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

**Ans :** Multiple inheritance is not supported by Dart.

One reason is that multiple inheritance can lead to ambiguity. For example, if a class inherits from two classes that both have a method with the same name, which method should be called?

Another reason is that multiple inheritance can make code more difficult to understand and maintain.

When a class inherits from multiple classes, it can be difficult to track which properties and methods it inherits from which class.

Finally, multiple inheritance is not necessary in Dart.

Hybrid inheritance is not supported by Dart.

The main reason why hybrid inheritance is not supported by Dart is because it can lead to ambiguity and complexity.

For example, if a class inherits from two base classes, and both base classes have a method with the same name, then which method is called when the method is invoked on the derived class?

Another reason why hybrid inheritance is not supported by Dart is because it can make code more difficult to understand and maintain. With hybrid inheritance, it can be difficult to track which methods are inherited from which base classes, and how those methods interact with each other.

Inheritance Advantage :

Code reuse: Inheritance allows you to reuse code by creating new classes that inherit from existing classes. This can save you a lot of time and effort, and can also help to improve the quality of your code by making it more consistent and easier to maintain.

Modularity: Inheritance can help you to create more modular code. Modular code is easier to understand, test, and maintain.

Polymorphism: Inheritance is a key part of polymorphism, which is the ability of objects to take on different forms and behaviors. Polymorphism is one of the most powerful features of object-oriented programming (OOP).

Extensibility: Inheritance makes it easy to extend the functionality of your code without having to rewrite existing code. For example, you can create a new class that inherits from an existing class and adds new methods or properties.

Structuring Code: Inheritance allows you to create a hierarchy of classes, which can help

organize and structure your code in a more logical and hierarchical manner. This makes it

easier to understand and maintain large codebases.

Abstraction: Inheritance supports the concept of abstraction by allowing you to define a

common interface (or abstract class) that specifies the methods and attributes that

subclasses should implement. This enforces a contract and helps in designing consistent

and well-structured code. Abstraction in Dart allows you to define abstract classes with abstract methods, which serve as blueprints for concrete classes.

Reduced Redundancy: Inheritance can help reduce redundancy in your code by

encapsulating common functionality in a single location (the superclass), which can be

reused by multiple subclasses. This not only saves development time but also minimizes

the potential for bugs.