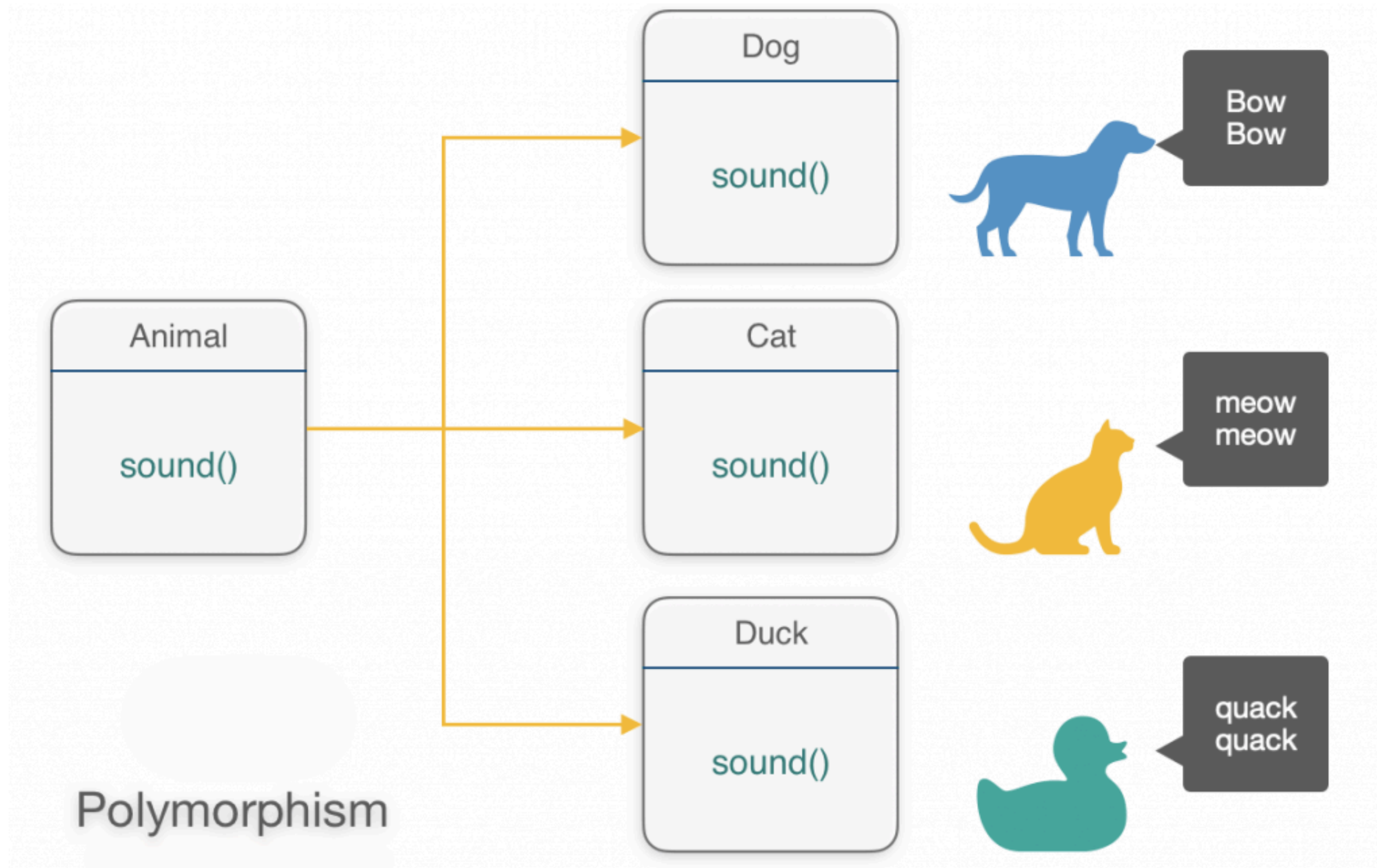


```
1 // كلاس مجرد يمثل السيارة
2 abstract class Car {
3     void start(); // دالة مجردة (يجب تنفيذها في الكلاسات المشتقة)
4 }
5
6 // كلاس تويوتا يرث من كار ويمثل الداله بطريقه
7 class Toyota extends Car {
8     @override
9     void start() {
10         print("Toyota is starting...sound: Vroom Vroom!");
11     }
12 }
13
14 // يرث من كار ويمثل الداله بطريقه bmw كلاس
15 class BMW extends Car {
16     @override
17     void start() {
18         print("BMW is starting...sound: Roooooom!");
19     }
20 }
21
22 void main() {
23     // إنشاء كائنات من Toyota و BMW
24     Car myToyota = Toyota();
25     Car myBMW = BMW();
26
27     // استدعاء نفس الدالة لكل سيارة
28     myToyota.start(); // Toyota is starting...sound: Vroom Vroom!
29     myBMW.start();    // BMW is starting...sound: Roooooom!
30 }
```





