Programming in Dart for Android & IOS App development: Part-2

Dart

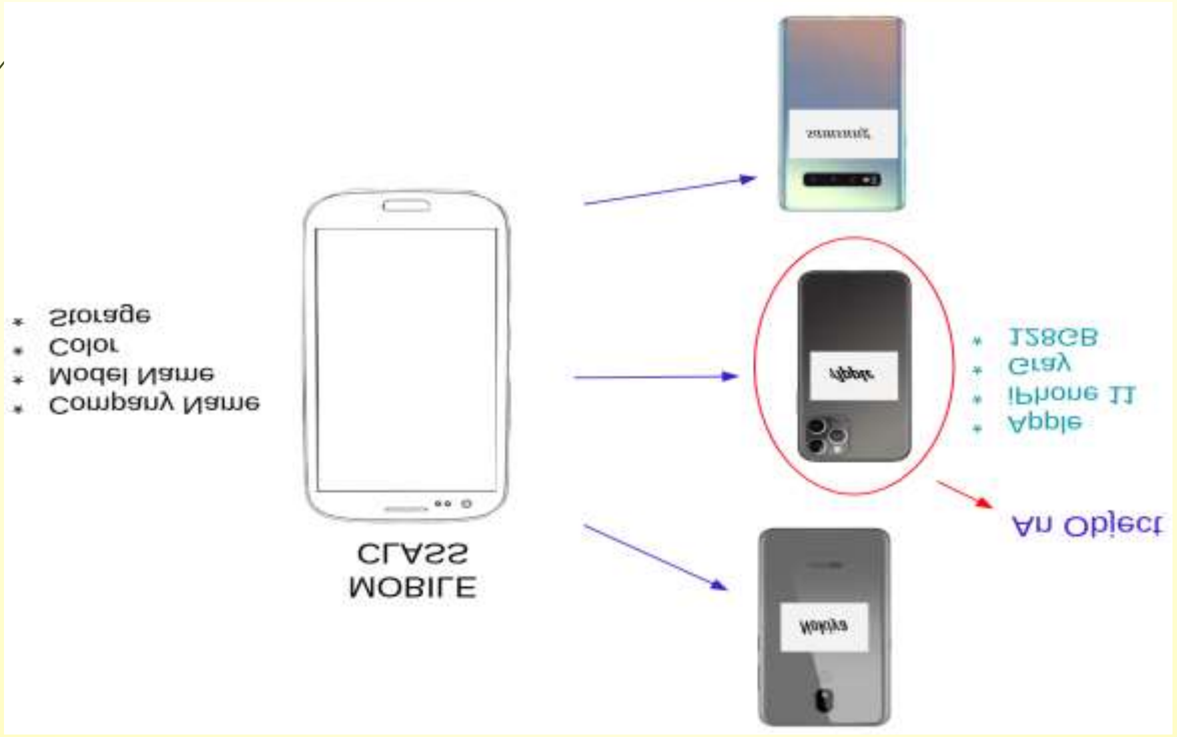
Md Bayazid Hossain
bh.190140@s.pust.ac.bd

Introduction to OOP

- Base OOP principles
  - **Encapsulation**
    - Put everything inside the box.
  - **Inheritance(Specialization)**
    - Add some extra feature with old one.
  - **Polymorphism(Generalization)**
    - In one various thing be included
  - **Abstraction**
    - Hiding internal details and showing only required features.

What is Class?

- Class is a blue print of an object or substance.
- Object is an example of that class.
  - If we write an esay then the Noun and Adjective is an property
  - And verb is the behaviour.



Class Declaration
Syntax for defining classes in Dart

Example of a simple class

```
class Person {
  String name;
  int age;

  // Constructor
  Person(this.name, this.age);
}
```

Object Creation Creating instances (objects) of a class

```
void main() {
  Person person1 = Person("John", 25);
  Person person2 = Person("Alice", 30);
}
```

What is constructor?

Constructor is a special type of method which is invocked/call when the object of that class being created.

-> Nonparameterized Constructor

-> Parameterized constructor

Access modifiers in Dart

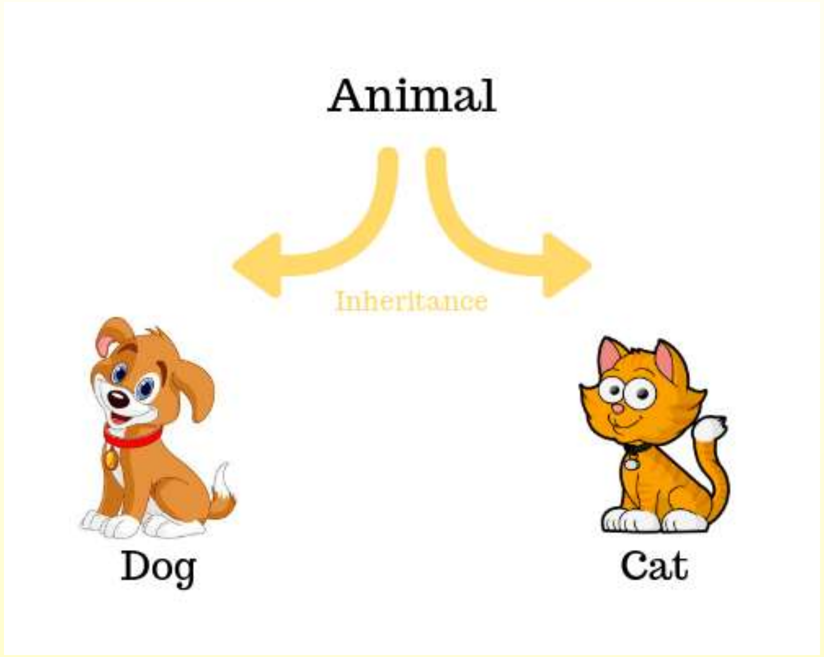
- Access modifiers, also known as access specifiers, are keywords in object-oriented programming languages that define the visibility or accessibility of class members (attributes, methods, etc.) from outside the class
  - **Public**
    - Accessable from anywhere.
  - **Private**
    - Accessable from only within the class
  - **Protected**
    - Accessable like public within the same package. And private to other package.

Private Code in Dart

```
class Circle {
  double _radius; // Private property
  // Getter
  double get radius => _radius;
  // Setter
  set radius(double value) {
    if (value > 0) {
      _radius = value;
    }
  }
}
```

Inheritance

- Inheritance is the process of acquiring feature of an existing class into a new class.
- Example:
  - A Cylinder have all the properties of circle and extra property is height.



Inheritance Code example

```
class Animal {
  void makeSound() {
    print("Generic Animal Sound");
  }
}

class Dog extends Animal {
  @override
  void makeSound() {
    print("Bark!");
  }
}
```

Abstraction in Dart

- **Defining a blueprint for subclasses**

```
abstract class Shape {
  void draw(); // Abstract method
}

class Circle extends Shape {
  @override
  void draw() {
    print("Drawing a circle");
  }
}
```

Polymorphism in Dart

- **Achieving polymorphism through method overriding Using abstract classes and interfaces.**

```
abstract class Shape {
  void draw();
}
class Circle implements Shape {
  @override
  void draw() {
    print("Drawing a circle");
  }
}
class Square implements Shape {
  @override
  void draw() {
    print("Drawing a square");
  }
}
```

Conclusion

- Recap of OOP principles in Dart
- Importance of OOP in building scalable and maintainable code
- Practical applications and best practices
- Feel free to customize and expand upon each section based on your audience's familiarity with OOP concepts and Dart programming. Add more examples, visuals, and interactive elements to enhance the learning experience.

This is it..

Thanks