EXERCISE 2: E-commerce Platform Search Function

class Product {

int productId;

String productName;

String category;

public Product(int id, String name, String category) {

this.productId = id;

this.productName = name;

this.category = category;

}

public String toString() {

return "[" + productId + "] " + productName + " - " + category;

}

}

public class EcommerceSearchFunction {

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

for (Product product : products) {

if (product.productId == targetId) {

return product;

}

}

return null;

}

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

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

while (low <= high) {

int mid = (low + high) / 2;

if (products[mid].productId == targetId) return products[mid];

else if (products[mid].productId < targetId) low = mid + 1;

else high = mid - 1;

}

return null;

}

public static void main(String[] args) {

Product[] products = {

new Product(101, "Shoes", "Footwear"),

new Product(102, "T-Shirt", "Clothing"),

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

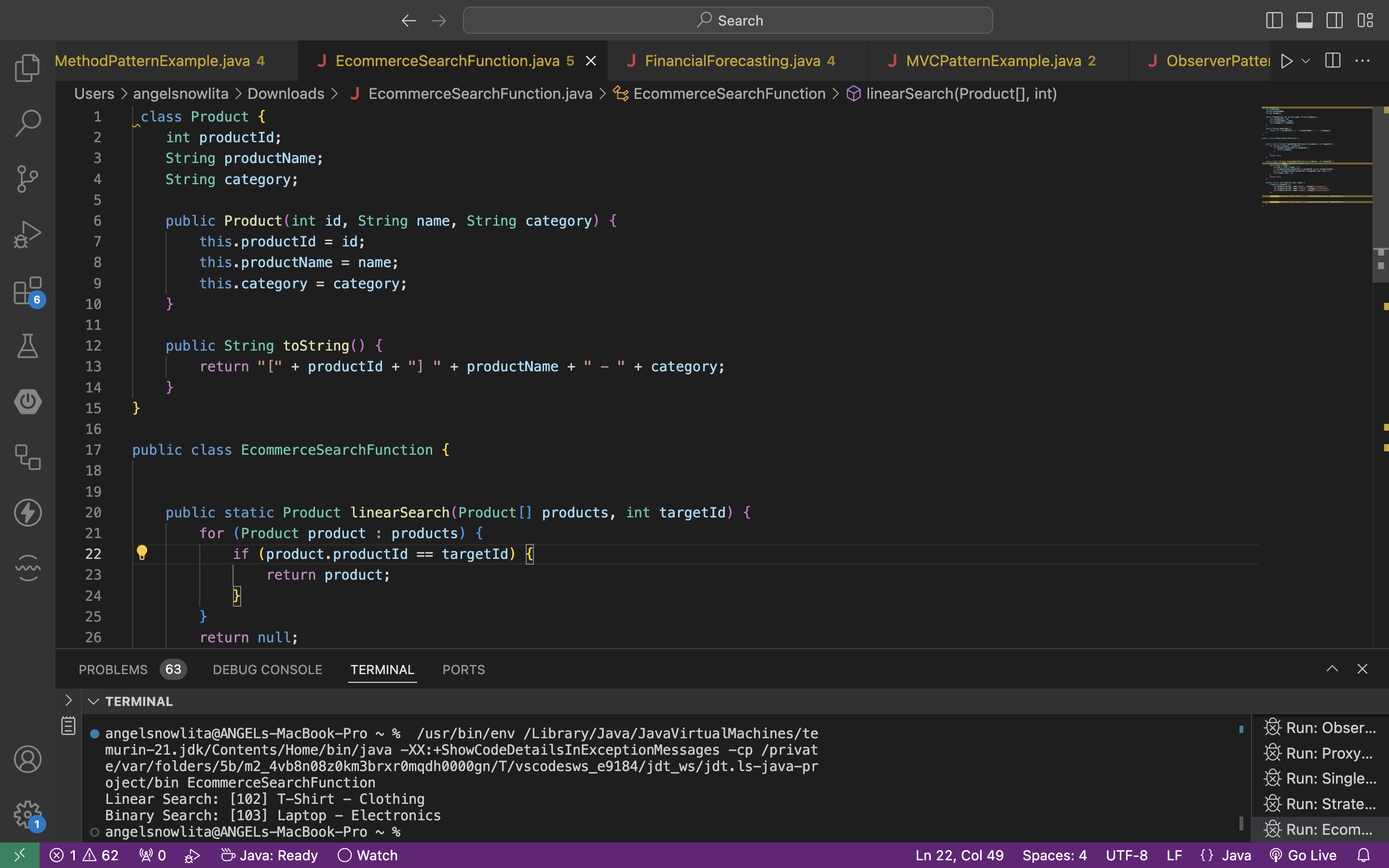
};

System.out.println("Linear Search: " + linearSearch(products, 102));

System.out.println("Binary Search: " + binarySearch(products, 103));

}

}



**Exercise 7: Financial Forecasting**

public class FinancialForecasting {

public static double forecastValue(double current, double rate, int years) {

if (years == 0) return current;

return forecastValue(current \* (1 + rate), rate, years - 1);

}

public static double forecastIterative(double current, double rate, int years) {

for (int i = 0; i < years; i++) {

current \*= (1 + rate);

}

return current;

}

public static void main(String[] args) {

double current = 10000;

double rate = 0.05;

int years = 5;

System.out.println("Recursive Forecast: " + forecastValue(current, rate, years));

System.out.println("Iterative Forecast: " + forecastIterative(current, rate, years));

}

}

