**Exercise 1: Implementing the Singleton Pattern**

public class LoggerSingletonExample {

    public static void main(String[] args) {

        Logger logger1 = Logger.getInstance();

        logger1.log("message (1)");

        Logger logger2 = Logger.getInstance();

        logger2.log("message(2)");

        if (logger1 == logger2) {

            System.out.println("Singleton works ");

        } else {

            System.out.println("Singleton has been failed" );

        }

    }

}

class Logger {

    private static Logger instance;

    public static Logger getInstance() {

        if (instance == null) {

            instance = new Logger();

        }

        return instance;

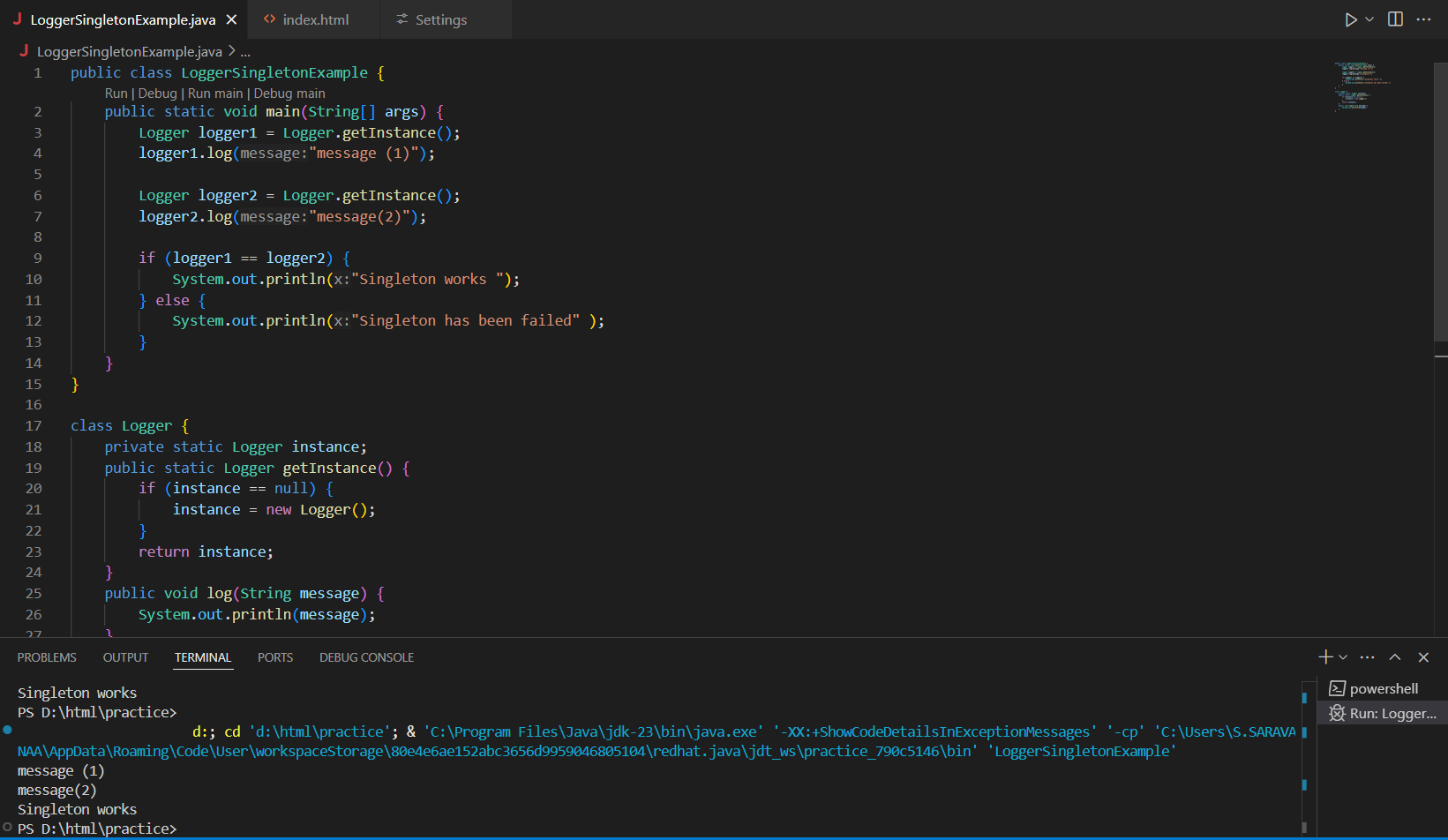
    }

    public void log(String message) {

        System.out.println(message);

    }

}



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

// Step 2

interface Document {

    void open();

    void save();

}

// Step 3

class WordDocument implements Document {

    public void open() { System.out.println("Word document opened"); }

    public void save() { System.out.println("Word document saved"); }

}

class PdfDocument implements Document {

    public void open() { System.out.println("PDF document opened"); }

    public void save() { System.out.println("PDF document saved"); }

}

class ExcelDocument implements Document {

    public void open() { System.out.println("Excel document opened"); }

    public void save() { System.out.println("Excel document saved"); }

}

// Step 4

abstract class DocumentFactory {

    public abstract Document createDocument();

}

class WordDocumentFactory extends DocumentFactory {

    public Document createDocument() { return new WordDocument(); }

}

class PdfDocumentFactory extends DocumentFactory {

    public Document createDocument() { return new PdfDocument(); }

}

class ExcelDocumentFactory extends DocumentFactory {

    public Document createDocument() { return new ExcelDocument(); }

}

// Step 5

public class FactoryMethodPatternExample {

    public static void main(String[] args) {

        DocumentFactory wordFactory = new WordDocumentFactory();

        Document wordDoc = wordFactory.createDocument();

        wordDoc.open();

        wordDoc.save();

        DocumentFactory pdfFactory = new PdfDocumentFactory();

        Document pdfDoc = pdfFactory.createDocument();

        pdfDoc.open();

        pdfDoc.save();

        DocumentFactory excelFactory = new ExcelDocumentFactory();