# 2-Polymorphism



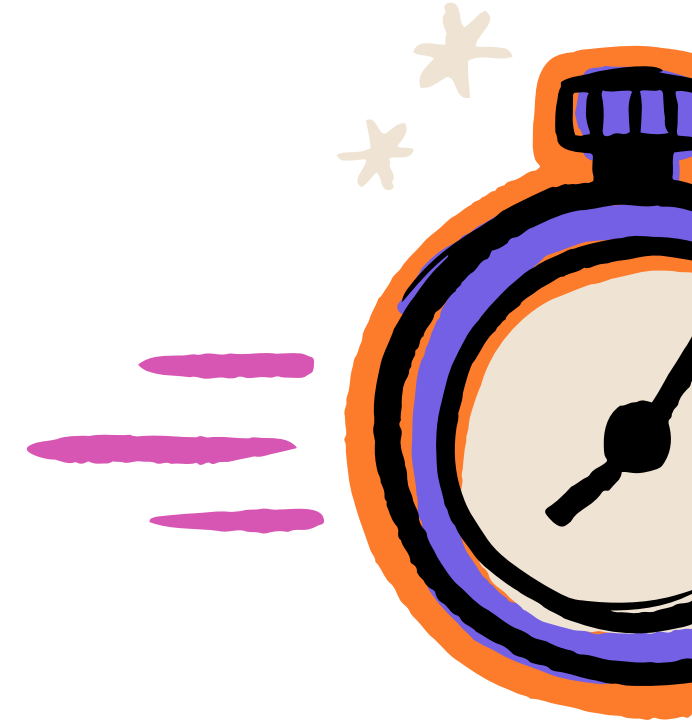
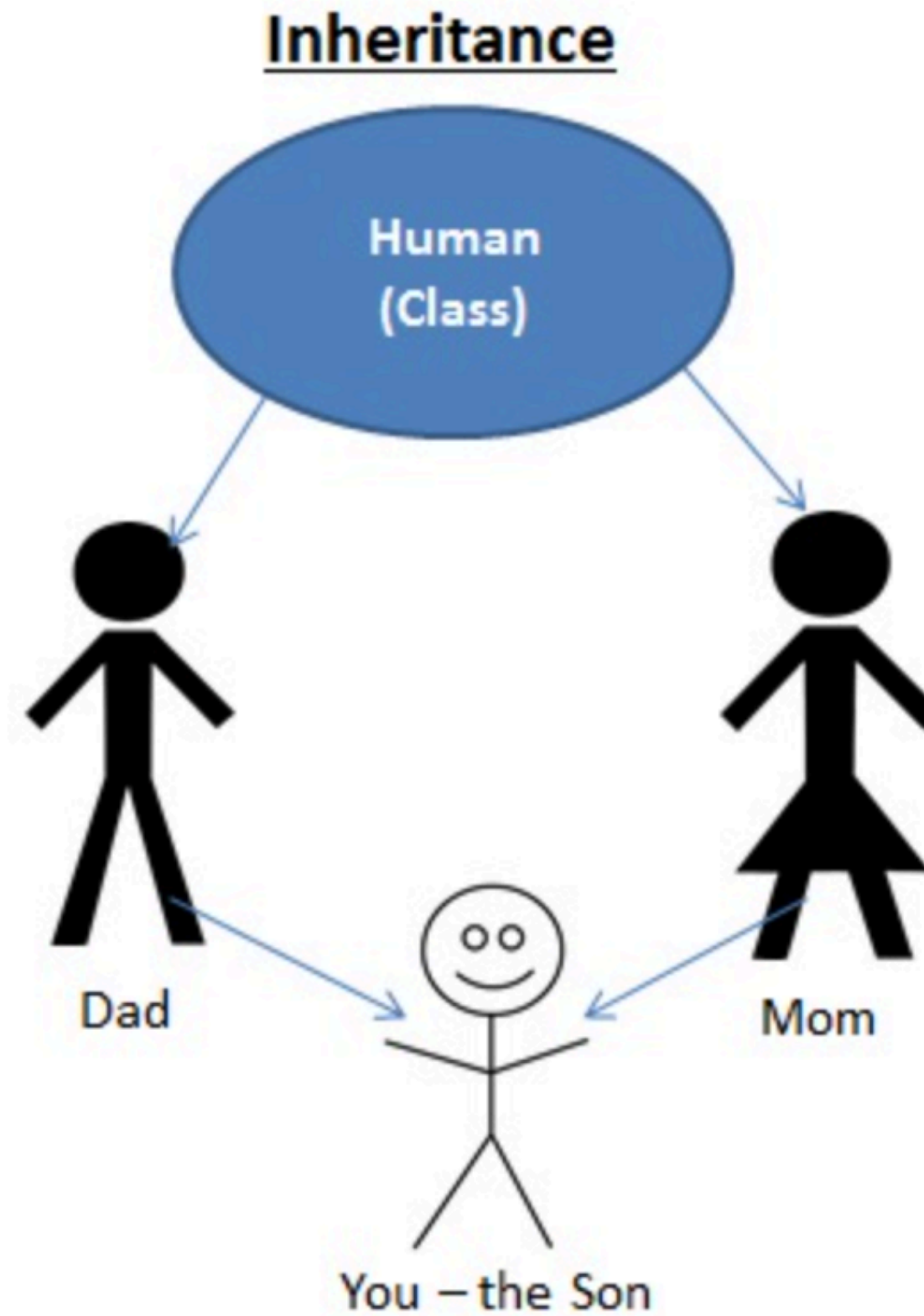


