**Week 1: Design Pattern and Principles**

**Exercise 1: Implementing the Singleton Pattern**

//.Logger class

public class Logger {

    private static Logger instance;

    private Logger() {

      }

    public static Logger getInstance() {

        if (instance == null) {

            instance = new Logger();

        }

        return instance;

    }

    public void log(String message) {

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

    }

}

//. SingletonPatternTest class

public class SingletonPatternTest{

    public static void main(String[] args) {

        Logger logger1 = Logger.getInstance();

        Logger logger2 = Logger.getInstance();

        logger1.log("This is a log message from logger1.");

        logger2.log("This is a log message from logger2.");

        System.out.println(logger1.hashCode());

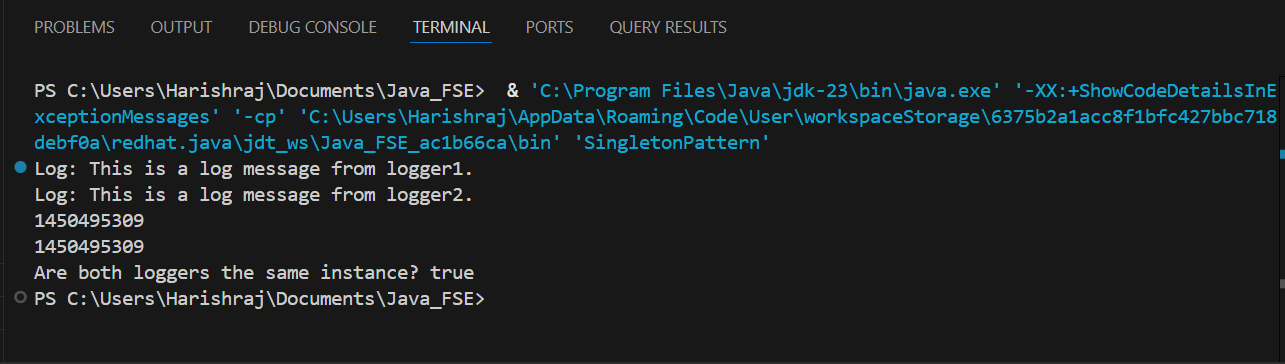
        System.out.println(logger2.hashCode());

        System.out.println("Are both loggers the same instance? " + (logger1 == logger2));

    }

}

Output:



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

import java.util.Scanner;

public class FactoryMethodPattern {

    public interface Document {

        void open();

        void close();

    }

    public static class WordDocument implements Document {

        @Override

        public void open() {

            System.out.println("Opening Word Document.");

        }

        @Override

        public void close() {

            System.out.println("Closing Word Document.");

        }

    }

    public static class PdfDocument implements Document {

        @Override

        public void open() {

            System.out.println("Opening PDF Document.");

        }

        @Override

        public void close() {

            System.out.println("Closing PDF Document.");

        }

    }

    public static class ExcelDocument implements Document {

        @Override

        public void open() {

            System.out.println("Opening Excel Document.");

        }

        @Override

        public void close() {

            System.out.println("Closing Excel Document.");

        }

    }

    public abstract static class DocumentFactory {

        public abstract Document createDocument();

    }

    public static class WordDocumentFactory extends DocumentFactory {

        @Override

        public Document createDocument() {

            return new WordDocument();

        }

    }

    public static class PdfDocumentFactory extends DocumentFactory {

        @Override

        public Document createDocument() {

            return new PdfDocument();

        }

    }

    public static class ExcelDocumentFactory extends DocumentFactory {

        @Override

        public Document createDocument() {

            return new ExcelDocument();

        }

    }

    public static class FactoryMethodTest {

        public static void main(String[] args) {

            Scanner scanner = new Scanner(System.in);

            System.out.println("Enter the type of document to create (Word, PDF, Excel):");

            String documentType = scanner.nextLine().trim().toLowerCase();

            DocumentFactory factory = null;

            switch (documentType) {

                case "word":

                    factory = new WordDocumentFactory();

                    break;

                case "pdf":

                    factory = new PdfDocumentFactory();

                    break;

                case "excel":

                    factory = new ExcelDocumentFactory();

                    break;

                default:

                    System.out.println("Invalid document type.");

                    System.exit(1);

            }

            Document document = factory.createDocument();

            document.open();

            document.close();

            scanner.close();

        }

    }

}

Output:

