Java OOPs - Abstraction (Interview Notes)

Definition

Abstraction is the process of hiding internal implementation details and showing only essential features to the user.

Why Abstraction?

- Reduces complexity
- Increases code reusability
- Secures internal details
- Provides a common interface

Real-Time Example 1: ATM Machine

You enter PIN and amount, but the backend logic is hidden. Shows only essential interface.

Real-Time Example 2: Google Maps

User sees route/time; hidden are GPS algorithms and location services.

Java Implementation: Abstract Class

```
abstract class Vehicle {
    abstract void start();
    void fuel() {
        System.out.println("Fueling vehicle...");
    }
}
class Car extends Vehicle {
    void start() {
        System.out.println("Car starts with key");
    }
```

}

Chaining Interview Questions

Q1. Difference between abstraction and encapsulation?

- Abstraction: Hides implementation; Encapsulation: Hides internal state.

Q2. Can abstraction be done without abstract classes?

- Yes, via interfaces.

Q3. Can we instantiate abstract class?

- No.

Q4. Can abstract class have constructor?

- Yes.

Q5. Abstract class vs Interface?

- Abstract class: base + logic; Interface: pure contract.

Q6. Can abstract class have all concrete methods?

- Yes, but not useful.

Q7. Can abstract class implement interface?

- Yes.

Q8. What if class doesn't implement all abstract methods?

- Declare it abstract.

In Your Projects

API logic in fetchRecipeData() was abstracted away. Components only used the method, unaware of how the

Java OOPs - Abstraction (Interview Notes)

API works.