Java 8 Diamond Problem - Solution

Solution to the Diamond Problem in Java 8

The diagram and question relate to the Diamond Problem in Java, particularly in the context of Java SE 8's support for behavioral multiple inheritance via interfaces.

(i) When the type D is a class:

If D is a class that inherits from classes B and C, and both B and C inherit from an abstract class A with a method method(), Java does not allow this because Java does not support multiple inheritance of classes. This restriction avoids the Diamond Problem involving state and implementation.

To resolve this, developers can use:

- Composition (i.e., use instances of B and C inside D)
- Inherit from only one class (B or C) and implement methods from the other

Reason: To avoid ambiguity in inherited method implementation.

(ii) When the type D is an interface:

Java 8 introduced default methods in interfaces, enabling behavioral multiple inheritance.

Suppose:

- A is an interface with a default method method()
- B and C are interfaces that both extend A (and may override method())
- D is an interface that extends both B and C

Java 8 Diamond Problem - Solution

Java 8 handles this scenario as follows:

- If both B and C override method(), D must override it to resolve ambiguity.

- If only one of B or C overrides it, Java uses that version.

- If neither overrides it and only A provides a default, D inherits it directly.

Java resolves such ambiguities at compile-time by enforcing explicit overriding of conflicting default methods.

Summary:

Scenario	Java's Handling
D is a class	Not allowed. Java disallows multiple class inheritance to avoid ambiguity
D is an interface	Allowed. Java resolves conflicts via explicit overriding of default methods