# Example

```
1 class TV {
2     void turn_On() {
3         print("The TV is now ON!");
4     }
5 }
6
7 class Computer {
8     void turn_On() {
9         print("The computer is now ON!");
10    }
11 }
12
13 void main() {
14     List<dynamic> devices = [TV(), Computer()];
15
16     for (var device in devices) {
17         device.turn_On();
18     }
19 }
20 // يعني كل نوع مختلف بيستخدمو نفس الداله ويعملو نفس الحاجه من غير ما تعرف او تهتم بنوعها يعني التلفزيون والتفلون متفقين في زر التشغيل ف يستخدمو نفس الفانكشن ونعمل لوب تلف علي كل واحد وتشغله
```



# 3-Inheritance



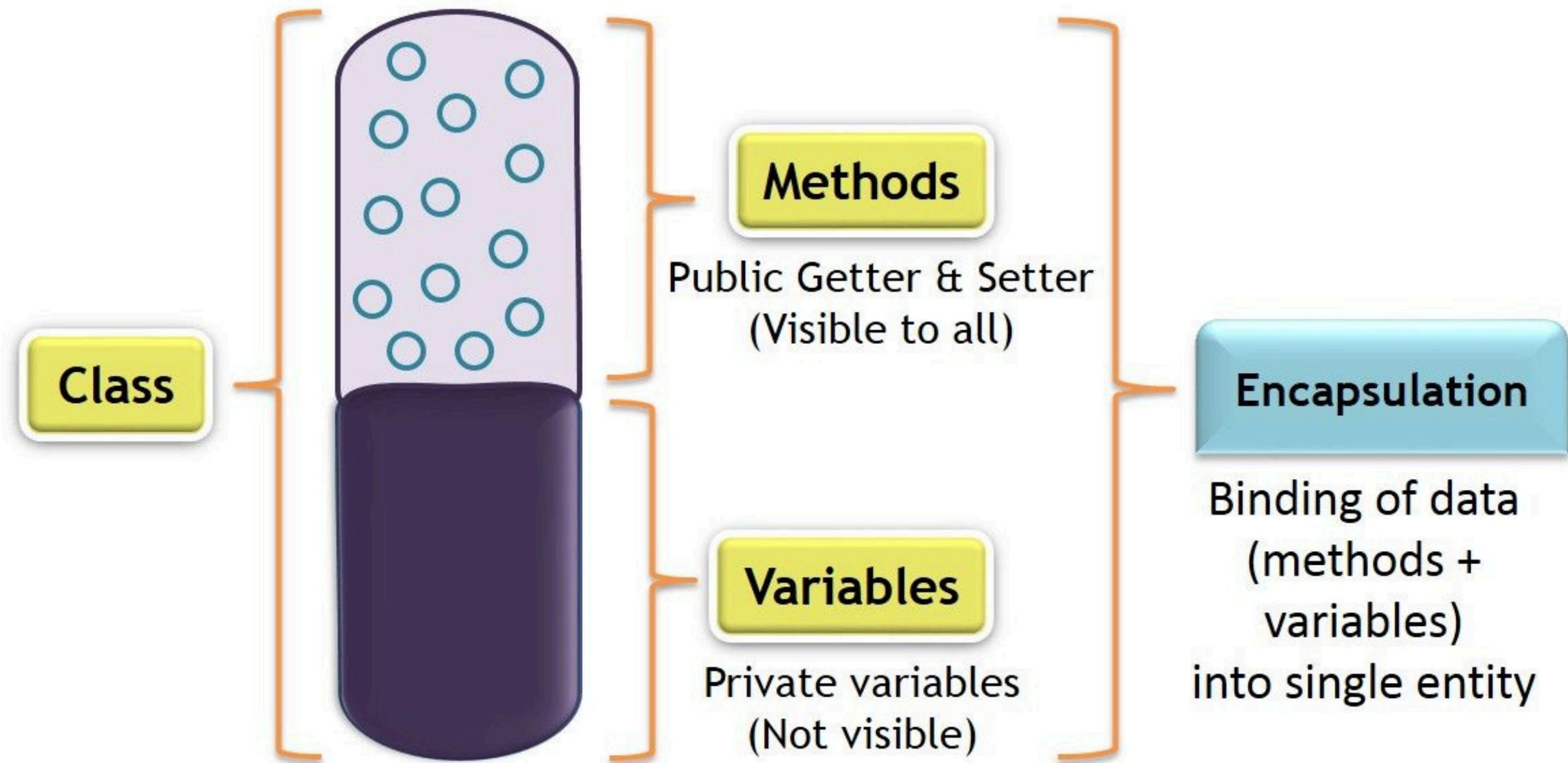
# Example



```
1 // كلاس الأب (Father)
2 class Father {
3     String Name = "Ahmed";
4     String job = "Engineer";
5
6     void work() {
7         print("$Name is working as a $job.");
8     }
9 }
10
11 // كلاس الأم (Mother)
12 class Mother {
13     String Name = "Sara";
14     String hobby = "Cooking";
15
16     void doHobby() {
17         print("$motherName loves $hobby.");
18     }
19 }
20
21 // يرث من الأب والأم (Son) كلاس الابن
22 class Son extends Father {
23     String Name = "Omar";
24     int age = 10;
25
26     void introduce() {
27         print("I am $Name,$age years old.");
28         print("My father is $Name and he is a $job.");
29     }
30 }
31
32 void main() {
33     // إنشاء كائن من الابن
34     Son mySon = Son();
35
36     // استدعاء دوال الابن والأب والأم
37     mySon.introduce();
38     mySon.work();
39 }
40
```



