Singleton Class vs Normal Class in Java

Both **singleton classes** and **normal classes** are used to create objects in Java, but they serve different purposes. Below is a detailed comparison, along with examples.

---

1. What is a Singleton Class?

A **Singleton Class** ensures that **only one instance** of the class is created throughout the program. This is useful when only one shared resource (e.g., a database connection, logger, or configuration manager) should exist.

Example of a Singleton Class:

```
class Singleton {
    private static Singleton instance;  // Single instance

    // Private constructor prevents instantiation from outside the class
    private Singleton() {}

    // Public method to provide access to the instance
    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();  // Lazy initialization
        }
        return instance;
    }

    public void showMessage() {
        System.out.println("Hello from Singleton!");
    }
}

public class Main {
    public static void main(String[] args) {
        Singleton obj1 = Singleton.getInstance();
        Singleton obj2 = Singleton.getInstance();

        obj1.showMessage();

        // Checking if both references point to the same object
        System.out.println(obj1 == obj2);  // Output: true (Same object)
    }
}
```

Key Features of Singleton:

✅ **Single Instance:** Only one object exists.
✅ **Controlled Access:** Instance is created only when needed (lazy initialization).

✅ **Private Constructor:** Prevents creating objects using `new`.
✅ **Global Access:** Available via `getInstance()` method.

---

2. What is a Normal Class?

A **normal class** allows multiple instances to be created whenever needed. Each object will have its own copy of instance variables.

Example of a Normal Class:

```
class NormalClass {
    public void showMessage() {
        System.out.println("Hello from Normal Class!");
    }
}

public class Main {
    public static void main(String[] args) {
        NormalClass obj1 = new NormalClass();
        NormalClass obj2 = new NormalClass();

        obj1.showMessage();
        obj2.showMessage();

        // Checking if both references point to the same object
        System.out.println(obj1 == obj2);  // Output: false (Different
objects)
    }
}
```

Key Features of Normal Class:

✅ **Multiple Instances:** Can create multiple objects.
✅ **Independent Objects:** Each object has separate memory allocation.
✅ **More Flexible:** Suitable when multiple instances are required.

---

3. Singleton vs Normal Class: A Comparison Table

Feature	Singleton Class	Normal Class
Instance	Only one instance	Multiple instances possible
Access	Accessed via `getInstance()`	Created using `new` keyword
Constructor	Private (to restrict object creation)	Public (can create multiple objects)
Memory Usage	Saves memory by reusing a single object	Each object takes separate memory
Use Case	When only one shared object is needed (e.g., Logger, Config, Database Connection)	When separate instances are needed for different tasks

---

4. When to Use Singleton?

🔹 **Managing a single shared resource:** Database connections, logging framework, thread pools.
🔹 **Restricting multiple instances:** Ensuring that only one instance exists (e.g., Configuration manager).
🔹 **Improving performance & memory usage:** Since only one instance exists, memory is saved.

5. When to Use Normal Class?

🔹 **When separate independent objects are needed.**
🔹 **For normal programming scenarios like User, Product, Employee objects.**
🔹 **When multiple configurations or states are needed across different objects.**

Final Thought

✅ **Use Singleton** when you need to control object creation (like a single DB connection).
✅ **Use Normal Class** when each object should be independent (like an Employee class).

Would you like me to create a PDF of this explanation? ☺