UNIVERSIDAD DE LAS FUERZAS ARMADAS "ESPE"

PROG. ORIENTADA A OBJETOS

NRC: 1940
FECHA: 28/08/2024

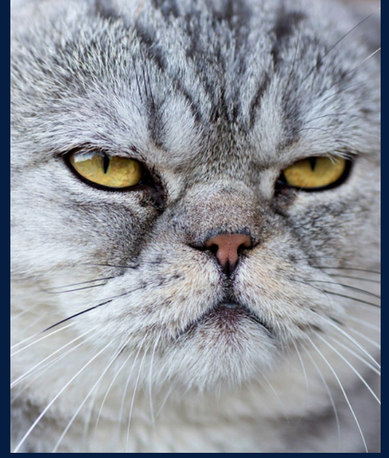
Members:

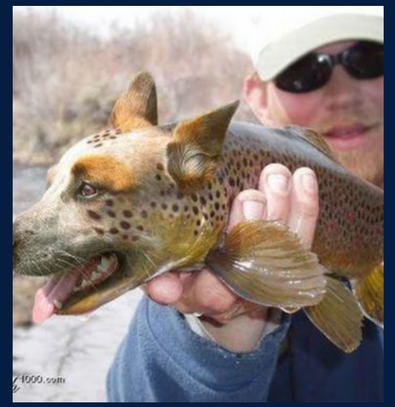
-CHASIPANTA IAN

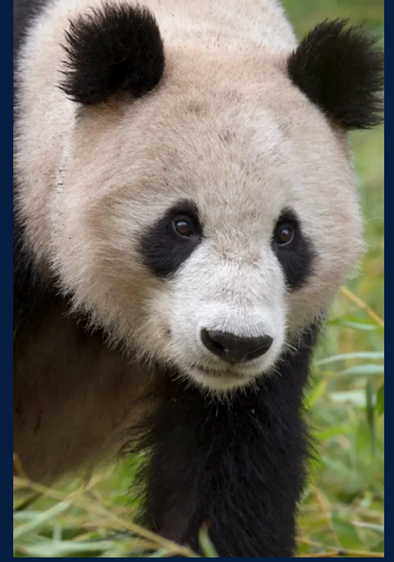
-CALVOPIÑA DAVID

-CHASIPANTA SAUL

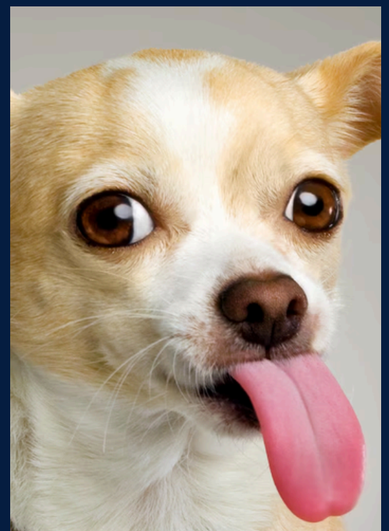
-ROJAS JONATHAN



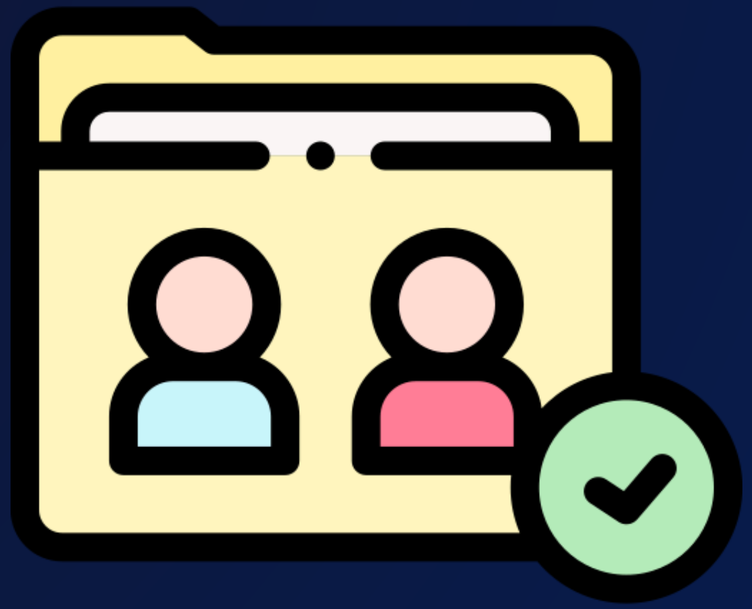




POLYMORPHISM IS A FUNDAMENTAL CONCEPT IN OBJECT-ORIENTED PROGRAMMING THAT REFERS TO THE ABILITY OF AN OBJECT TO ACT IN MANY WAYS.



Allows different classes to be treated as instances of a common class through a shared interface

