**21) What is Inheritance?**

-> Inheritance is a fundamental concept in object-oriented programming (OOP) that allows a class (known as the "subclass" or "derived class") to inherit properties and behaviors from another class (known as the "superclass" or "base class"). The subclass can reuse and extend the attributes and methods of the superclass, which promotes code reuse and facilitates the creation of hierarchical relationships between classes.

**22) Which inheritance is not supported by Dart? Why? B3. What is advantage of inheritance?**

-> Multiple Inheritance is not supported by Dart. 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.

Advantage of inheritance: It promotes reusability of the code and reduces redundant code. It helps to design a program in a better way. It makes code simpler, cleaner and saves time and money on maintenance. It facilitates the creation of class libraries.

**23) Difference between inheritance and encapsulation. B5. Difference between inheritance and abstraction.**

**-> Inheritance and Encapsulation:**

Encapsulation is a mechanism that binds together code and the data it manipulates whereas Inheritance is the mechanism by which a class acquires the properties and methods of another class. An ATM binding together the different denominations of currency notes and all the operations required to withdraw cash is an example of Encapsulation. Classifying Vehicles as Car, Bike, Bus, Truck, etc. is an example of Inheritance.

**-> Inheritance and Abstraction:**

The main difference between abstraction and inheritance is that abstraction allows hiding the internal details and displaying only the functionality to the users, while inheritance allows using properties and methods of an already existing class.

**24) Difference between inheritance and polymorphism.**

**-> Inheritance:**

- Inheritance is one in which a new class is created (derived class) that inherits the features from the already existing class (Base class).

- It is basically applied to classes.

- Inheritance supports the concept of reusability and reduces code length in object-oriented programming.

- Inheritance can be single, hybrid, multiple, hierarchical and multilevel inheritance.

- It is used in pattern designing.

- **Example**: The class bike can be inherit from the class of two-wheel vehicles, which is turn could be a subclass of vehicles.

**-> Polymorphism:**

- Polymorphism is that which can be defined in multiple forms.

- It is basically applied to functions or methods.

- Polymorphism allows the object to decide which form of the function to implement at compile-time (overloading) as well as run-time (overriding).

- Whereas it can be compiled-time polymorphism (overload) as well as run-time polymorphism (overriding).

- It is also used in pattern designing.

- **Example:** The class bike can have method name set\_color(), which changes the bike’s color based on the name of color you have entered.

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

-> No, you cannot override static methods in Dart. In object-oriented programming, method overriding allows a subclass to provide a specific implementation for a method that is already defined in its superclass. This concept applies to instance methods, where a subclass can provide its own implementation of an inherited instance method.

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

-> No, you cannot overload static methods in Dart. Overloading refers to defining multiple methods in the same class with the same name but different parameter lists. In Dart, method overloading is based on the number and types of parameters, which allows you to have multiple methods with the same name but different parameter lists. However, this concept does not apply to static methods.

**27) Can a class implement more than one interface? B10. Can a class extend more than one class in Dart?**

-> Yes, a class in Dart can implement more than one interface. Dart supports multiple inheritance through the use of interfaces. In Dart, interfaces are implicitly defined through class declarations, and a class can implement multiple interfaces.

-> No, a class in Dart cannot extend more than one class. Dart does not support multiple inheritance in the traditional sense where a class can directly inherit from multiple parent classes. In Dart, a class can only have a single superclass from which it directly inherits. This is in contrast to some other programming languages that allow for multiple inheritance, where a class can inherit from multiple parent classes.

**28) Can an interface extend more than one interface in Dart?**

-> Yes, an interface can extend more than one interface in Dart. Dart supports multiple inheritance for interfaces, allowing you to create new interfaces by extending multiple existing interfaces.

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

-> If a class implements two interfaces and both of those interfaces have a method with the same name and signature, there will be no issues. Dart will not consider this a conflict or ambiguity. The class will effectively implement both methods, and there won't be any clashes or errors.

**30) Can we pass an object of a subclass to a method expecting an object of the super class? B14. Are static members inherited to sub classes?**

-> Yes, in most object-oriented programming languages, including Dart, you can pass an object of a subclass to a method that expects an object of the superclass. This is known as polymorphism and is a fundamental concept in object-oriented programming. When you pass a subclass object to a method that expects a superclass object, the method can work with the object as if it were an instance of the superclass. This is possible because a subclass inherits all the properties and behaviors of its superclass.

-> Static members are not inherited by subclasses in Dart. Inheritance in Dart refers to the process by which a subclass acquires the instance members (fields and methods) of its superclass. However, static members (static fields and static methods) are associated with the class itself, not with instances of the class, so they are not inherited by subclasses.

**31) What happens if the parent and the child class have a field with same identifier? B16. Are constructors and initializers also inherited to sub classes?**

-> If both the parent class and the child class have a field with the same identifier (name), the child class's field will effectively hide the parent class's field. This is known as variable hiding.

-> Constructors and initializers are not inherited by subclasses in Dart. When a subclass is created, it doesn't automatically inherit the constructors and initializers from its superclass.

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

-> In Dart, you can restrict a member of a class from being inherited by its subclasses by marking the member with the ‘final’ modifier. When a member is declared as ‘final’, it means that the member's implementation cannot be overridden or changed by subclasses. This effectively prevents any further modification or inheritance of that specific member.

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

-> Dart does not support traditional multiple inheritance, where a class directly inherits from multiple parent classes. However, Dart provides a feature called mixins, which can be used to achieve some of the benefits of multiple inheritance in a more controlled and modular way.

