Module 2

Flutter – Dart Programming

21. What is Inheritance .

When one class inherits the properties of another class is know as Inheritance.

22. Which Inheritance is not support by Dart ? Why ? What is advantage of Dart ?

Dart dose not support Multiple Inheritance . It mean that a class inherit single class . It cannot inherit two and more classes .

It promote reusability of the code and reduces redundant code . It help to design a program in better way . It make code simple , clear , save time and money on maintenance .

23. Difference between Inheritance and Encapsulation ? Difference between Inheritance and Abstractions ?

Inheritance -

Inheritance in the dart programming language is a fundamental object oriented

Programming (OOP) concept that make allows you to create a new class based on an existing class . The new class called subclass and derived class , inherits

the properties and behaviors (fields and methods) of the existing class , know as

super class and base class .This enable you to reuse code and create structure a program .

Encapsulation -

Encapsulation is one of the four fundamental concepts of object oriented programming language (OOP), and it’s essential in dart as well . Encapsulation refers to the practice of bundling the data (attributes and fields) and the method

(functions and behavior) that operator on that data into a single unit knows as a

Class .

Abstraction –

Abstraction is a concept that focuses on simplifying complex by breaking them down into smaller, manageable components while hiding the unnecessary details. It allows you to create abstract classes or interfaces that define a contract or set of methods that must be implemented by concrete classes. These abstract classes or interfaces provide a blueprint for how classes should behave without specifying the exact implementation.

24. Difference between Inheritance and Polymorphism .

Inheritance -

Inheritance in the dart programming language is a fundamental object oriented

Programming (OOP) concept that make allows you to create a new class based on an existing class . The new class called subclass and derived class , inherits

the properties and behaviors (fields and methods) of the existing class , know as

super class and base class .This enable you to reuse code and create structure a program

Polymorphism –

Polymorphism is a concept that allows objects of different classes to be treated a

Common super class. It enables you to write code that can work object of classes in a uniform way. This is often achieved through method overriding , where subclasses provide their own implementation of methods defined in the super class. Polymorphism is used to create flexible and extensible code. It allows you to write generic code that operated on a common interface or super class and can be used with any object that adheres to that inheritance.

25. Can we override Static method in Dart ?

In dart, you cannot override or hide static methods in the same way that you can override instance methods. Static methods are associated with the class itself, not with instance of the class, and they cannot be override by subclasses. In dart, if a subclass declares a static method with the same as a static method in the super class with the same name.

26. Can we overload static method in Dart ?

In dart, you cannot overload static methods in the same way you can overload instance methods. Method overloading involves defining multiple methods with the same name in a class but with different parameter lists. The Dart language dose not support method overloading based on the number or types of parameters, whether for static or instance methods.

27. Can a class implement more than one interface ? Can a class extend more than one class in Dart ?

Dart has n interface keyword. Instead, all classes are implicit interfaces. Use an abstract class as an interface. A class can implement multiple interfaces but only can extend a single class.

Dart dose not support multiple inheritance, which means a class con not extend more than one class.

28. Can an interface extend more than one interface in Dart ?

Dart don’t have a separate keyword for an interface, instead we can simply implement a class. Most of the rules are same as java. When implementing for all methods or make your calls an abstract one. One interface can implement multiple inheritance .

29. What will happen if a class implement tow interface and they both have a methods with same and signature ?

In Dart, if a class implements two interfaces, and both interface have a method with the same name and signature, it is not considered an issue or a conflict . Dart will allow the class to implement both interface without any problem because method names in interface are not considered as part of the class’s method signature .

This is known as interface-based multiple inheritance, and Dart handles it by allowing the class to implement both interface with methods that have the same name and signature . Here’s an example to illustrate this .

30. Can we pass an object of a subclass to a method expecting an object of the super class ? Are static members inherited to sub classes ?

Yes , In object – oriented programming (oop), you can pass an object of a sub-class to a method or function that expects an object of the super class . This is known as polymorphism and is a fundamental concept in inheritance . The method that expects the super class object can accept object of the subclass because the subclass relationship with the super class . This allows for greater flexibility and reusability in your code .

31. What happens if the parent and the child class have a field with same identifier ? Are constructors and initializers also inherited to sub class ?

If the parent class and the child class have a field with the same identifier, the means that the shadows the parent class field . This means that the child class field takes precedence over the parent class field within the scope of the child class . the child field essentially hides the parent class field with the same name.

In Dart , constructors are not inherited by subclasses . however , a sub class can a constructor form its super class using the super keyword . Initilizers, like constructors , are also not inherited by sub classes . When you initialize an object using a constructor , the initilizers of the parent class are executed first , followed by the initialiers of the child class.

32. How do you restrict a member of a class from inheriting by its sub classes ?

In object – oriented programming , you might encounter situations where want to restrict a member of a class form being inherited by its subclasses . in Dart , you can achieve this by making the member private , using an underscore (\_) before its name .

