1Z0-808 Exam Topic Reviewer

TopicId: 1015
Topic: Classes and Objects Fundamentals

August 5, 2025

The Foundation of OOP: Classes and Objects

Welcome to the heart of Java. Everything we do from this point on is built on the concepts of Object-Oriented Programming (OOP). To master Java, you must think in terms of objects. As my colleague said, the exam tests your deep understanding, and it all starts here. Let's use a simple, powerful analogy:

- A **Class** is a *blueprint*. It defines the properties (state) and behaviors that all objects of that type will have. Think of the architectural drawings for a car.
- An **Object** is an *instance* of a class. It is a concrete entity created from the blueprint. It's the actual car that you can drive, which was built from the plans.

From one class (blueprint), you can create many objects (instances).

1 Defining a Class

A class definition consists of members, which are primarily fields and methods.

- **Fields (Instance Variables):** These variables define the **state** of an object. Each object created from the class has its own distinct copy of these fields.
- **Methods:** These functions define the **behavior** of an object. They operate on the object's fields to perform actions.

```
// This is the blueprint for a Dog object.
public class Dog {
    // Fields (State)
    String name;
    String breed;
    int age;

    // Method (Behavior)
    void bark() {
        System.out.println(name + " says: Woof!");
    }
}
```

2 Creating and Using an Object

Creating an object is called **instantiation**. This is done using the `new` keyword, which does two things: allocates memory for the new object and calls its constructor to initialize it.

```
public class PetStore {
    public static void main(String[] args) {
        // 1. Declaration: 'myDog' is a reference variable that can point to a Do
        Dog myDog;
```

```
        // 2. Instantiation & Initialization: Create a new Dog object and assign
        myDog = new Dog();

        // You can access members (fields and methods) using the dot (.) operator
        myDog.name = "Buddy";
        myDog.age = 3;

        // Call the object's method
        myDog.bark(); // Output: Buddy says: Woof!

        // Create another, separate object
        Dog anotherDog = new Dog();
        anotherDog.name = "Lucy";
        anotherDog.bark(); // Output: Lucy says: Woof!
    }
}
```

3 Constructors: Initializing Your Objects

A constructor is a special block of code that is called when an object is created. Its primary job is to initialize the object's state.

Rules for Constructors

- A constructor's name must be **exactly the same** as the class name.
- A constructor has **no return type**, not even `void`.

The Default Constructor

If you do not define any constructor in your class, the Java compiler provides one for you automatically. This is the **default constructor**. It takes no arguments and initializes instance variables to their default values (e.g., `0` for `int`, `null` for objects like `String`, `false` for `boolean`). This is a very common exam topic.

Custom Constructors and the `this` Keyword

You can write your own constructors to provide specific initial values. When a parameter name is the same as a field name, you use the `this` keyword to refer to the *current object's field*.

```
public class Dog {
    String name;
    String breed;

    // A constructor to initialize the Dog object.
    public Dog(String name, String breed) {
        // 'this.name' refers to the instance field.
        // 'name' refers to the method parameter.
```

```
        this.name = name;
        this.breed = breed;
    }

    void bark() { ... }
}

// Usage:
Dog d1 = new Dog("Rex", "German Shepherd");
```

Once you define ANY constructor (even one with arguments), the compiler **no longer provides the default no-argument constructor**. If you still need a no-arg constructor, you must define it yourself.

4 Setting the Stage for the Four Pillars of OOP

This fundamental concept of classes and objects is the launchpad for the four pillars of OOP. You don't need to master them yet, but you must know what they are and how they relate to classes.

- **Encapsulation:** Bundling data (fields) and methods together. We protect data by making fields `private` and providing `public` methods (getters/setters) to access them. This hides internal complexity.
- **Inheritance:** Creating a new class (e.g., `GoldenRetriever`) from an existing one (`Dog`). The new class inherits the fields and methods, promoting code reuse.
- **Polymorphism:** An object's ability to be treated as an instance of its parent class. A `GoldenRetriever` object is also a `Dog` object.
- **Abstraction:** Hiding implementation details and exposing only the necessary functionality, often through `abstract` classes and `interfaces`.

5 Key Takeaways for the 1Z0-808 Exam

- **Distinguish Class and Object:** A class is a template; an object is a runtime instance.
- **Default Values:** Know the default initialization values for instance variables (`0`, `false`, `null`). A question might show code that reads an uninitialized field, and you need to know what value it holds.
- **Constructors:** Memorize the rules. Remember that defining any constructor removes the default one.
- **The `new` keyword** is for creating objects.
- **The `this` keyword** is a reference to the current object instance.