1Z0-808 Exam Topic Reviewer

TopicId: 1023
Topic: The 'final' Keyword

August 5, 2025

The `final` Keyword: No More Changes

The `final` keyword in Java is a modifier used to restrict the user. It can be applied to variables, methods, and classes. Its core meaning is "this cannot be changed," but what "this" refers to is key. The exam will test your precise understanding of its three different contexts.

0.1 `final` Variables: Creating Constants

A `final` variable can only be assigned a value once.

- **Initialization:** A `final` instance variable must be initialized by the time the constructor completes. This can be done at declaration, in an instance initializer block, or in the constructor itself. A `final static` variable must be initialized at declaration or in a static block.
- **Final Primitives:** The value itself is constant.

  ```
  final int MAX_USERS = 100;
  // MAX_USERS = 101; // COMPILE ERROR
  ```
- **Final Object References (CRITICAL EXAM TRAP):** This is the most misunderstood aspect. When a reference variable is declared `final`, it means the **reference cannot be changed** to point to a different object. However, the **state of the object it points to can be changed**.

```
final StringBuilder sb = new StringBuilder("Hello");
// sb = new StringBuilder("World"); // COMPILE ERROR: Cannot reassign final varia

// But this is perfectly legal!
sb.append(" World"); // Modifies the internal state of the StringBuilder object.
System.out.println(sb); // Prints "Hello World"
```

Think of it as a constant pointer to a mutable object.

0.2 `final` Methods: Preventing Overrides

When you declare a method as `final`, you are forbidding any subclass from overriding it.

- **Purpose:** To enforce a specific implementation that is critical to the class's function and should not be altered by subclasses.
- **Rule:** A subclass cannot have a method with the same signature as a final method in its superclass.

```
class SuperClass {
    public final String getVersion() {
        return "1.0";
    }
}
class SubClass extends SuperClass {
    // public String getVersion() { ... } // COMPILE ERROR: Cannot override final
```

```
}
```

Note: `private` methods are implicitly `final` because they are not visible to subclasses and thus cannot be overridden.

0.3 `final` Classes: Preventing Inheritance

When you declare a class as `final`, you are making it impossible for any other class to extend it.

- **Purpose:** Often used for security and to create immutable classes. If a class cannot be extended, you can guarantee its behavior.
- **Famous Example:** The `java.lang.String` class is final. This is a key part of its immutability and security, preventing malicious subclasses from altering its behavior.

```
public final class MyImmutableData {
    // ... implementation details
}
// class MaliciousSubclass extends MyImmutableData { ... } // COMPILE ERROR
```

Key Takeaways for the 1Z0-808 Exam

- **Variables:** `final` means one-time assignment. For primitives, the value is constant. For objects, the *reference* is constant, not the object's contents.
- **Methods:** `final` means cannot be *overridden.*
- **Classes:** `final` means cannot be *extended* (inherited).
- Watch for illegal combinations. A method cannot be both `abstract` and `final`. A class cannot be both `abstract` and `final`. The compiler will catch these logical contradictions.