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

Scenario:

You are developing a document management system that needs to create different types of documents (e.g., Word, PDF, Excel). Use the Factory Method Pattern to achieve this.

**Product.java**  
public class Product {

int productId;

String productName;

String category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

public String toString() {

return "ID: " + productId + ", Name: " + productName + ", Category: " + category;

}

}

**LinearSearch.java**

public class LinearSearch {

public static Product search(Product[] products, int targetId) {

for (Product product : products) {

if (product.productId == targetId) {

return product;

}

}

return null;

}

}

**BinarySearch.java**

import java.util.Arrays;

import java.util.Comparator;

public class BinarySearch {

public static Product search(Product[] products, int targetId) {

int left = 0;

int right = products.length - 1;

while (left <= right) {

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

if (products[mid].productId == targetId) {

return products[mid];

} else if (products[mid].productId < targetId) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return null;

}

public static void sortProducts(Product[] products) {

Arrays.sort(products, Comparator.comparingInt(p -> p.productId));

}

}

**SearchTest.java**

public class SearchTest {

public static void main(String[] args) {

Product[] products = {

new Product(103, "Laptop", "Electronics"),

new Product(101, "Shirt", "Clothing"),

new Product(105, "Mobile", "Electronics"),

new Product(102, "Shoes", "Footwear"),

new Product(104, "Book", "Stationery")

};

int targetId = 104;

// Linear Search

Product linearResult = LinearSearch.search(products, targetId);

System.out.println("Linear Search Result: " + (linearResult != null ? linearResult : "Not found"));

// Binary Search (requires sorting)

BinarySearch.sortProducts(products);

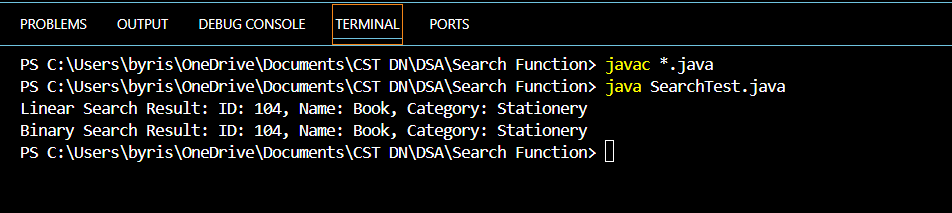
Product binaryResult = BinarySearch.search(products, targetId);

System.out.println("Binary Search Result: " + (binaryResult != null ? binaryResult : "Not found"));

}

}

**Output**



**Exercise 7:** Financial Forecasting

Scenario:

You are developing a financial forecasting tool that predicts future values based on past data.

**FinancialForecast.java**

public class FinancialForecast {

// Recursive method to calculate future value

public static double forecast(double currentValue, double growthRate, int years) {

if (years == 0) {

return currentValue; // base case: no more years to forecast

} else {

// Apply growth and recursively call for remaining years

double nextValue = currentValue \* (1 + growthRate);

return forecast(nextValue, growthRate, years - 1);

}

}

public static void main(String[] args) {

double initialValue = 10000; // ₹10,000 initial investment

double growthRate = 0.08; // 8% annual growth

int years = 5; // forecast for 5 years

double futureValue = forecast(initialValue, growthRate, years);

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

}

}

**Output**

