**Week – 1**

**DESIGN PRINCIPLES AND PATTERNS**

**Exercise – 1:** Implementing the Singleton Pattern

**Singleton Pattern:**

The Singleton is a *creational design pattern* that ensures only one instance of a class is created throughout an application, and it provides a global point of access to that instance.

**CHARACTERISTICS:**

**1. Private constructor** — prevents instantiation from other classes.

**2.Static field** — holds the sole instance of the class.

**3.Public static access method** — typically getInstance(), to retrieve the singleton instance

**Why Use It?**

* **Coordinate shared resources** like loggers, config managers, or thread pools.
* **Lazy initialization** helps save resources by delaying object creation until it's needed .

**SOURCE CODE:**

enum SingletonEnum {

INSTANCE;

int value;

public int getValue() { return value; }

public void setValue(int value) { this.value = value; }

}

public class Main {

public static void main(String[] args) {

SingletonEnum singleton = SingletonEnum.INSTANCE;

System.out.println(singleton.getValue()); // default value

singleton.setValue(2);

System.out.println(singleton.getValue()); // after setting to 2

}

}

OUTPUT:

