**SUPERSET ID:** **6376594**

**DATA STRUCTURES AND ALGORITHMS-HANDS ON**

**IDE:ECLIPSE**

**Exercise 1: Inventory Management System**

**Project Name:** **InventoryManagementSystem**

Product.java Code:

**public** **class** Product {

**int** productId;

String productName;

**int** quantity;

**double** price;

**public** Product(**int** productId, String productName, **int** quantity, **double** price) {

**this**.productId = productId;

**this**.productName = productName;

**this**.quantity = quantity;

**this**.price = price;

}

**public** String toString() {

**return** "ProductID: " + productId + ", Name: " + productName + ", Quantity: " + quantity + ", Price: " + price;

}

}

InventoryManager.java Code:

**import** java.util.HashMap;

**public** **class** InventoryManager {

**static** HashMap<Integer, Product> *inventory* = **new** HashMap<>();

**public** **static** **void** addProduct(Product product) {

*inventory*.put(product.productId, product);

System.***out***.println("Product added.");

}

**public** **static** **void** updateProduct(**int** id, **int** quantity, **double** price) {

**if** (*inventory*.containsKey(id)) {

Product p = *inventory*.get(id);

p.quantity = quantity;

p.price = price;

System.***out***.println("Product updated.");

} **else** {

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

}

}

**public** **static** **void** deleteProduct(**int** id) {

**if** (*inventory*.containsKey(id)) {

*inventory*.remove(id);

System.***out***.println("Product deleted.");

} **else** {

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

}

}

**public** **static** **void** displayInventory() {

System.***out***.println("Current Inventory:");

**for** (Product p : *inventory*.values()) {

System.***out***.println(p);

}

}

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

*addProduct*(**new** Product(101, "Mouse", 50, 299.99));

*addProduct*(**new** Product(102, "Keyboard", 30, 499.50));

*updateProduct*(101, 45, 279.99);

*deleteProduct*(102);

*displayInventory*();

}

}

Expected Output:

Product added.

Product added.

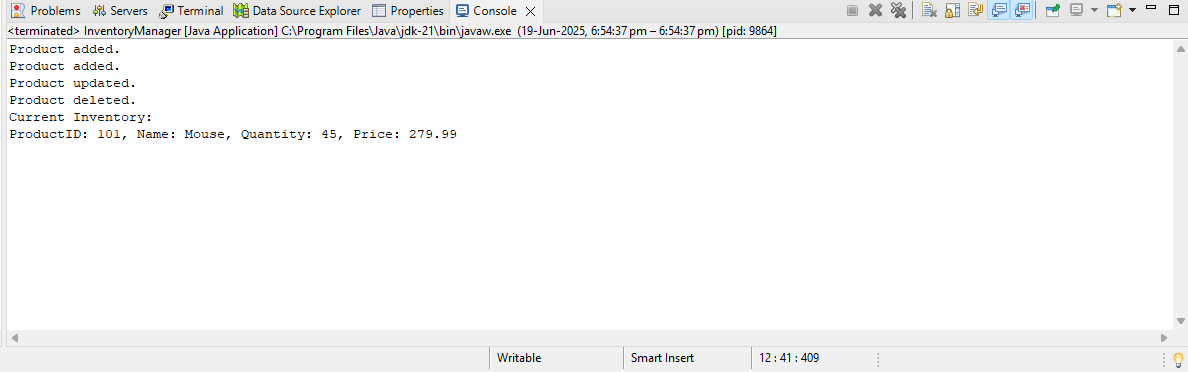
Product updated.

Product deleted.

Current Inventory:

ProductID: 101, Name: Mouse, Quantity: 45, Price: 279.99

Output:



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

**Project Name:** **EcommerceSearchSystem**

Product.java Code:

**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 + ")";

}

}

SearchFunction.java Code:

**import** java.util.Arrays;

**import** java.util.Comparator;

**public** **class** SearchFunction {

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

**while** (low <= high) {

**int** mid = (low + high) / 2;

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

**if** (result == 0)

**return** products[mid];

**else** **if** (result < 0)

low = mid + 1;

**else**

high = mid - 1;

}

**return** **null**;

}

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

Product[] products = {

**new** Product(101, "Laptop", "Electronics"),

**new** Product(102, "Shirt", "Clothing"),

**new** Product(103, "Book", "Stationery"),

**new** Product(104, "Shoes", "Footwear"),

**new** Product(105, "Watch", "Accessories")

};

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

String searchName = "Book";

System.***out***.println("Linear Search:");

