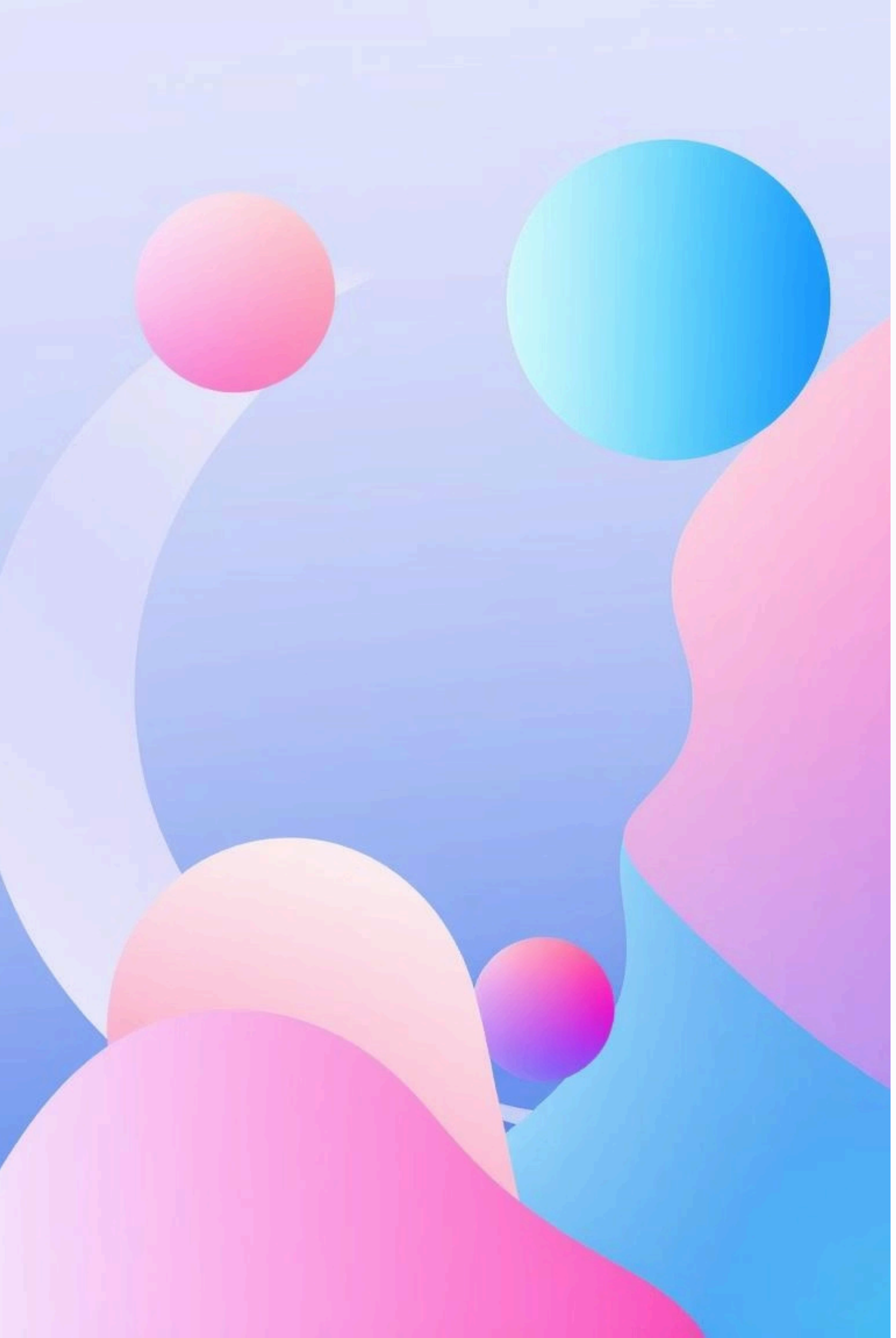


lec 4:

OOP Part 2:

multiple inheritance problem

Solid violate

```
void main() {}

class Employee {

void recordAttendce() {}

}

class Nurse extends Employee {

int takeTemp() {

return 0;

  }

}

class Doctor extends Employee extends Medical {}

class ItEngineer extends Employee {}

class Medical{}
```

Mixins

to add new behavior for specific case

When you want to **avoid multiple inheritance (**which Dart doesn’t support **)**

`mixin Medical {}`

task

You are building a **pet management system**.

