**Module – 2**

**Flutter - Dart Programing**

**21. What is inheritance?**

The capability of a class to derive properties and characteristics from another class is called **Inheritance**. Inheritance is one of the most important features of Object-Oriented Programming.

Inheritance is a feature or a process in which, new classes are created from the existing classes. The new class created is called “derived class” or “child class” and the existing class is known as the “base class” or “parent class”. The derived class now is said to be inherited from the base class.

**22. Which inheritance is not supported by Dart? Why? What is advantage of inheritance?**

Multiple Inheritance: This inheritance occurs when a class inherits more than one parent class. Dart doesn't support this. Multi-Level Inheritance: This inheritance occurs when a class inherits another child class.

**Advantages of Inheritance:**

The base class code will be already tested and debugged. As the existing code is reused, it leads to less development and maintenance costs. Inheritance makes the sub classes follow a standard interface. Inheritance helps to reduce code redundancy and supports code extensibility.

**23. Difference between inheritance and encapsulation. Difference between inheritance and abstraction.**

|  |  |
| --- | --- |
| **Inheritance** | **Encapsulation** |
| * Inheritance enables new classes to receive or inherit the properties and methods of existing classes. * It supports code reusability. * It allows us to do hierarchical classification of data. | * Encapsulation binds the data and the functions that operate on that data into a single unit. * It supports Data Hiding. * It keeps data safe from outside interference. |

|  |  |
| --- | --- |
| **Inheritance** | **Abstraction** |
| * Inheritance does provide full abstraction. * Using an interface we can active multiple inheritance. * We can not declare a member field in an interface. | * An abstract class doesn’t provide full abstraction. * Using abstract we can not achieve multiple inheritance. * We can declare a member field. |

**24. Difference between inheritance and polymorphism**

|  |  |
| --- | --- |
| **Inheritance** | **Polymorphism** |
| * It allows a class to use the properties and methods of a superclass. * Occurs in class level * It provides code reusability * Implementation of inheritance occurs in class level | * It allows an object to behave in multiple ways * Occurs in method level * It allows calling methods accordingly at compile time and runtime. * Implementation of polymorphism occurs in method level |

**25. Can we override static method in Dart?**

You can't override static methods. The two static methods you declared there are in fact two different static methods, not the same, overriden one. Answer for a different question, but related: Dart doesn't inherit static methods to derived classes.

**26. Can we overload static method in Dart?**

Overloading is also a feature of OOP languages like Java that is related to compile-time (or static) polymorphism. This feature allows different methods to have the same name, but different signatures, especially the number of input parameters and type of input parameters.

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

Your class can implement more than one interface, so the implements keyword is followed by a comma-separated list of the interfaces implemented by the class. By convention, the implements clause follows the extends clause, if there is one.

**Can a class extend more than one class in Dart**

In Dart, multi-level inheritance occurs when various classes inherit in a chain (i.e., one class extends a parent class, and another class extends the class that extended the parent 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 an interface, either provide implementation for all methods or make your calls an abstract one. One interface can implement multiple interface.

**29. What will happen if a class implements two interfaces and they both have a method with same name and signature?**

Interfaces only proscribe a method name and signature. If both interfaces have a method of exactly the same name and signature, the implementing class can implement both interface methods with a single concrete method.

**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, you can pass that because subclass and superclass are related to each other by Inheritance which provides IS-A property.  I mean Banana is a Fruit so you can pass banana if somebody expect fruit. Now there are scenario, where you can't do e.g. when subclass violates the Liskov Substitution principle i.e. you cannot pass a plastic banana to someone expecting fruit :-), The eat() function will throw exception.

**static members inherited to sub classes?**

The static keyword is used to create methods that will exist independently of any instances created for the class.

Static methods do not use any instance variables of any object of the class they are defined in. Static methods take all the data from parameters and compute something from those parameters, with no reference to variables.

We can inherit static methods in Java.

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

The reference holding the child class object reference will not be able to access the members (functions or variables) of the child class. This is because the parent reference variable can only access fields that are in the parent class.

**constructors and initializers also inherited to sub classes?**

No, Constructors and initializers (Static initializers and instance initializers) are not inherited to sub classes. But, they are executed while instantiating a sub class.

