**Deep Skilling - Week 1 Mandatory Hands-on Exercises**

**Skill : Data structures and Algorithms**

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

**Code:**

**Product.java**

**package SearchFunction;**

**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 productId + " - " + productName + " -> " + category;**

**}**

**}**

**LinearSearch.java**

**package SearchFunction;**

**public class LinearSearch {**

**public static Product searchForProduct(Product[] products, String targetProduct) {**

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

**if(product.productName.equalsIgnoreCase(targetProduct))**

**return product;**

**}**

**return null;**

**}**

**}**

**BinarySearch.java**

**package SearchFunction;**

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

**public class BinarySearch {**

**public static Product searchForProduct(Product[] products, String targetProduct) {**

**// Sorting the Products based in the name of products**

**Arrays.*sort*(products, Comparator.*comparing*(p -> p.productName.toLowerCase()));**

**int left = 0, right = products.length - 1;**

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

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

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

**if(compare == 0) {**

**return products[mid];**

**} else if(compare < 0) {**

**left = mid + 1;**

**} else {**

**right = mid - 1;**

**}**

**}**

**return null;**

**}**

**}**

**Test.java**

**package SearchFunction;**

**public class Test {**

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

**Product[] products = {**

**new Product(101, "Apple", "Fruits"),**

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

**new Product(103, "Mango", "Fruits"),**

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

**new Product(105, "Walnut", "Nuts"),**

**new Product(106, "Almond", "Nuts"),**

**new Product(107, "Guava", "Fruits"),**

**new Product(108, "Smartphone", "Electronics")**

**};**

**Product linearSearch = LinearSearch.*searchForProduct*(products, "Almond");**

**System.*out*.println("Linear Search Result: " + (linearSearch != null ? linearSearch : "Not Found"));**

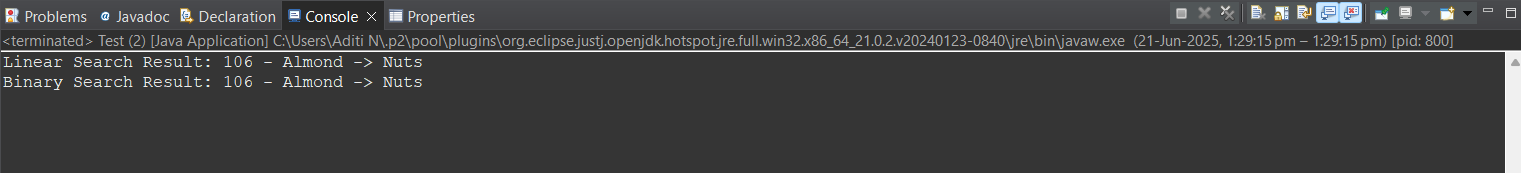
**Product binarySearch = BinarySearch.*searchForProduct*(products, "Almond");**

**System.*out*.println("Binary Search Result: " + (binarySearch != null ? binarySearch : "Not Found"));**

**}**

**}**

**Output:**

****

**Exercise - 7: Financial Forecasting**

**Code:**

**FinancialForecast.java**

**package RecursionExample;**

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

**public class FinancialForecast {**

**public static double findFutureValue(double resultValue, double growthRate, int years) {**

**if(years == 0)**

**return resultValue;**

**return (1 + growthRate) \* *findFutureValue*(resultValue, growthRate, years - 1);**

**}**

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

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

**double givenValue, growthRate;**

**int years;**

**System.*out*.print("Enter the actual amount: ");**

**givenValue = scan.nextDouble();**

**System.*out*.print("Enter growth rate: ");**

**growthRate = scan.nextDouble();**

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

**years = scan.nextInt();**

**double futureVal = *findFutureValue*(givenValue, growthRate, years);**

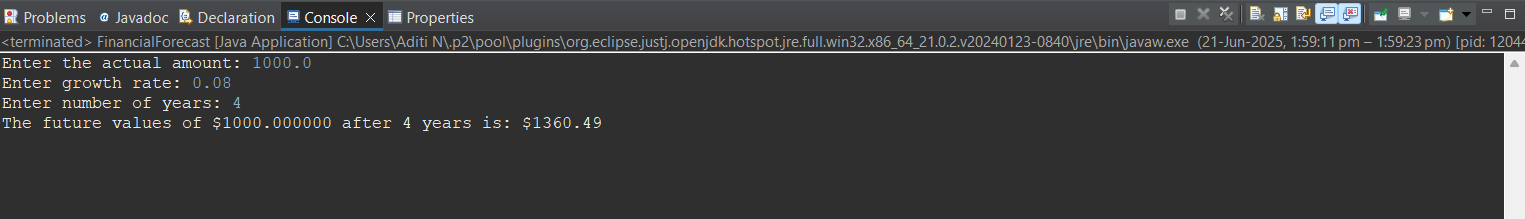
**System.*out*.printf("The future values of $%f after %d years is: $%.2f", givenValue, years, futureVal);**

**scan.close();**

**}**

**}**

**Output:**

****