Private members are not accessible from outside the class , including sub- classes .

33. How do you implement multiple inheritance in Dart ?

Dart dose not support multiple inheritance in the traditional sense , where a class can directly inherit from more than one class . However , Dart provides a way to achieve similar behavior using a concept called mixins .

34. Can a class extend by itself in dart ?

In dart , a class can not directly extend itself . Attempting to declare a class that extends itself will result in a compilation error . This restriction is in place to prevent infinite recursion and ensure that class hierarchies are well-formed and maintainable .

35. How do you override a private method in dart ?

In Dart , private methods (and fields) are denoted by an underscore (\_) before their names . Private methods can not be accessed directly form outside the class , including in subclasses . Therefore , you can not override a private method in the traditional sense .

36. When to overload a method in Dart and when to on Dart class declaration ?

In Dart , method overloading and method overriding serve different purposes in the context of object – oriented programming .

37.What the order is of extends and implements keyword on Dart class declaration .

In dart , when declaring a class , the extends keyword is used to specify the super class (the class that the new class inherits from) , and the implements keyword is used to specify the interfaces that the class implements .

38. How do you prevent overriding a Dart method without using the final modifier ?

In Dart , if you want to prevent a method from being overridden by sub classes without using the final modifier , you can achieve this by making the method private . By prefixing the method name with an underscore \_ , you make it private to the current library . Private methods can not be accessed or overridden by subclasses outside of the same library .

39. What are the rules of method overriding in Dart ?

In Dart , method overriding allows a subclass to provide a specific implementation for a method that is already in its super class . Here are the rules for method overriding in Dart .

1. Method Signature
2. Keyboard @override
3. Visibility
4. Covariant Parameters (Dart 2.12 and above)

40. Difference between method overriding and overloading in Dart .

In Dart , method overriding and method overloading are two different concepts used to achieve polymorphism and handle different method signature . Let’s explore the differences between method overriding and overloading in Dart .

1. Method Overriding : Method overriding occurs when a sub class provides a specific implementation for a method that is already defined in its superclass . The overriding method in the subclass . It is used for achieving runtime polymorphism and is a fundamental concept in object – oriented programming .
2. Method Overloading : It allows a subclass to provide a specialized implementation of a method inherited from its superclass . This enables dynamic method invocation , where the appropriate method invocation , where the appropriate method implementation is determined at runtime based in the object’s actual type .

41. What happens when a class implements two interfaces and both declare field (variable) with same name ?

In Dart , if a class implements two interfaces and both interfaces declare a field (variable) with the same name , the implementing class is required to provide an implementation for that field . This can be done using a getter and setter method or by directly the field in the implementing class . The implementing by explicitly referencing the interface’s name followed by the field name .

42. Can a subclass instance method override a superclass static method ?

No , In Dart , a subclass instance method cannot override a superclass static method . Method overriding in Dart applies to instance methods , not static methods . When you define a static method in a superclass , it belongs to the class itself and not to any specific instance . Subclasses cannot override static method form their superclasses .

43. Can a subclass static method hide super class instance method ?

No , In Dart , a subclass static method cannot hide a superclass instance method . Static methods are associated with the class itself , not with instance of the class . They operate at the class level , not at the instance level . Instance methods , on the other hand , are associated with instances of the class and operate in the state of those instances .

44. Can a superclass access subclass member ?

No, in object – oriented programming , the superclass cannot asses member (field or methods) of a subclass directly . The principle of encapsulation , which is one of the core principles of object – oriented programming , dictates that the internal details of a class, including its private and protected member , should not be directly accessible from outside the class , including its superclass .

45. Difference between object – oriented and object based language ?

Object –Oriented Programming Language

1. Class and Object : Object – oriented programming language have support for classes and objects . Classes are user – defined data types that encapsulation data (attributes) and behaviors (methods) into a single unit .

Object are instances of classes .

1. Inheritance : OOP languages support inheritance, allowing one class(subclass/derived class) to inherit properties and behaviors from another class (superclass/base class) .Inheritance promotes code reusability and establishes a relationship between classes .
2. Encapsulation : OOP emphasizes encapsulation , where data (attributes) and methods that operate on the data are bundled together within a class . Access to data is controlled through getter and setter methods , allowing data hiding and protecting .
3. Polymorphism : Polymorphism enables objects of different classes to be treated as objects of a common superclass . This allows methods to be called on object without knowing their specific types . facilitating flexibility and extensibility in the code .
4. Abstraction : Abstraction allows developers to define abstract classes and interfaces , providing a blueprint for concrete classes . Abstract classes cannot be instantiated and can have methods that must be implemented by subclasses .

* Language like Java, C++, Python and Ruby are considered truly object – oriented because they support all these principles .