Mixins are a way to reuse code in multiple class hierarchies without using full multiple inheritance. A mixin is a class that provides certain behaviors that can be added to other classes using the ‘with’ keyword.

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

-> No, a class cannot directly extend itself in Dart or in any object-oriented programming language. Inheritance establishes a relationship between a subclass and its superclass, creating a hierarchy. This hierarchy requires that a class must inherit from another class, and that other class becomes its superclass. It's not possible for a class to be its own superclass or to directly extend itself.

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

-> In Dart, private methods are denoted by prefixing an underscore (‘\_’) before the method name. Private methods are not meant to be accessed outside of the class they are defined in. Because of this encapsulation, you cannot override private methods in subclasses. Private methods are not visible or accessible to subclasses, and thus they cannot be overridden.

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

**-> Method overloading:**

Method overloading refers to defining multiple methods in a class with the same name but different parameter lists. Dart does not support traditional method overloading like some other programming languages do, where you can have multiple methods with different parameter types or counts. In Dart, if you define multiple methods with the same name, they must have different parameter lists (different number of parameters or different parameter types).

**-> Method overriding:**

Method overriding occurs when a subclass provides a specific implementation for a method that's already defined in its superclass. It allows you to replace or extend the behavior of a method in a subclass while keeping the same method signature (name and parameter list).

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

-> In Dart, the order of the ‘extends’ and ‘implements’ keywords in a class declaration follows a specific pattern:

i) First, the ‘extends’ keyword is used to indicate that a class is extending another class, creating a superclass-subclass relationship. The class being extended is the superclass, and the class that's extending it is the subclass.

ii) After the ‘extends’ keyword, if there are any, the ‘implements’ keyword is used to indicate that a class is implementing one or more interfaces. An interface specifies a contract that a class must adhere to by providing implementations for the methods declared in the interface.

**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 subclasses without using the ‘final’ modifier, you can achieve this by marking the method as ‘static’. Static methods are associated with the class itself, rather than with instances of the class. They cannot be overridden by subclasses because they are not part of the inheritance hierarchy.

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

**-> Rules of method overriding:**

- The method which we want to override can only be written in the subclass, not in the same class.

- 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 overriding occurs when a subclass provides a specific implementation for a method that's already defined in its superclass.

- The overriding method in the subclass must have the exact same method signature (name, parameters, and return type) as the method in the superclass.

- You use the ‘@override’ annotation before the method in the subclass to indicate that it's intended to override a method in the superclass.

- Overriding is related to inheritance and is used to customize or extend the behavior of methods inherited from the superclass.

- The access modifier of the overriding method must be at least as accessible as the method in the superclass.

**-> Method overloading:**

- Method overloading refers to defining multiple methods in a class with the same name but different parameter lists.

- Overloaded methods must have different parameter lists (different number of parameters or different parameter types).

- Overloading is not directly related to inheritance. It's used to provide multiple methods with similar behavior but differing parameter requirements.

- The appropriate overloaded method is selected at compile time based on the arguments used when invoking the method.

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

-> When a class implements two interfaces and both interfaces declare a field (variable) with the same name, there is no issue or conflict. Dart allows a class to implement multiple interfaces that may declare fields with the same name because fields do not conflict in the same way that methods might.

**42) Can a subclass instance method override a superclass static method?**

-> No, a subclass instance method cannot override a superclass static method in Dart. Method overriding involves providing a specific implementation for a method in a subclass that's already defined in its superclass. However, static methods are associated with the class itself, not with instances of the class, and they cannot be overridden.

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

-> Yes, a subclass static method can hide a superclass instance method in Dart. This occurs because static methods are associated with the class itself, and they don't participate in the normal inheritance and method overriding mechanisms that apply to instance methods.

When a subclass defines a static method with the same name as an instance method in its superclass, it effectively hides the superclass's instance method, and the two methods are treated as unrelated.

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

-> No, a superclass cannot directly access members (fields or methods) of its subclasses in Dart. The relationship between a superclass and its subclasses is one of inheritance, where subclasses inherit the members and behaviors of the superclass, but the superclass does not have direct access to the members of its subclasses.

Inheritance allows subclasses to access and potentially override members from their superclass, but the reverse is not true. Subclasses have knowledge of their superclass's members, but the superclass is not aware of the members added or overridden in its subclasses.

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

**-> Object oriented language:**

- An object-oriented language is a programming language that supports all the key principles of object-oriented programming (OOP), including encapsulation, inheritance, abstraction, and polymorphism.

- Object-oriented languages allow you to define classes and create objects (instances) from those classes. Objects encapsulate data (attributes) and behavior (methods).

- Object-oriented languages support inheritance, allowing classes to be derived from other classes, enabling code reuse and hierarchy.

- Object-oriented languages support polymorphism, which allows objects of different classes to be treated as objects of a common superclass through interfaces or inheritance.

- **Example**: Java, C++, C#, Python.

**-> Object based language:**

- An object-based language is a programming language that supports most of the features of object-oriented programming but might lack certain features like inheritance or polymorphism.

- Object-based languages allow you to define classes and create objects, similar to object-oriented languages. They also encapsulate data and behavior.

- Object-based languages may support inheritance to some extent, but not all types of inheritance might be available (e.g., multiple inheritance).

- Object-based languages may support polymorphism but might have limitations or restrictions on how it can be implemented.

- **Example**: JavaScript (in its earlier versions), VBScript.