OOP_ Object Oriented Programming

*nesneyi merkezine alan programlama yaklaşımı, paradigması

* OOP- Nesne Yönelimli/Yönelik Programlama

*ilk olarak 1966_1967 yıllarında "Alan Kay" tarafından kullanılmış

*ilk OOP programlama dili "Simula" (OOP'nin bazı özelliklerini destekliyor.)

*ilk tam anlamıyla OOP programlama dili Smalltalk

* 1969_1972 yıllarında Alan Kay, Dan Ingalls, Adele Goldberg ve arkadaşları tarafından Smalltalk Xerox PARC'da geliştirilmiş.

* C++, C#, Java, Python OOP'yi destekliyor.

* Objects interact with each other to perform the program functions.

*each object can be characterized by a state and behavior.
An object keeps the current state in the fields and the behavior in the methods

4 basic principles of OOP

-Encapsulation= Hiding the internal data
-Abstraction=simplified abstract version of implementation.
-Inheritance= defining parent-child relationships.
-Polymorphism= one name and many forms
It is done with <u>overloading</u> or <u>overriding</u>

-Encapsulation = Saklama/sarmalama
-Abstraction = Soyutlama
-Inheritance = Kalıtım
-Polymorphism = Çok biçimlilik

$$\begin{tikzcd}
 & \text{Soyutlama} \arrow[dl] \arrow[dr] & \\
\text{Veri} & & \text{Kontrol}
\end{tikzcd}$$

(Fonksiyon, alt programlar)
(veri tipinin yapılandırılması)

• Saklama/sarmalama
Arabanın deposunda ne kadar yakıt kalmışı anlamak için yakıt göstergesine bakılır Deposuna bakılmaz.
Saklama sınıflarda gerçekleştirilir

• Soyutlama ③ (!)

$$\begin{tikzcd}
 & \text{Bir insan (örnek yaşlı teyze)} \\
\text{Kedi} \arrow[ur] \arrow[dr] & \\
 & \text{Veteriner}
\end{tikzcd}$$

* Kalıtım

$$\begin{tikzcd}
 & \text{Taşıt} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{Kara} \arrow[d] \arrow[dr] & \text{Hava} & \text{Deniz} \\
\text{Otomobil} & \text{Kamyonet} &
\end{tikzcd}$$

* Çok Biçimlilik
  Otomobil Sür()
  Kamyonet Sür()

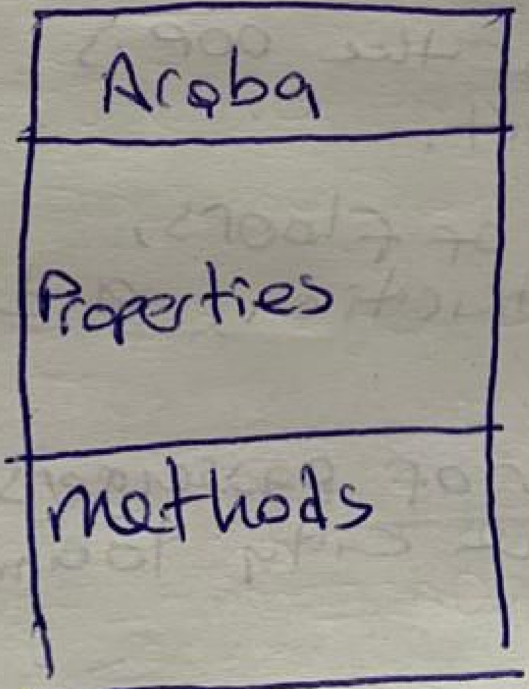
* The key notion of the OOP is naturally, an object.
- a building → number of floors, the year of construction, and the total area.

