1Z0-808 Exam Topic Reviewer

TopicId: 1012
Topic: Enums

August 5, 2025

Introduction to Enums: Type-Safe Constants

Alright team, let's talk about `enum`. Before Java 5, developers often defined constants like this:

```
public static final int SEASON_WINTER = 0;
public static final int SEASON_SPRING = 1;
// etc.
```

This works, but it's not safe. What stops you from passing a value of '5' to a method expecting a season? Nothing. The compiler can't help you. Enums solve this problem by creating a special type that can only hold a fixed set of constant values. They provide compile-time type safety, which is a huge win for robust code. For the 1Z0-808 exam, you need to know not just how to declare them, but also their more advanced features, because that's where the tricky questions lie.

1 The Basics of Enum Declaration

An enum is a special kind of class. The simplest form looks like this:

```
public enum Season {
    WINTER, SPRING, SUMMER, FALL
}
```

Here, `WINTER`, `SPRING`, `SUMMER`, and `FALL` are not just values; they are instances of the `Season` enum. They are implicitly `public`, `static`, and `final`. You don't need to, and cannot, instantiate an enum using the `new` keyword.

Using Enums

You use them like any other variable. They are especially powerful in `switch` statements.

```
Season currentSeason = Season.SUMMER;

switch (currentSeason) {
    case WINTER: // Notice: No "Season.WINTER"
        System.out.println("It's cold!");
        break;
    case SUMMER:
        System.out.println("It's hot!");
        break;
    default:
        System.out.println("It's a moderate season.");
}
```

Critical Exam Trap: Inside a `switch` statement, you refer to the enum constants directly (e.g., `WINTER`), not with their qualified name (`Season.WINTER`). Using the qualified name will cause a compilation error. The exam loves to test this.

2 Enums with Constructors, Fields, and Methods

This is where enums show their power and where the exam gets interesting. An enum can have instance variables, methods, and constructors, just like a regular class.

```
public enum Season {
    WINTER("Low"),          // Calls constructor with "Low"
    SPRING("Medium"),
    SUMMER("High"),
    FALL("Medium");

    private final String expectedVisitors; // An instance field

    // Constructor - must be private or package-private
    private Season(String expectedVisitors) {
        this.expectedVisitors = expectedVisitors;
    }

    // A regular method
    public void printExpectedVisitors() {
        System.out.println(expectedVisitors);
    }
}

// Usage:
Season.SUMMER.printExpectedVisitors(); // Prints "High"
```

Key Rules for Enum Constructors

- The constructor is called once for each constant at the time the enum class is loaded. You never call it yourself.
- The constructor **cannot be declared** `public` **or** `protected`. If you don't specify an access modifier, it is implicitly `private`. The compiler will reject a `public` constructor.
- The list of enum constants (e.g., `WINTER("Low")`) **must** be the first thing declared in the enum body. A semicolon is required after the last constant if there are other members (fields, methods) in the enum.

3 Essential Enum Methods

All enums implicitly extend the abstract class `java.lang.Enum`, so they inherit its methods. The compiler also adds a few special static methods.

- `public static Season[] values()`:
  Returns an array containing all of the enum constants in the order they are

declared. You can use this for iteration:

```
for (Season s : Season.values()) {
    System.out.println(s);
}
```

- `public static Season valueOf(String name)`:
  Returns the enum constant with the specified name. It's case-sensitive. For example, `Season.valueOf("SUMMER")` returns `Season.SUMMER`. Passing an invalid name throws an `IllegalArgumentException`.
- `public final String name()`:
  Returns the name of the constant exactly as it's declared (e.g., `Season.WINTER.name()` returns `"WINTER"`).
- `public final int ordinal()`:
  Returns the zero-based position of the constant in its declaration. `Season.WINTER.ordinal()` is 0, `Season.SPRING.ordinal()` is 1, and so on. While useful for the exam, relying on this in real-world code is fragile, as reordering the constants will change the ordinal values.

4 Key Takeaways for the 1Z0-808 Exam

- **Comparison:** Use `==` to compare enum constants. It's safe and fast because each constant is a singleton. `.equals()` works too, but `==` is preferred.
- **Inheritance:** An enum **cannot extend another class** because it implicitly extends `java.lang.Enum`. However, an enum **can implement interfaces**.
- **Constructors:** They are implicitly `private` and are called only when the enum is initialized. You cannot invoke them with `new`.
- `switch` **Syntax:** Remember to use the constant name directly (e.g., `case WINTER;`) inside a switch statement.
- **Semicolon:** A semicolon is mandatory after the list of enum constants if the enum body contains any other members like methods or fields.