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

You can prevent a class from being subclassed by using the final keyword in the class's declaration. Similarly, you can prevent a method from being overridden by subclasses by declaring it as a final method. An abstract class can only be subclassed; it cannot be instantiated.

**33. How do you implement multiple inheritance in Dart?**

No, Dart does not support multiple implementation inheritance. Dart has interfaces, and like most other similar languages it has multiple interface inheritance. For implementation, there is only a single super-class chain that a class can inherit member implementations from.

**34. Can a class extend by itself in Dart?**

Dart allows single inheritance, which means a class can extend from a single parent. But a single parent can have multiple children, and each child can have its own children, building up a hierarchy many classes deep.

**35. How do you override a private method in Dart?**

1) Copy the package folder in your current code and change it as per your need.

2) Create a public method in library class and pass that private method in it. You can have the access of that private method as defined public method.

**36. When to overload a method in Dart and when to override it?**

Method Overriding in Dart

Overriding is done so that a child class can give its implementation to the method that is already provided by the parent class. In this case, the method in the parent class is called the overridden method and the method in the child class is called the overriding method.

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

The **extends always precedes** the implements keyword in any Java class declaration.

When the Java compiler compiles a class into bytecode, it must first look to a parent class because the underlying implementation of classes is to point to the bytecode of the parent class - which holds the relevant methods and fields.

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

There are three ways to stop method overriding in Java inheritance. Final, static, and private methods cannot be overridden by a subclass. If a method is declared final, static or private in base class, it cannot be overridden by the subclass. Here, we will see all three ways of preventing method overriding with help of suitable examples.

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

Rules for Method Overriding

The return type should be the same as in the method present in the superclass. The argument list should be the same as in the method present in the superclass. We cannot override a method if it is declared static or final. If we can't inherit a method, we can't override it.

**40. Difference between method overriding and overloading in Dart.**

|  |  |
| --- | --- |
| **Method Overriding** | **Method Overloading** |
| * It is possible only in derived classis. * The method must be a non-virtual or static method for overriding. * Also known as dynamic binding or late binding. * Used to implement run time polymorphism. * It helps to overwrite or change the existing functionalities. | * It is possible only same class. * Static method can be overloaded. * Also known as static binding or early binding. * Used to implement compile time polymorphism. * It helps to extend functionalities. |

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

If a type implements two interfaces, and each interface define a method that has identical signature, then in effect there is only one method, and they are not distinguishable. If, say, the two methods have conflicting return types, then it will be a compilation error. This is the general rule of inheritance, method overriding, hiding, and declarations, and applies also to possible conflicts not only between 2 inherited interface methods, but also an interface and a super class method, or even just conflicts due to type erasure of generics.

**43. Can a subclass static method hide superclass instance method?**

When a child class defines a static method with the same signature as a parent class's static method, **the child's method hides the parent class's method**. Method overriding refers to the same behavior involving instance methods.

Method hiding is **similar to method overriding** in terms of functionality. Overriding allows you to call methods based on the type of instance if you construct a method in a subclass with the same type and signature.

When static methods in a superclass and a subclass have the same type and signature, the **method in the subclass hides the method in the superclass**.

Static methods cannot be overridden because method overriding is predicated on dynamic binding at runtime, whereas static methods are bonded via static binding at compile time. As a result, **we cannot override static methods**.

**44. Can a superclass access subclass member?**

[**1.**](https://www.oreilly.com/library/view/java-a-beginners/9780071606325/ch7lev1sec16.html#ch7qus1) Does a superclass have access to the members of a subclass? Does a subclass have access to the members of a superclass?

No, a superclass has no knowledge of its subclasses. Yes, a subclass has access to all nonprivate members of its superclass.

[**2.**](https://www.oreilly.com/library/view/java-a-beginners/9780071606325/ch7lev1sec16.html#ch7qus2) Create a subclass of **TwoDShape** called **Circle**. Include an **area( )** method that computes the area of the circle and a constructor that uses **super** to initialize the **TwoDShape** portion.

**45. Difference between object oriented and object based language.**

|  |  |
| --- | --- |
| **Object oriented** | **Object Based** |
| * Object oriented language supports all the features of OOPs * Object oriented language doesn’t have built-in object. * Examples: c++,c#,java etc. | * Object based language doesn’t support all the feature of OOPs like polymorphism and inheritance. * Object based language has built-in object like java script has window object. * Examples : javascript, vb etc. |