1Z0-808 Exam Topic Reviewer

TopicId: 1022
Topic: Abstract Classes and Interfaces

August 5, 2025

Abstraction: Hiding the Details

Alright team, let's talk about Abstraction. This is the OOP principle of hiding complex implementation details and showing only the necessary features of an object. In Java, we achieve this primarily through two tools: Abstract Classes and Interfaces. The exam will expect you to know exactly when and why to use each, especially with the changes introduced in Java 8.

0.1 Abstract Classes: The Partial Blueprint

An abstract class is a class that cannot be instantiated. Think of it as an incomplete template that provides common features for a group of subclasses.

- It's declared with the `abstract` keyword.
- It can contain both **abstract methods** (methods without a body, ending in a semicolon) and **concrete methods** (regular methods with a body).
- If a class contains even one abstract method, the class itself *must* be declared abstract.
- It can have instance variables, static variables, and constructors. The constructors are called by subclasses using `super()`.
- A concrete class that `extends` an abstract class **must** provide an implementation for all inherited abstract methods.

```
public abstract class Vehicle {
    private int speed;
    public Vehicle() { this.speed = 0; }
    public abstract void startEngine(); // Abstract method - no body
    public void stop() { // Concrete method
        System.out.println("Vehicle stopped.");
    }
}
public class Car extends Vehicle {
    @Override
    public void startEngine() { // Must implement this
        System.out.println("Car engine started.");
    }
}
```

0.2 Interfaces: The Pure Contract

An interface is a reference type that is a collection of abstract methods and constants. It defines a "contract" of behaviors that a class must adhere to if it `implements` the interface.

- A class can **implement multiple interfaces**, which is how Java achieves a form of multiple inheritance (of type/behavior).
- Before Java 8, all methods were implicitly `public abstract`.

- All variables are implicitly `public static final` (they are constants).

0.3 Java 8 Interfaces: A Game Changer

The 1Z0-808 exam will definitely test you on the new features added to interfaces in Java 8.

- `default` **Methods:** These are methods with a concrete implementation directly in the interface. They allow you to add new methods to an interface without breaking existing implementing classes. A class can override a default method if it needs to.
- `static` **Methods:** These are also methods with implementation, but they belong to the interface itself, not to the implementing class. You must call them using the interface name, e.g., `MyInterface.myStaticMethod()`.

```
public interface Flyable {
    String PLANET = "Earth"; // public static final
    void fly(); // public abstract
    default void land() { // default method
        System.out.println("Landing on " + PLANET);
    }
    static int getAltitude() { // static method
        return 10000;
    }
}
```

Abstract Class vs. Interface: The Showdown

Memorize this table. It's a goldmine for exam questions.

Feature	**Abstract Cl**
Multiple Inheritance	No (extends c
Variables final	Instance public static f
Constructors	Yes
Methods Abstract, default, static	Abstract
Usage	To share com

Key Takeaways for the 1Z0-808 Exam

- You cannot create an instance of an abstract class or an interface.
- A concrete class must implement all abstract methods from its superclass and interfaces.
- Know the Java 8 interface changes: `default` and `static` methods are fair game.
- Choose an abstract class when you want to share code among closely related classes. Choose an interface when you want to define a role that disparate

classes can play.