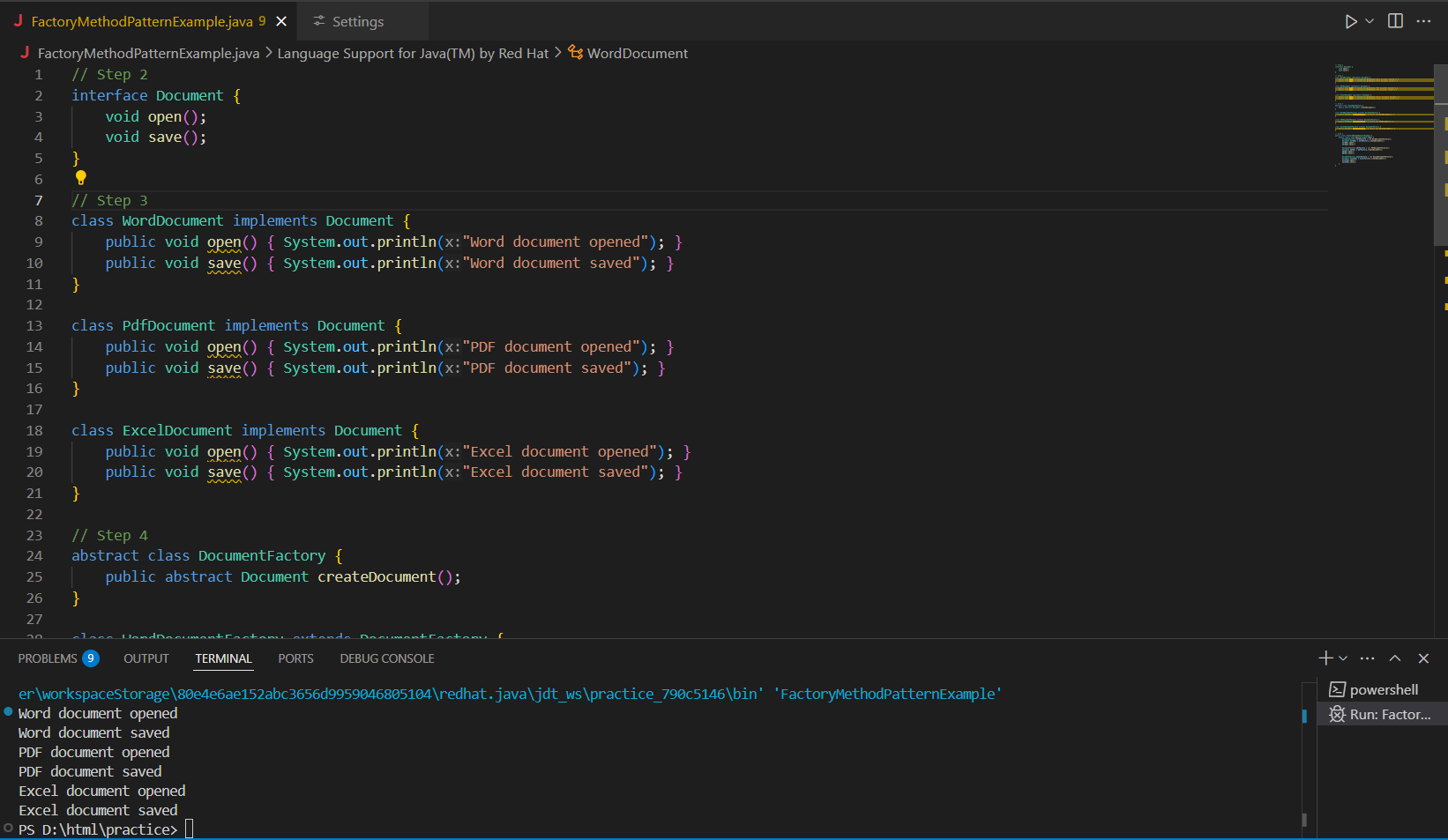
        Document excelDoc = excelFactory.createDocument();

        excelDoc.open();

        excelDoc.save();

    }

}



**Exercise 2: E-commerce Platform Search Function**

class Product {

    int id;

    String name;

    String category;

    public Product(int i, String n, String c) {

        id = i;

        name = n;

        category = c;

    }

}

public class Main {

    static void linearSearch(Product[] p, String searchName) {

        for (Product product : p) {

            if (product.name.equalsIgnoreCase(searchName)) {

                System.out.println("Found: " + product.name + " in " + product.category);

                return;

            }

        }

    }

    static void binarySearch(Product[] p, String searchName) {

        java.util.Arrays.sort(p, Main::compareByName);

        int low = 0, high = p.length - 1;

        while (low <= high) {

            int mid = (low + high) / 2;

            int result = p[mid].name.compareToIgnoreCase(searchName);

            if (result == 0) {

                System.out.println("Found: " + p[mid].name + " in " + p[mid].category);

                return;

            } else if (result < 0) {

                low = mid + 1;

            } else {

                high = mid - 1;

            }

        }

    }

    static int compareByName(Product a, Product b) {

        return a.name.compareToIgnoreCase(b.name);

    }

    public static void main(String[] args) {

       Product[] products = {

    new Product(1, "Phone", "Electronics"),

    new Product(2, "Shoes", "Footwear"),

    new Product(3, "Bag", "Accessories"),

    new Product(4, "T-Shirt", "Clothing"),

    new Product(5, "Watch", "Accessories")

