Q-21) What is inheritance?

Ans: inheritance allows a class to inherit properties and methods from another class, enabling code reuse and establishing a hierarchy among classes through the use of the extends keyword.

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

Ans: Dart does not support **multiple inheritance**, where a class can inherit from more than one superclass, to avoid complexities like the "diamond problem," which causes ambiguity in method resolution when multiple super classes define the same method. Additionally, Dart does not support **hybrid inheritance**, which combines multiple inheritance and hierarchical inheritance, as it can lead to similar complications and increased complexity in the class hierarchy.

- ❖ **Advantages of Inheritance:**
  - **Code Reusability:** Inheritance allows classes to reuse methods and properties from other classes, reducing redundancy and improving maintainability.
  - **Hierarchical Structure:** It establishes a clear relationship among classes, making code organization and management more intuitive.
  - **Polymorphism:** Inheritance supports polymorphism, enabling objects of different classes to be treated as instances of a common superclass, which enhances flexibility in code design.

Q-23) Difference between inheritance and encapsulation. B5. Difference between inheritance and abstraction.

Ans:

- ❖ **Difference between inheritance and encapsulation:**

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

- ❖ **Difference between Inheritence and abstraction:**

Inheritance	Abstaraction
Mechanism to derive a class from another class.	Hides complex implementation details, exposing only essential features.
Achieves reusability and class hierarchy.	Simplifies complexity for users.
Created using the extends keyword.	Implemented with abstract classes or interfaces.

Q-24) Difference between inheritance and polymorphism.

Inheritence	Polymorphism
Inheritance is one in which a new class is created (derived class) that inherits the features from the already existing class(Base class).	Whereas polymorphism is that which can be defined in multiple forms.
It is basically applied to classes.	Whereas it is basically applied to functions or methods.
It is used in pattern designing.	While it is also used in pattern designing.

Q-25) Can we override static method in Dart?

Ans: No, you cannot override static methods in Dart. Static methods belong to the class itself, not to instances, which means they cannot be polymorphic. Instead, a static method in a subclass can hide a static method of the superclass, but it does not override it. When a static method with the same name is defined in a subclass, it hides the superclass method, and both methods can be called independently.

Q-26) Can we overload static method in Dart?

Ans: Yes, you can overload static methods in Dart. Method overloading allows multiple methods with the same name to exist within the same class, as long as they have different parameters (like differing numbers or types). For example, you can have one static method with an `int` parameter and another with a `String` and an `int` parameter, enabling you to call them based on the arguments provided.

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

Ans:

- **Can a class implement more than one interface?**
  - Yes, a class can implement more than one interface in Dart. When a class implements multiple interfaces, it must provide implementations for all the methods defined in those interfaces. This allows for flexible design and promotes code reusability.

- **Can a class extend more than one class in Dart?**
  - No, a class **cannot extend more than one class** in Dart. Dart supports **single inheritance**, meaning a class can only have one direct superclass. However, a class can implement multiple interfaces, allowing for some level of multiple inheritance behaviour. This design choice helps avoid complexity and ambiguity associated with multiple inheritance, such as the "diamond problem."

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

Ans: Yes, an interface can **extend more than one interface** in Dart. This allows the new interface to inherit method signatures from multiple interfaces, enabling a class that implements it to provide implementations for all inherited methods. This feature promotes modularity and code reuse by allowing interfaces to combine behaviors from different sources.

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

Ans: If a class implements two interfaces in Dart that have a method with the same name and signature, the class must provide a single implementation for that method. This eliminates ambiguity, as the method signature is identical in both interfaces. The implementing class will then use this single implementation when the method is called.

Q-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?

Ans:

- **Can we pass an object of a subclass to a method expecting an object of the super class?**
  - Yes, you can pass an object of a subclass to a method that expects an object of the superclass in Dart. This is possible due to polymorphism, where a subclass object is treated as an instance of its superclass. This allows for more flexible and reusable code, as methods can operate on a superclass type while still using subclass objects.

- **Are Static Members Inherited to Subclasses?**
  - Static members (methods and properties) are not inherited in the same way as instance members. While a subclass can access static members of its superclass, it does not inherit them. Instead, the static members belong to the class itself. If a subclass defines a static member with the same name, it hides the superclass's static member, but they are considered distinct.

Q-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?

Ans:

- **What Happens if the Parent and the Child Class Have a Field with the Same Identifier?**
  - If the parent and child class have a field with the same identifier, the child class shadows the parent's field. This means that the child class's field will be used when accessed from an instance of the child class. However, the parent's field can still be accessed using the super keyword.

❖ **Are Constructors and Initializers Also Inherited to Subclasses?**

- No, constructors and initializers are not inherited by subclasses in Dart. Each class must define its own constructors. However, a subclass can call the constructor of its superclass using the super keyword to initialize inherited properties. This allows for custom initialization while still leveraging the parent class's constructor.

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

Ans: To restrict a member of a class from being inherited by its subclasses in Dart, you can make the member private by prefixing its name with an underscore (`_`). This makes the member accessible only within the defining class, preventing subclasses from accessing it. For example, if a class has a private field `_privateField`, it cannot be accessed or inherited by any subclass, ensuring encapsulation.

Q-33) How do you implement multiple inheritance in Dart?

