**WEEK 1**

**DESIGN PRINCIPLES AND PATTERNS**

**Exercise 1: Implementing the Singleton Pattern**

**package** weekone;

**public** **class** SingletonPattern {

**static** **class** Logger {

**private** **static** Logger *instance*;

**private** Logger() {

System.***out***.println("Logger Initialized");

}

**public** **static** Logger getInstance() {

**if** (*instance* == **null**) {

*instance* = **new** Logger();

}

**return** *instance*;

}

**public** **void** log(String message) {

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

}

}

**public** **static** **void** main(String[] args) {

Logger logger1 = Logger.*getInstance*();

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

Logger logger2 = Logger.*getInstance*();

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

**if** (logger1 == logger2) {

System.***out***.println("Both logger instances are the same (Singleton works).");

} **else** {

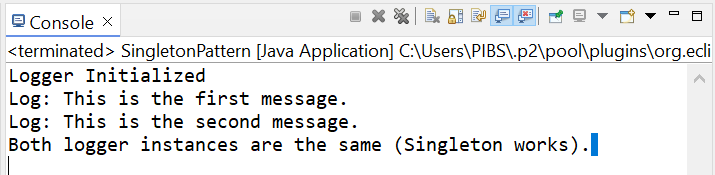
System.***out***.println("Different instances (Singleton failed).");

}

}

}

**OUTPUT:**



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

**package** weekone;

**public** **class** FactoryMethodPattern {

**interface** Document {

**void** open();

}

**static** **class** WordDocument **implements** Document {

**public** **void** open() {

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

}

}

**static** **class** PdfDocument **implements** Document {

**public** **void** open() {

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

}

}

**static** **class** ExcelDocument **implements** Document {

**public** **void** open() {

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

}

}

**static** **abstract** **class** DocumentFactory {

**abstract** Document createDocument();

}

**static** **class** WordFactory **extends** DocumentFactory {

Document createDocument() {

**return** **new** WordDocument();

}

}

**static** **class** PdfFactory **extends** DocumentFactory {

Document createDocument() {

**return** **new** PdfDocument();

}

}

**static** **class** ExcelFactory **extends** DocumentFactory {

Document createDocument() {

**return** **new** ExcelDocument();

}

}

**public** **static** **void** main(String[] args) {

DocumentFactory wordFactory = **new** WordFactory();

Document word = wordFactory.createDocument();

word.open();

DocumentFactory pdfFactory = **new** PdfFactory();

Document pdf = pdfFactory.createDocument();

pdf.open();

DocumentFactory excelFactory = **new** ExcelFactory();

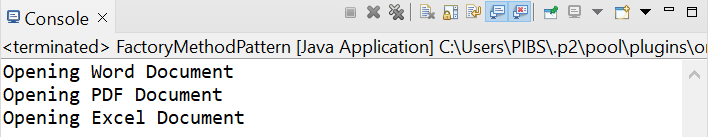
Document excel = excelFactory.createDocument();

excel.open();

}

}

**OUTPUT:**



**DATA STRUCTURES AND ALGORITHMS**

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

**package** weekone;

**import** java.util.\*;

**public** **class** EcommercePlatformSearchFunction {

**static** **class** Product {

**int** productId;

String productName;

String category;

Product(**int** productId, String productName, String category) {

**this**.productId = productId;

**this**.productName = productName;

**this**.category = category;

}

**public** String toString() {

**return** productId + " - " + productName + " (" + category + ")";

}

}

**public** **static** Product linearSearch(Product[] products, String name) {

**for** (Product p : products) {

**if** (p.productName.equalsIgnoreCase(name)) {

**return** p;

}

}

**return** **null**;

}

**public** **static** Product binarySearch(Product[] products, String name) {

**int** left = 0;

**int** right = products.length - 1;

**while** (left <= right) {

**int** mid = (left + right) / 2;

**int** compare = products[mid].productName.compareToIgnoreCase(name);

**if** (compare == 0) **return** products[mid];

**else** **if** (compare < 0) left = mid + 1;

**else** right = mid - 1;

}

**return** **null**;

}

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

Product[] products = {

**new** Product(101, "Shoes", "Fashion"),

**new** Product(102, "Phone", "Electronics"),

**new** Product(103, "Watch", "Accessories"),

**new** Product(104, "Laptop", "Electronics"),

**new** Product(105, "Bag", "Travel")

};

System.***out***.print("Enter product name to search: ");

String searchName = scanner.nextLine();

System.***out***.print("Enter search method (linear/binary): ");

String method = scanner.nextLine().toLowerCase();

Product result = **null**;

**if** (method.equals("linear")) {

result = *linearSearch*(products, searchName);

} **else** **if** (method.equals("binary")) {

Arrays.*sort*(products, (a, b) -> a.productName.compareToIgnoreCase(b.productName));

result = *binarySearch*(products, searchName);

} **else** {

System.***out***.println("Invalid search method.");

}

**if** (result != **null**) {

System.***out***.println("Product found: " + result);

} **else** {

System.***out***.println("Product not found.");

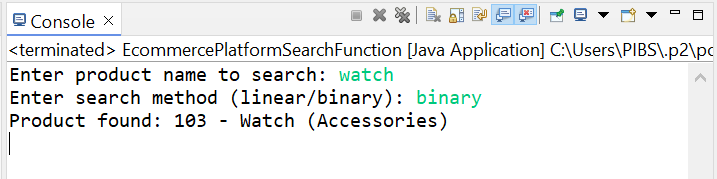
}

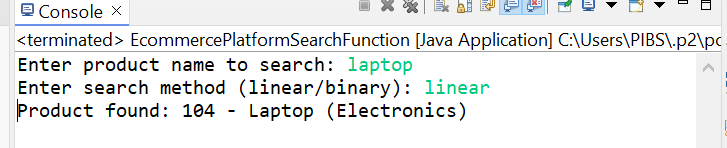
scanner.close();

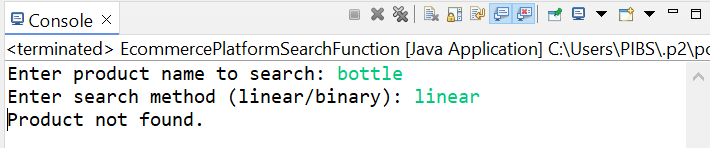
}

}

**OUTPUT:**







**Exercise 7: Financial Forecasting**

**package** weekone;

**import** java.util.\*;

**public** **class** FinancialForecasting {

**public** **static** **double** forecast(**double** presentValue, **double** growthRate, **int** years) {

**if** (years == 0) {

**return** presentValue;

}

**return** *forecast*(presentValue, growthRate, years - 1) \* (1 + growthRate);

}

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter present value: ");

**double** presentValue = scanner.nextDouble();

System.***out***.print("Enter annual growth rate (e.g., 0.05 for 5%): ");

**double** growthRate = scanner.nextDouble();

System.***out***.print("Enter number of years: ");

**int** years = scanner.nextInt();

**double** futureValue = *forecast*(presentValue, growthRate, years);

System.***out***.printf("Future Value after %d years: %.2f\n", years, futureValue);

scanner.close();

}

}

**OUTPUT:**