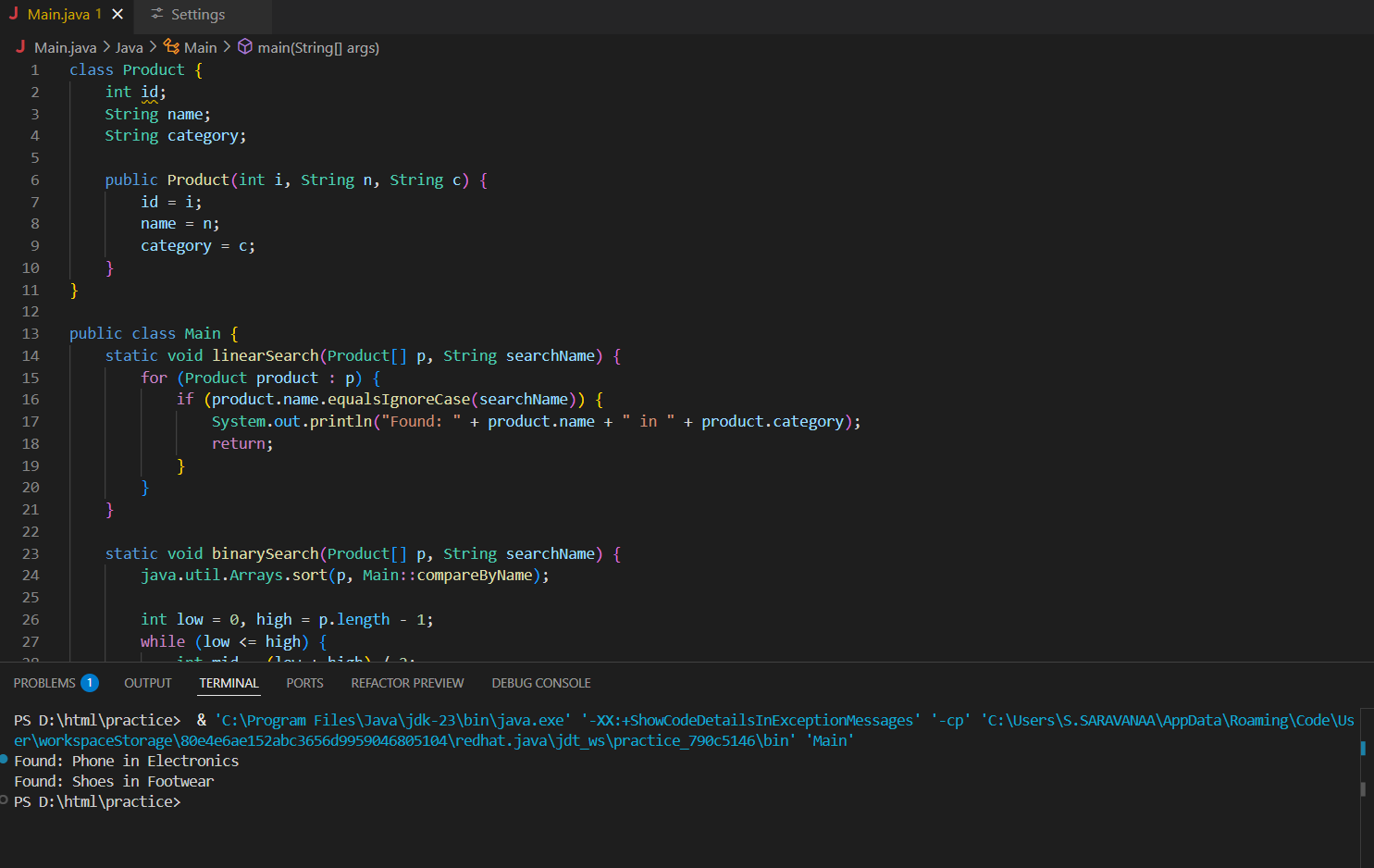
};

        linearSearch(products, "Phone");

        binarySearch(products, "Shoes");

    }

}



**Exercise 7: Financial Forecasting**

public class Main {

    public static void main(String[] args) {

        double result = predict(1000, 0.10, 5);

        System.out.println("Future Value: " + result);

    }

    static double predict(double value, double rate, int years) {

        if (years == 0) return value;

        return predict(value, rate, years - 1) \* (1 + rate);

    }

}

