1Z0-808 Exam Topic Reviewer

TopicId: 1016

Topic: Constructors and Initialization Blocks

August 5, 2025

The Birth of an Object: A Precise Sequence

Team, we know that the `new` keyword creates an object, but the 1Z0-808 exam demands that you know the *exact* sequence of events that happens during an object's creation. The questions will often feature strange-looking code with print statements in various places, and you must be able to trace the output perfectly. Let's master this sequence.

1 Revisiting Constructors

A constructor's job is to initialize an object's state. Let's build on what we know.

Constructor Overloading

A class can have multiple constructors, as long as their parameter lists are different (in number, type, or order of parameters). This provides flexibility in how objects are created.

```
public class Shirt {
    String color;
    char size;

    // Constructor 1: No arguments
    public Shirt() {
        this.color = "White";
        this.size = 'M';
    }

    // Constructor 2: Takes a color
    public Shirt(String color) {
        this.color = color;
        this.size = 'M';
    }

    // Constructor 3: Takes color and size
    public Shirt(String color, char size) {
        this.color = color;
        this.size = size;
    }
}
```

Constructor Chaining with `this()`

Notice the code duplication in the constructors above. We can eliminate this by having one constructor call another using `this()`. This is called constructor chaining.

The Rule: A call to `this(...)` must be the **very first statement** in a constructor.

```
public class Shirt {
    String color;
    char size;

    // The "main" constructor that does the work
    public Shirt(String color, char size) {
        this.color = color;
        this.size = size;
    }

    // This constructor calls the main one, providing a default size
    public Shirt(String color) {
        this(color, 'M'); // Calls the (String, char) constructor
    }

    // This constructor calls the second one, providing a default color
    public Shirt() {
        this("White"); // Calls the (String) constructor
    }
}
```

2 Initialization Blocks

Sometimes you need logic to run during initialization that doesn't fit well in a constructor. Java provides two special code blocks for this.

Instance Initializer Block

This is a block of code written directly in the class body, enclosed in curly braces {...}. It is executed **every time an instance of the class is created**.

```
class Log {
    {
        // This instance initializer runs for every new Log object
        System.out.println("Initializing new log instance...");
    }
}
```

Static Initializer Block

This block is marked with the `static` keyword. It is executed only **once**, when the JVM first loads the class into memory. It runs before any static members are used and long before any instances are created.

```
class DatabaseConnection {
    static {
        // This runs only once when the class is first loaded
        System.out.println("Loading database driver...");
    }
}
```

3 The Exam-Critical Order of Initialization

This is the key takeaway. You must memorize this order. Let's trace the creation of a `Child` object where `Child` extends `Parent`.

Order of Events:

(a) **Class Loading Phase:**

  i. **Parent** class is loaded. All `static` variable declarations and `static` initializers of `Parent` are run in the order they appear.

  ii. **Child** class is loaded. All `static` variable declarations and `static` initializers of `Child` are run in the order they appear.

(b) **Instance Creation Phase (for `new Child()`):**

  i. **Parent** object part is created first. All *instance* variable declarations and *instance* initializers of `Parent` are run in order.

  ii. The **Parent** constructor is run.

  iii. **Child** object part is created. All *instance* variable declarations and *instance* initializers of `Child` are run in order.

  iv. The **Child** constructor is run.

Example Trace:

```
class Parent {
    static { System.out.println("1. Parent static block"); }
    { System.out.println("3. Parent instance block"); }
    Parent() { System.out.println("4. Parent constructor"); }
}
class Child extends Parent {
    static { System.out.println("2. Child static block"); }
    { System.out.println("5. Child instance block"); }
    Child() { System.out.println("6. Child constructor"); }
    public static void main(String[] args) {
        new Child();
    }
}
```

Output:

```
1. Parent static block
2. Child static block
3. Parent instance block
4. Parent constructor
5. Child instance block
6. Child constructor
```

4 Key Takeaways for the 1Z0-808 Exam

- **Order is King:** Burn the initialization order into your memory. Statics of parent, then statics of child. Then for the instance: instance blocks of parent, constructor of parent, instance blocks of child, constructor of child.
- `this()` **call:** Must be the first statement in a constructor.
- **Static blocks run once.** Instance blocks run for every new object.
- When you see an initialization question, don't rush. Grab a piece of paper and trace the execution step-by-step according to the rules.