Ans: Dart does not support multiple inheritance directly, but you can achieve similar functionality using mixins. A mixin allows you to add methods and properties from multiple classes to a single class without extending them. You define a mixin with the `mixin` keyword and use the `with` keyword in the class definition to include multiple mixins, enabling code reuse and flexibility while maintaining Dart's single inheritance model.

Q-34) Can a class extend by itself in Dart?

Ans: No, a class cannot extend itself in Dart. Attempting to do so would create a recursive situation that leads to infinite recursion and ambiguity in method resolution. Dart enforces this restriction to maintain a clear and manageable inheritance structure. Instead, you can use mixins or abstract classes to share functionality without self-referencing.

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

Ans: In Dart, you **cannot override a private method** defined in a superclass. Private methods, indicated by a leading underscore (`_`), are only accessible within the library where they are declared. Therefore, if a method is private in a superclass, it cannot be accessed or overridden in subclasses from different libraries. Subclasses can only call private methods indirectly through public methods of the superclass.

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

Ans:

- **When to Overload a Method in Dart:**
  - You should overload a method when you want to provide multiple versions of the same method name that differ in the number or type of parameters. This is useful for enhancing readability and flexibility, allowing the same operation to handle different types of input without changing the method name.

- **When to Override a Method in Dart:**
  - You should override a method when you need to provide a specific implementation of a method that is already defined in a superclass. This is typically done to modify or extend the behavior of the inherited method, allowing polymorphism where a subclass can define its own version of the method.

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

Ans: In Dart class declarations, the order of the `extends` and `implements` keywords is important:

1. extends: This keyword comes first to specify the superclass from which the class inherits.

2. implements: After `extends`, you use `implements` to list one or more interfaces that the class will implement.

For example, `class Child extends Parent implements InterfaceA, InterfaceB` is the correct syntax. This order must be followed for proper class declaration in Dart.

Q-38) How do you prevent overriding a Dart method without using the final modifier?

Ans: In Dart, to prevent a method from being overridden without using the `final` modifier, you can use the `@protected` annotation from the `meta` package. This annotation indicates that the method should not be overridden outside the class or its subclasses. While it serves as a guideline and communicates intent, it does not enforce non-overriding behavior at the language level. Therefore, developers must adhere to this convention to avoid overriding protected methods.

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

Ans: The rules of method overriding in Dart are as follows:

1. Same Signature: The overriding method must have the same name, return type, and parameters as the method in the superclass.

2. Access Level: The access level of the overriding method must be the same or less restrictive than that of the superclass method.

3. Use of `@override` Annotation: While optional, using the `@override` annotation is recommended for clarity and to enable compile-time checks.

4. Final Methods: If a method is marked as `final` in the superclass, it cannot be overridden in the subclass.

These rules ensure consistency and proper behavior in the inheritance hierarchy.

Q-40) Difference between method overriding and overloading in Dart.

Ans:

Overriding	Overloading
Method overriding is a run-time polymorphism.	Method overloading is a compile-time polymorphism.
It is performed in two classes with inheritance relationships.	It occurs within the class.
Method overriding always needs inheritance.	Method overloading may or may not require inheritance.
In method overriding, methods must have the same name and same signature.	In method overloading, methods must have the same name and different signatures.

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

Ans: When a class implements two interfaces in Dart that both declare a field (variable) with the same name, the class does not inherit those fields directly. Instead, it must define its own field with a different name or provide an implementation that resolves the ambiguity. Attempting to access the conflicting fields directly will result in a compile-time error, as the compiler cannot determine which field to reference. Thus, the implementing class effectively has to manage this naming conflict.

Q-42) Can a subclass instance method override a superclass static method?

Ans: No, a subclass instance method **cannot override** a superclass static method in Dart. Static methods belong to the class itself, not to instances, and thus they do not participate in method overriding. If a subclass defines a static method with the same name as a static method in the superclass, it **hides** the superclass method instead of overriding it. This means the superclass static method can still be accessed using the superclass name, but it cannot be overridden by an instance method in the subclass.

Q-43) Can a subclass static method hide superclass instance method?

Ans: No, a subclass static method **cannot hide** a superclass instance method in Dart. Static methods belong to the class itself, while instance methods are tied to instances of the class. If a subclass defines a static method with the same name as an instance method in the superclass, both methods exist independently and do not affect each other. The static method can only be called on the subclass, whereas the instance method must be accessed through an instance of the superclass. Thus, there is no hiding of the superclass instance method.

Q-44) Can a superclass access subclass member?

Ans: No, a superclass cannot access subclass members in Dart. The superclass is not aware of the specific members defined in its subclasses, as inheritance is a one-way relationship where only subclasses inherit from super classes. The superclass can only access its own methods and properties. Even when a subclass instance is passed to a superclass method, the superclass can only call its own methods and cannot reference any subclass-specific members.

Q-45) Difference between object oriented and object based language.

Ans:

Object Oriented	Object Based
Object Oriented Languages supports all the features of Oops including inheritance and polymorphism.	Object based languages supports the usage of object and encapsulation.
They support built-in objects.	Object based languages does not supports built-in objects.
C#, Java, VB. Net are the examples of object oriented languages.	Javascript, VB are the examples of object bases languages.