# 4-Encapsulation

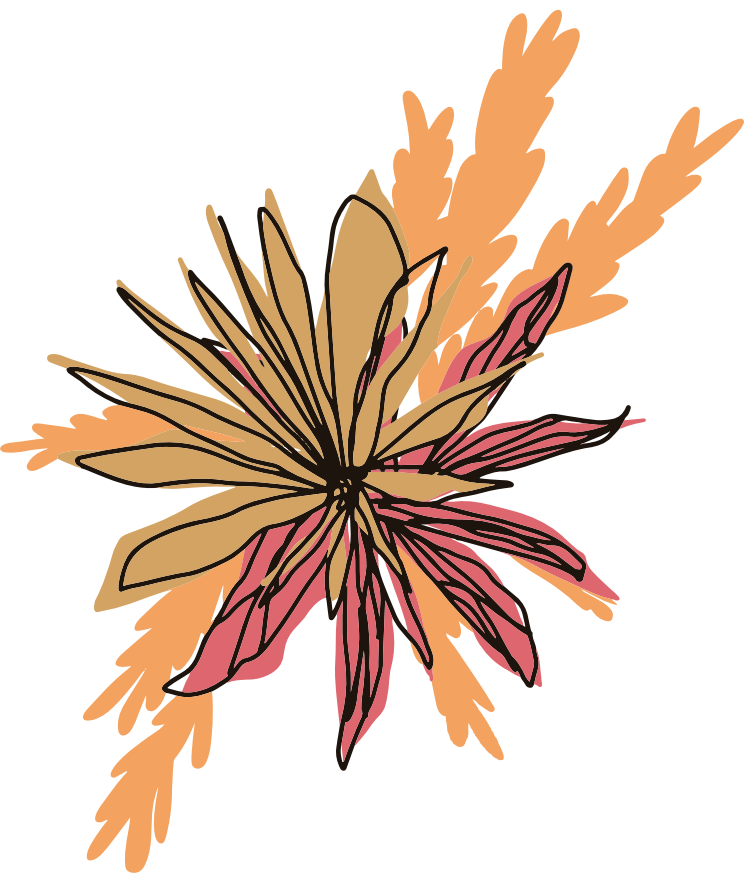


# Example

```
1 class Person {
2     String _name = ""; (Private)//برایفت
3
4     // Getter لقراءة الاسم
5     String get name => _name;
6
7     // Setter لتعديل الاسم بشرط ألا يكون فارغاً
8     void set name(String newName) {
9         if (newName.isNotEmpty) {
10             _name = newName;
11         } else {
12             print("Name can not be empty!");
13         }
14     }
15 }
16
17 void main() {
18     Person person = Person();
19
20     // تعيين اسم جديد
21     person.name = "Ali";
22
23     // طباعة الاسم
24     print("Name: ${person.name}");
25
26     // محاولة تعيين اسم فارغ
27     person.name = "";
28 }
29
30
31
```







# Thank You