Product foundLinear = *linearSearch*(products, searchName);

System.***out***.println(foundLinear != **null** ? "Found: " + foundLinear : "Not Found");

System.***out***.println("\nBinary Search:");

Product foundBinary = *binarySearch*(products, searchName);

System.***out***.println(foundBinary != **null** ? "Found: " + foundBinary : "Not Found");

}

}

Expected Output:

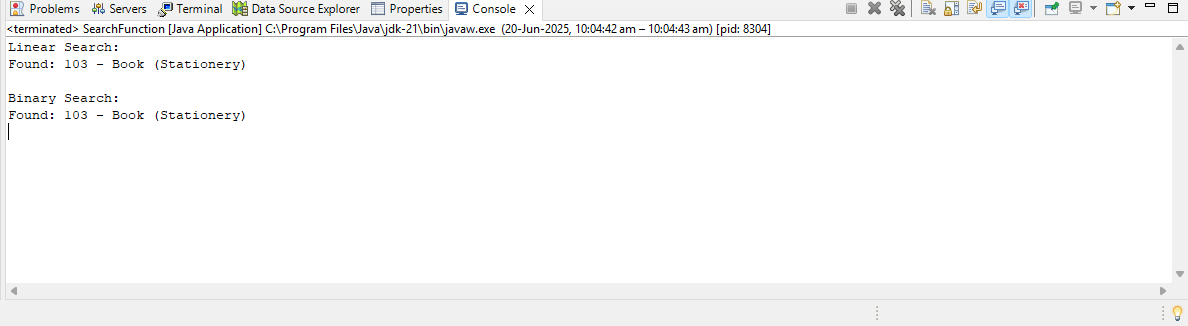
Linear Search:

Found: 103 - Book (Stationery)

Binary Search:

Found: 103 - Book (Stationery)

Output:



**Exercise 3: Sorting Customer Orders**

**Project Name: OrderSortingSystem**

Order.java Code:

**public** **class** Order {

**int** orderId;

String customerName;

**double** totalPrice;

**public** Order(**int** orderId, String customerName, **double** totalPrice) {

**this**.orderId = orderId;

**this**.customerName = customerName;

**this**.totalPrice = totalPrice;

}

**public** String toString() {

**return** "OrderID: " + orderId + ", Customer: " + customerName + ", TotalPrice: " + totalPrice;

}

}

SortingDemo.java Code:

**public** **class** SortingDemo {

**public** **static** **void** bubbleSort(Order[] orders) {

**int** n = orders.length;

**for** (**int** i = 0; i < n-1; i++) {

**for** (**int** j = 0; j < n-i-1; j++) {

**if** (orders[j].totalPrice > orders[j+1].totalPrice) {

Order temp = orders[j];

orders[j] = orders[j+1];

orders[j+1] = temp;

}

}

}

}

**public** **static** **void** quickSort(Order[] orders, **int** low, **int** high) {

**if** (low < high) {

**int** pi = *partition*(orders, low, high);

*quickSort*(orders, low, pi-1);

*quickSort*(orders, pi+1, high);

}

}

**public** **static** **int** partition(Order[] orders, **int** low, **int** high) {

**double** pivot = orders[high].totalPrice;

**int** i = low-1;

**for** (**int** j = low; j < high; j++) {

**if** (orders[j].totalPrice <= pivot) {

i++;

Order temp = orders[i];

orders[i] = orders[j];

orders[j] = temp;

}

}

Order temp = orders[i+1];

orders[i+1] = orders[high];

orders[high] = temp;

**return** i+1;

}

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

Order[] orders = {

**new** Order(101, "Alice", 2500),

**new** Order(102, "Bob", 500),

**new** Order(103, "Charlie", 4500),

**new** Order(104, "Daisy", 1000),

**new** Order(105, "Ethan", 1500)

};

System.***out***.println("Original orders:");

**for** (Order o : orders) System.***out***.println(o);

Order[] ordersBubble = orders.clone();

*bubbleSort*(ordersBubble);

System.***out***.println("\nBubble sorted orders:");

**for** (Order o : ordersBubble) System.***out***.println(o);

Order[] ordersQuick = orders.clone();

*quickSort*(ordersQuick, 0, ordersQuick.length-1);

System.***out***.println("\nQuick sorted orders:");

**for** (Order o : ordersQuick) System.***out***.println(o);

}

}

Expected Output:

Original orders:

