1Z0-808 Exam Topic Reviewer

TopicId: 1017
Topic: Static Members and 'this' Keyword

August 5, 2025

Class vs. Instance: The Meaning of `static`

Alright team, let's clarify one of the most fundamental concepts in Java: the difference between members that belong to a class (the blueprint) and members that belong to an object (the instance). This distinction is controlled by a single keyword: `static`.

Instance Members (Non-Static)

Without the `static` keyword, a field or method is an **instance member**.

- **Instance Fields:** Each object gets its own separate copy. If you have 100 `Car` objects, you have 100 different `color` fields in memory.
- **Instance Methods:** These methods operate on the state of a specific object. They have access to the instance's fields.

1 The `static` Keyword

When you add the `static` keyword, the member now belongs to the **class itself**, not to any individual object.

Static Variables (Class Variables)

- There is only **one copy** of a static variable, and it is shared among all instances of the class.
- If any object modifies a static variable, the change is visible to all other objects of that class.
- They are accessed using the class name, e.g., `ClassName.variableName`.

```
class Car {
    String color; // Instance variable
    static int carCount = 0; // Static variable

    public Car() {
        carCount++; // Increment the shared count for each new car
    }
}

// Usage:
System.out.println("Cars created: " + Car.carCount); // Prints 0
Car c1 = new Car();
Car c2 = new Car();
System.out.println("Cars created: " + Car.carCount); // Prints 2
```

Static Methods

- A static method is called on the class, not on an instance, e.g., `Math.random()`.

- **The Most Important Rule:** A static method is not associated with any particular object instance. Therefore, it **cannot access instance members** (non-static fields or methods) directly.

```
public class Calculator {
    int lastResult; // Instance field

    // Instance method - CAN access instance field 'lastResult'
    public int add(int a, int b) {
        lastResult = a + b;
        return lastResult;
    }

    // Static method - CANNOT access 'lastResult'
    public static int multiply(int a, int b) {
        // lastResult = a * b; // COMPILE ERROR!
        return a * b;
    }
}
```

Why the error? The `multiply` method is called via `Calculator.multiply(5, 10)`. It doesn't know *which* object's `lastResult` field it should access. There is no associated object.

2 The `this` Keyword: A Reference to the Current Object

The `this` keyword is a reference to the **current object instance**. It's an implicit variable available inside any non-static method or constructor.

Primary Uses of `this`

(a) **To resolve ambiguity** between instance variables and parameters:

```
public Person(String name) {
    this.name = name; // this.name is the field, name is the parameter
}
```

(b) **To call another constructor** from a constructor (constructor chaining):

```
public Person() {
    this("Unknown"); // Calls the Person(String) constructor
}
```

3 The Inevitable Collision: `static` and `this`

This is a guaranteed exam concept. Since a `static` method belongs to the class and not to any object, there is no "current object" when a static method is running.

Therefore, you **cannot use the `this` keyword** from within a static context (a static method or static initializer block). Doing so will result in a compilation error: *'non-static variable this cannot be referenced from a static context'*.

```
public class MyClass {
    String instanceName = "Instance";

    public void printInstanceName() {
        System.out.println(this.instanceName); // OKAY - 'this' exists here.
    }

    public static void tryToPrint() {
        // System.out.println(this.instanceName); // COMPILE ERROR!
        // 'this' does not exist in a static context.
    }

    public static void main(String[] args) {
        // tryToPrint();
    }
}
```

4 Key Takeaways for the 1Z0-808 Exam

- **`static` means "one per class".** Non-static means "one per object".
- **Static Access Rule:** A static method can access other static members, but **cannot** access instance members.
- **Instance Access Rule:** An instance method can access both instance members (using an implicit 'this') and static members.
- **`this` is for instances only.** It is a reference to the current object and is not available in a static context. The exam will test this directly.
- You can access static members through an instance variable (e.g., `c1.carCount`), but this is bad practice. The exam may show this valid code to confuse you. The correct way is to use the class name (`Car.carCount`).