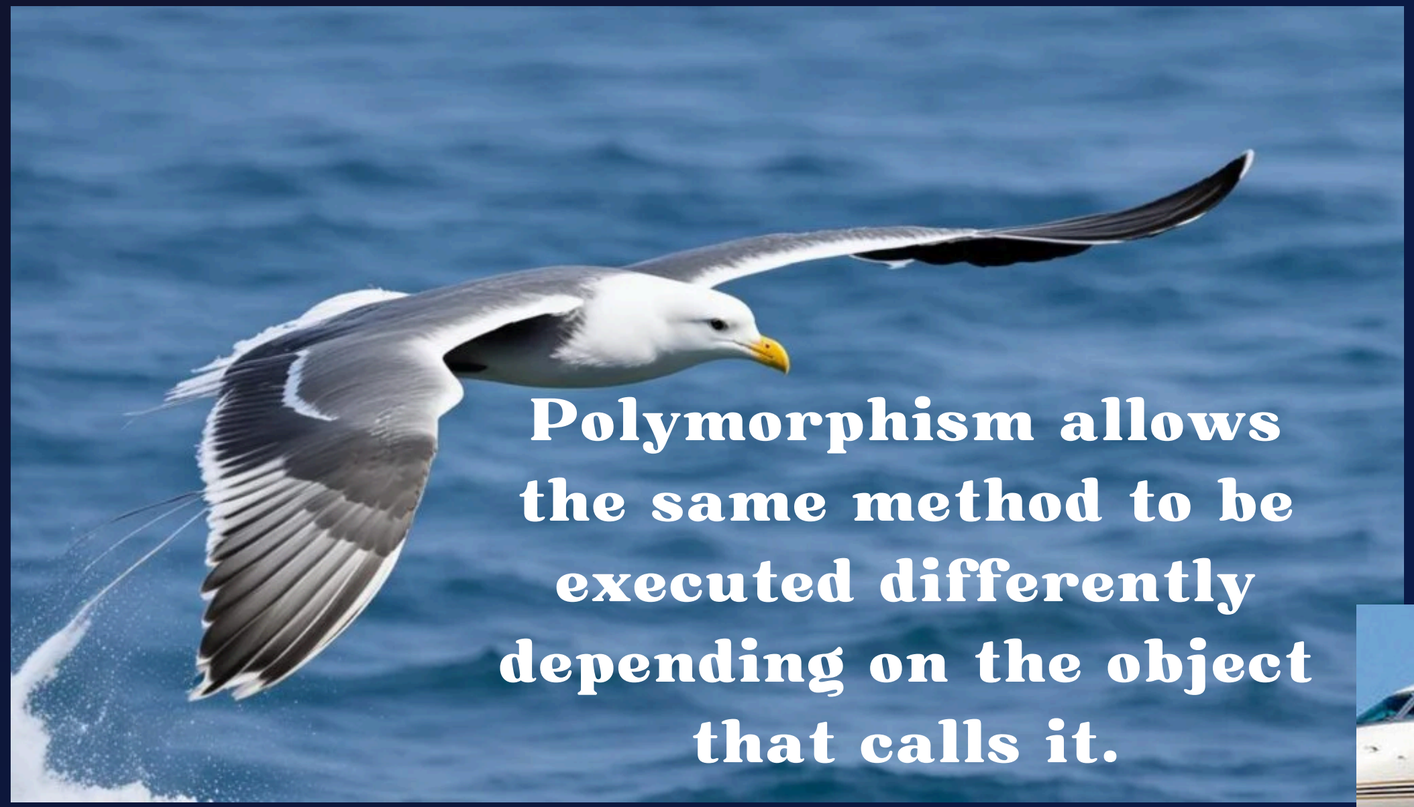


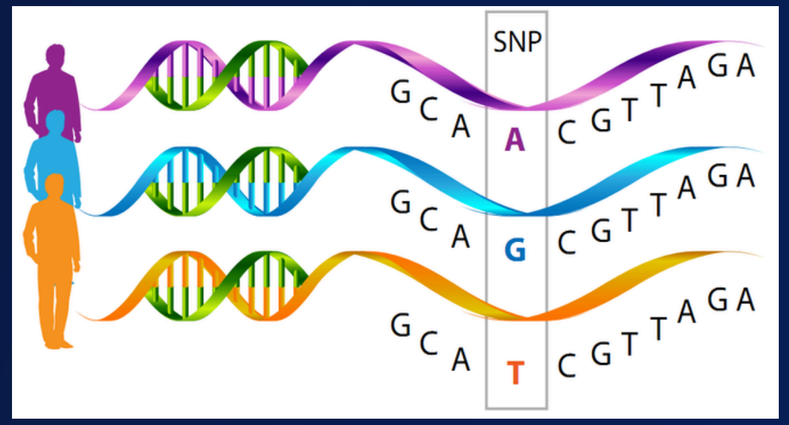
This means that you can have classes (Dog, Cat), are different and specific in their conduct, can share certain ”similarities”.

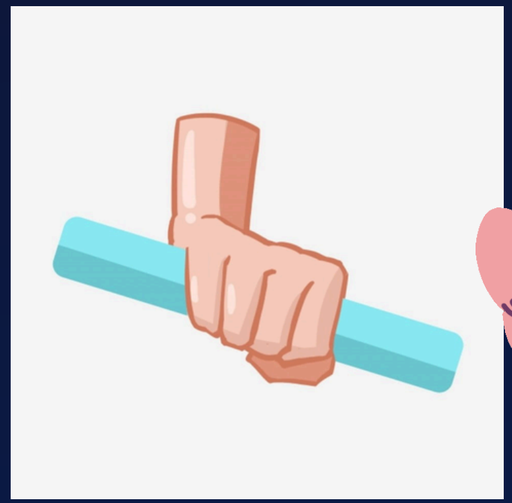
No matter what type of specific object you are using (Dog or Cat), you can invoke the same methods defined in the base class.

All of these specific classes (Dog, Cat) share a base class or ”superclass” (Animal)

The ”shared interface” refers to the set of methods or behaviors that are defined in the base class and that the child classes must implement (or already inherit).


Polymorphism allows
the same method to be
executed differently
depending on the object
that calls it.


SNP
G C A A C G T T A G A
G C A G C G T T A G A
G C A T C G T T A G A



Runtime polymorphism (overwrite)

It allows a subclass to provide its own implementation of some method that has already been defined in a superclass, this is important since it allows code reuse since subclasses can reuse the name of the superclass method and modify its behavior without altering the superclass code. And second, it provides flexibility in execution

polymorphism by method overloading

Method overloading polymorphism in Java Object Oriented Programming (OOP) is a concept that allows multiple methods with the same name to be defined within a single class, but with different parameter lists. This allows methods to perform similar tasks but adapted to different types or numbers of arguments.

Method overloading occurs when you have multiple methods with the same name but that differ in:

- **Number of parameters.**
- **Types of parameters.**
- **Order of parameters.**

Overloading does not depend on the return type of the method; it should only be differentiated by the parameters.