There are different types of pets:

- `Dog`
- `Bird`
- `Fish`

Each pet can have **different abilities**:

- `walk()`
- `fly()`
- `swim()`

Not all pets have all abilities:

- Dogs can **walk**
- Birds can **walk and fly**
- Fish can **swim**

What is Polymorphism in OOP?

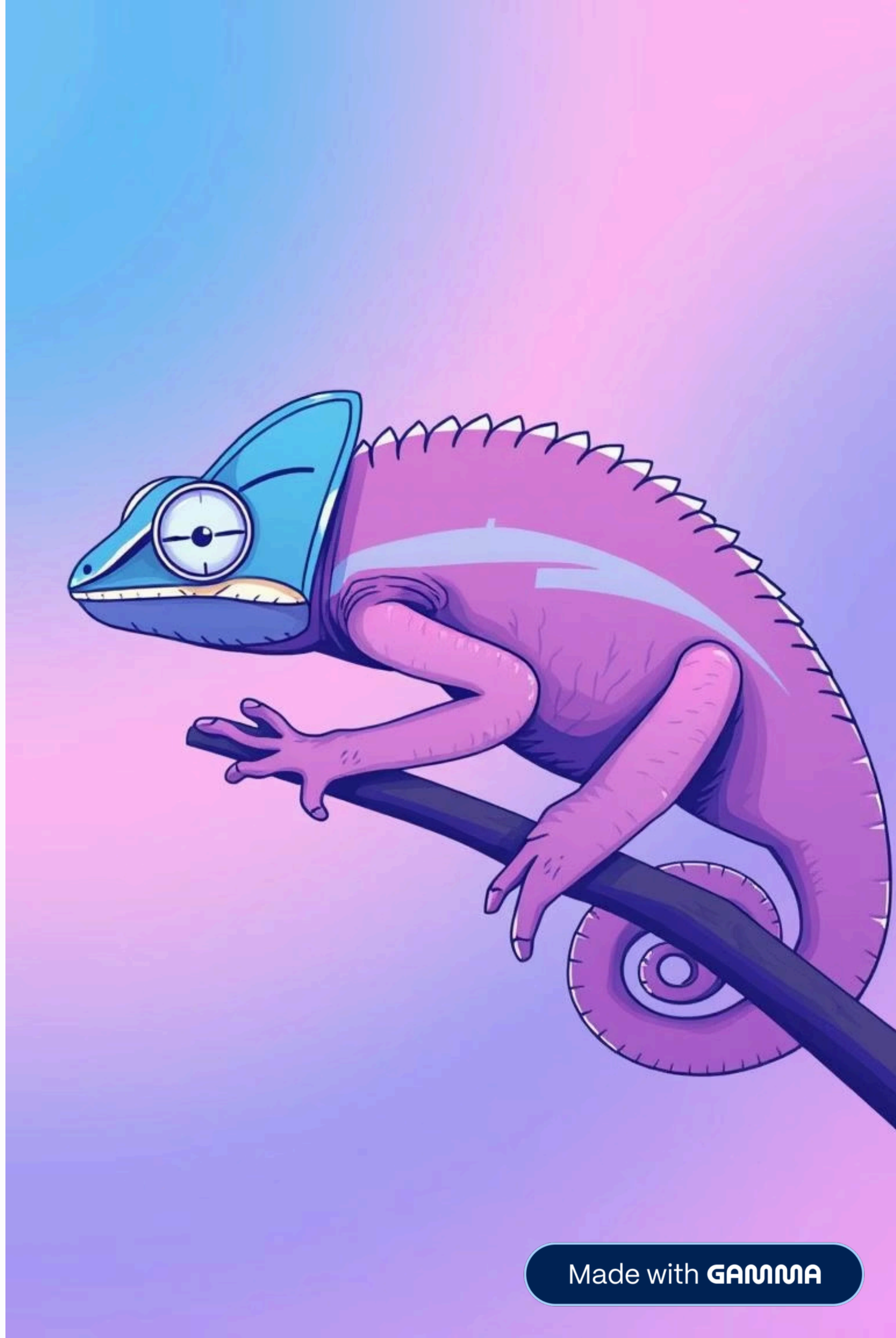
When ? ==> common behavior bettwen Objects

