**WEEK-1**

**Exercise 1: Implementing the Singleton Pattern**

**CODE:**

class Logger {

private static Logger instance;

private Logger() {

System.out.println("Logger instance created.");

}

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) {

System.out.println(" " + message);

}

}

public class Main {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

logger1.log("This is the first message.");

logger2.log("This is the second message.");

if (logger1 == logger2) {

System.out.println("Both logger1 and logger2 are the same instance.");

} else {

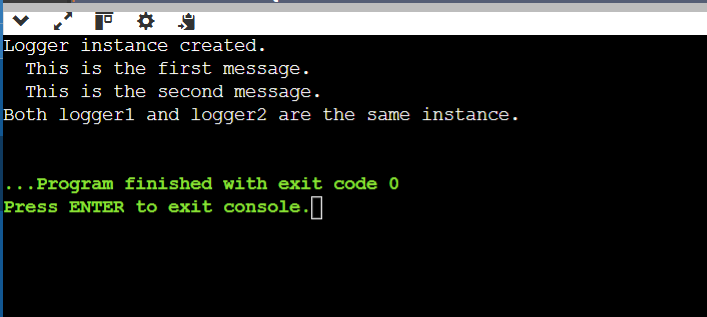
System.out.println("Different instances! Not Singleton.");

}

}

}

**OUTPUT:**

****

**Exercise 2: Implementing the Factory Method Pattern**

**CODE:**

interface Animal {

void speak();

}

class Dog implements Animal {

public void speak() {

System.out.println("Woof! I’m a Dog.");

}

}

class Cat implements Animal {

public void speak() {

System.out.println("Meow! I’m a Cat.");

}

}

class Cow implements Animal {

public void speak() {

System.out.println("Moo! I’m a Cow.");

}

}

class AnimalFactory {

public Animal getAnimal(String animalType) {

if (animalType == null) return null;

if (animalType.equalsIgnoreCase("DOG")) {

return new Dog();

} else if (animalType.equalsIgnoreCase("CAT")) {

return new Cat();

} else if (animalType.equalsIgnoreCase("COW")) {

return new Cow();

}

return null;

}

}

public class Main {

public static void main(String[] args) {

AnimalFactory factory = new AnimalFactory();

Animal animal1 = factory.getAnimal("DOG");

animal1.speak();

Animal animal2 = factory.getAnimal("CAT");

animal2.speak();

Animal animal3 = factory.getAnimal("COW");

animal3.speak();

Animal unknown = factory.getAnimal("LION");

if (unknown == null) {

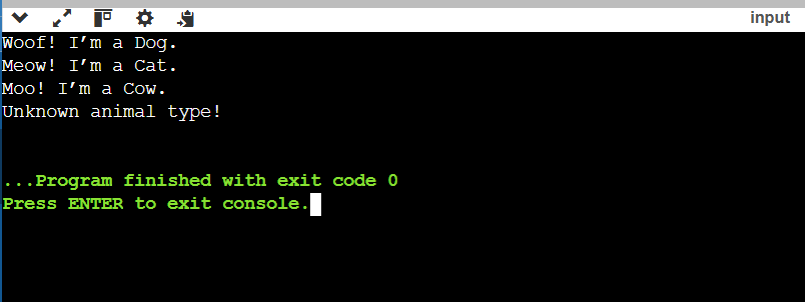
System.out.println("Unknown animal type!");

}

}

}

**OUTPUT:**

****