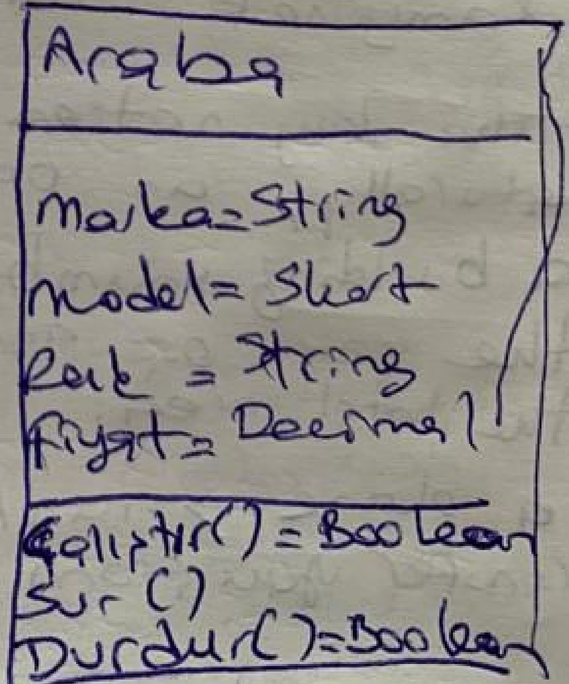
- a plane → number of passengers, transfer you from one city to another

* A class is ④ another important notion of OOP.
* A class describes a common structure of similar objects = their <u>fields</u> and <u>methods</u>.
* It may be considered a <u>template</u> or <u>blueprint</u> for similar objects.
* We can say these object belong to the same <u>type</u> or <u>class</u>
* An object is an individual instance of a class.

* UML

Class





*Sınıfın tek sorumluluğu olmalıdır.
SOLID prensibinden S→Single Responsibi
(single responsibility)

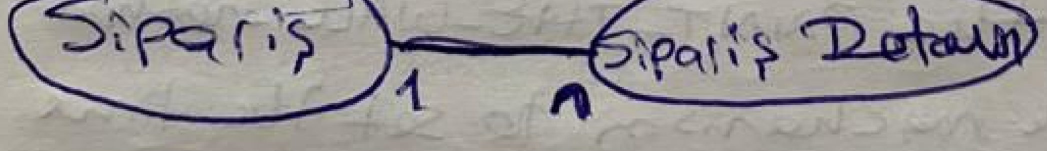
*Her sınıf Object sınıfından türetilmektedir.

$\underbrace{\text{public}}_{1}$ $\underbrace{\text{class}}_{2}$ $\underbrace{\text{Araba}}_{3}$

erişim belirleyici, keyword, sınıfın adı
(access modifier)

public (UML gösteriminde +)
protected (// // #) <u>inheritance</u>
internal (// // ~) <u>package</u>
private (// // —) <u>ait olduğu sınıf</u>

*ilişki (Relationship)



*Design Principles
*Design Patterns
*Clean Code
*Coupling, Cohesion

OOP ⑥

* doğaldır
* basittir
* yapısal programlama yaklaşımının daha geliştirilmiş ve gerçeğe yakın versiyonudur.
* birbiriyle iletişim ve etkileşim halinde nesneler tasarlanır.
* projenin erken aşamasında, gerekli tüm nesneler tanımlanır, sınıflar oluşturulur ve ilişkiler tanımlanır.

* Encapsulation = restricting direct access to some components.
- Think of the wall color.
  Call the PAINT_THE_WALL method
- Clock mechanism to set the time.

* Getters and Setters
Arbitrarily change an object is not true
* Domain-specific methods
- operations for changing the time
- SET_TIME, ADD_HOUR, ADD_MINUTE, ADD_SECONDS, CHANGE_TIMEZONE

Protecting Data

1. keywords = to set the visibility for the fields of a class exact scope for accessing the variables

2. name convention = setting up a rule, like attributes starting wit a capital letter are public.

3. name mangling = changing a fieldname for external access.

Interfaces:

* A contract, legal aggreement
* A collection of methods that describes the behavior of an object.

* One-method Interface

- loudspeakers and headphones implement the interface with a method MAKESOUND
- sensible to choose suitable names for interfaces

  SOUNDMAKER

* use verbs for method names
  - nouns or adjectives for interface names

* method READ
  interface READER, READABLE

method TRANSFORM
interface TRANSFORMER, TRANSFORMABLE

*Complex Interface

interface AUDIOPLAYER

+PLAY()
+ STOP()
+NEXT()
+ VOLUME-UP()
+ VOLUME_DOWN()

*Responsibility of an Interface

-There is no need for some methods, move them to a new interface, or even remove them

interface AUDIOPLAYER

method = REWIND() but

no need RECORD()