1. **What is inheritance?**

* Inheritance is a fundamental principle in object-oriented programming that allows a new class (called a derived or subclass) to inherit properties and behaviors (methods and fields) from an existing class (called a base or superclass).

1. **Which inheritance is not supported by Dart? Why?**

* Dart doesn't support multiple inheritance. This is to maintain code simplicity and avoid the "diamond problem" where ambiguity arises if a class inherits from two classes that have a method or attribute with the same name.

1. **What is advantage of inheritance?**

* The primary advantage of inheritance is code reusability. It allows for the creation of a hierarchical relationship among classes, enabling subclasses to inherit attributes and methods from their super classes, reducing redundancy and promoting a more organized and maintainable codebase.

1. **Difference between inheritance and encapsulation.**

* Inheritance enables a new class to acquire properties and behaviors from an existing class.
* Encapsulation is the mechanism of bundling data (attributes) and methods that operate on the data into a single unit (class) and restricting access to some components, ensuring data integrity.

1. **Difference between inheritance and abstraction.**

* Inheritance allows a subclass to inherit properties and behaviors from a superclass.
* Abstraction is a concept that focuses on hiding implementation details while providing a clear interface for working with functionality.

1. **Difference between inheritance and polymorphism.**

* Inheritance is a mechanism where one class inherits properties and behaviors from another class.
* Polymorphism refers to the ability to present the same interface for different data types. It can be achieved through inheritance and interface implementations.

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

* No, static methods cannot be overridden in Dart as they are tied to the class itself rather than instances. Subclasses cannot provide a new implementation for a static method.

1. **Can a class implement more than one interface?**

* Yes, Dart supports multiple interface implementations for a class.

1. **Can a class extend more than one class in Dart?**

* No, Dart does not support multiple inheritance for classes to avoid complexities arising from conflicts and ambiguity that can occur with multiple parent classes.

1. **Can an interface extend more than one interface in Dart?**

* Yes, Dart allows interfaces to extend multiple interfaces, facilitating better code organization and structure.

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

* In Dart, the class implementing these interfaces will have to provide an implementation for the method that satisfies the contracts of both interfaces. There's no conflict resolution as in languages with default methods in interfaces.

1. **Can we pass an object of a subclass to a method expecting an object of the super class?**

* Yes, Dart supports substitutability, allowing passing an object of a subclass where a superclass object is expected. This is part of Dart's support for subtype polymorphism.

1. **What happens if the parent and the child class have a field with same identifier?**

* In Dart, if both the parent and the child class have a field with the same identifier, the child class's field will hide or override the parent class's field. Accessing the field from an instance of the child class will refer to the child class's field.

1. **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 making that member private. Private members in Dart, denoted by an underscore \_ prefix, are only accessible within the same library and cannot be inherited.

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

* Dart does not directly support multiple inheritance for classes. However, Dart provides a mechanism called mixins, which allow code reuse by composing classes in a different way than traditional inheritance.

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

* No, a class cannot directly extend itself in Dart. Such self-referencing would create a circular dependency and is not allowed.

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

* In Dart, you cannot directly override a private method from a superclass in a subclass. Private methods are not accessible outside the class where they are defined, including in subclasses.

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

* **Overloading** in Dart is not supported, whether it's for static or instance methods. Therefore, there's no scenario for method overloading in Dart.
* **Overriding** occurs when a subclass provides a specific implementation for a method that is already defined in its superclass. It's used when you want to modify the behavior of a method inherited from a superclass.

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

* In Dart, the order of keywords in a class declaration is typically class ClassName extends SuperclassName implements InterfaceName.
* extends is used to specify the superclass.
* implements is used to specify interfaces that the class will adhere to.
* Please note that Dart's approach to inheritance and related concepts might differ from other programming languages, and understanding these nuances will help in writing effective and clean Dart code.

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

* To prevent method overriding in Dart without using the final modifier, you can implement the method in the superclass as a private method (prefixed with an underscore \_). Private methods cannot be accessed or overridden by subclasses outside of the defining class.

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

* In Dart, the rules for method overriding are:
* The method in the superclass and the overriding method in the subclass must have the same name, parameters, and return type.
* The overriding method should use the @override annotation (though it's optional) to ensure it's intended to override a superclass method.
* The access level of the overriding method should be the same or less restrictive than the superclass method.

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

* Method Overriding: Occurs when a subclass provides a specific implementation for a method already defined in its superclass.
* Method Overloading: Is not supported in Dart. Method overloading refers to having multiple methods with the same name but different parameter lists within the same class or scope, enabling different ways to call the same method name.

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

* If a class implements two interfaces that declare a field with the same name, the class must provide an implementation for that field, satisfying both interface contracts. Dart does not have a conflict resolution mechanism for field names in interfaces.

1. **Can a subclass instance method override a superclass static method?**

* No

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

* No

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

* No

1. **Difference between object oriented and object-based language.**

* Object-oriented languages support concepts like inheritance, encapsulation, polymorphism, and abstraction. Examples include Java, C++, and Dart. They allow defining classes, objects, and relationships between them.
* Object-based languages are those that support objects but might lack some features of full-fledged object-oriented languages, like inheritance or polymorphism. JavaScript is an example of an object-based language.