"Many Forms"

Polymorphism literally means **"many forms"**. In OOP, it refers to the ability of an object to take on many forms. It allows one interface to be used for a general class of actions.

Dynamic Behavior

Polymorphism enables flexibility and dynamic method behavior at runtime. The specific method that is called is determined by the actual type of the object, not by the type of the reference variable.



Why Use Polymorphism?

Simplifies Code

Code Reusability

Polymorphism Example: Animal Sounds

```
class Vehicle {
 int year;
 String brand;

 Vehicle({required this.brand, required this.year});

 void start() {
  print("Vehicle start");
 }

 void stop() {
  print(" stop");
 }

 //show info
 void showInfo() {
  print("brand: $brand, year: $year");
 }
}
import 'package:dart_package_1/src/vehicle.dart';

class Motocycle extends Vehicle {
 Motocycle({required super.brand, required super.year});

 @override
 void start() {
  print("Motocycle is start");
 }
}
import 'package:dart_package_1/src/vehicle.dart';

class Car extends Vehicle {
 Car({required super.brand, required super.year});
 @override
 void start() {
  print("Car is start");
 }
}
```

task

You are building an **online store** with different types of products:

- `Book`
- `Electronic`
- `Clothes`

Each product has:

- `name` (product name)
- `price` (product price)

Each product type has a method `showDiscountedPrice()` that returns the price **after discount**:

- Book → 10% discount
- Electronic → 20% discount
- Clothes → 15% discount

What is Abstraction in OOP?

hide the internal details

```
abstract class Animal

{

void makeSound(); // abstract method

void sleep() { print("Sleeping..."); } }

class Dog extends Animal {

@override

 void makeSound()

{ print("Woof! Woof!"); }

}

void main()

{

// Animal a = Animal();

// ❌ Cannot instantiate abstract class

Dog dog = Dog();

dog.makeSound();

// Output: Woof! Woof! dog.sleep();

 // Output: Sleeping... }
```

Task

You are building a **payment system** for an online store.

There are different types of payment methods:

- `CreditCard`
- `Paypal`
- `CashOnDelivery`

Each payment method should have a **pay** function, but the implementation differs for each one.

Assignment

Focus: Apply OOP principles (**Abstraction, Encapsulation, Inheritance, Polymorphism**)

Requirements

1. **Create a Patient class** with:
    - `id` (unique identifier)
    - `name`
    - `age`
    - `medicalHistory` (list of past diseases or treatments)
2. **Create a Doctor class** with:
    - `id`
    - `name`
    - `specialization`
    - `patients` (list of assigned patients)
3. **Create a Hospital class** that manages:
    - A list of patients
    - A list of doctors
    - Operations such as registering patients, hiring doctors, and assigning patients to doctors
4. **Make all attributes private** and provide safe getter/setter methods to access or update them.
5. **Add a class called Surgeon** that extends **Doctor** and has extra functionality:
    - Perform surgery for a patient
    - Keep track of the number of surgeries performed
6. **Each class should have a method** to display information about the object in a clear format (Polymorphism: same method name but different behavior per class).
7. In the **main()** function:
    - Add a few patients and doctors
    - Assign patients to doctors
    - Let a surgeon perform a surgery for a patient
    - Show all doctors and their assigned patients
    - Show details of all patients