OrderID: 101, Customer: Alice, TotalPrice: 2500.0

OrderID: 102, Customer: Bob, TotalPrice: 500.0

OrderID: 103, Customer: Charlie, TotalPrice: 4500.0

OrderID: 104, Customer: Daisy, TotalPrice: 1000.0

OrderID: 105, Customer: Ethan, TotalPrice: 1500.0

Bubble sorted orders:

OrderID: 102, Customer: Bob, TotalPrice: 500.0

OrderID: 104, Customer: Daisy, TotalPrice: 1000.0

OrderID: 105, Customer: Ethan, TotalPrice: 1500.0

OrderID: 101, Customer: Alice, TotalPrice: 2500.0

OrderID: 103, Customer: Charlie, TotalPrice: 4500.0

Quick sorted orders:

OrderID: 102, Customer: Bob, TotalPrice: 500.0

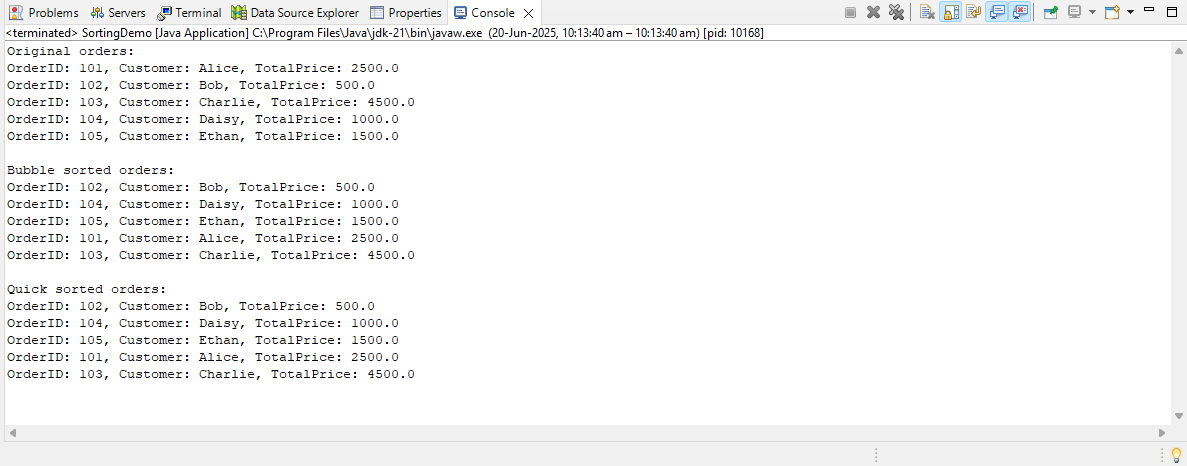
OrderID: 104, Customer: Daisy, TotalPrice: 1000.0

OrderID: 105, Customer: Ethan, TotalPrice: 1500.0

OrderID: 101, Customer: Alice, TotalPrice: 2500.0

OrderID: 103, Customer: Charlie, TotalPrice: 4500.0

Output:



**Exercise 4: Employee Management System**

**Project Name:** EmployeeManagementSystem

Employee.java Code:

**public** **class** Employee {

**int** employeeId;

String name;

String position;

**double** salary;

**public** Employee(**int** employeeId, String name, String position, **double** salary) {

**this**.employeeId = employeeId;

**this**.name = name;

**this**.position = position;

**this**.salary = salary;

}

**public** String toString() {

**return** "ID: " + employeeId + ", Name: " + name + ", Position: " + position + ", Salary: " + salary;

}

}

EmployeeManager.java Code:

**public** **class** EmployeeManager {

**static** Employee[] *employees* = **new** Employee[10];

**static** **int** *count* = 0;

**public** **static** **boolean** addEmployee(Employee e) {

**if** (*count* < *employees*.length) {

*employees*[*count*++] = e;

**return** **true**;

}

System.***out***.println("Array is full, cannot add more employees.");

**return** **false**;

}

**public** **static** Employee searchEmployee(**int** id) {

**for** (**int** i = 0; i < *count*; i++) {

**if** (*employees*[i].employeeId == id) {

**return** *employees*[i];

}

}

**return** **null**;

}

**public** **static** **void** traverseEmployees() {

System.***out***.println("\nCurrent Employees:");

**for** (**int** i = 0; i < *count*; i++) {

System.***out***.println(*employees*[i]);

}

}

**public** **static** **boolean** deleteEmployee(**int** id) {

**int** index = -1;

**for** (**int** i = 0; i < *count*; i++) {

**if** (*employees*[i].employeeId == id) {

index = i;

**break**;

}

}

**if** (index != -1) {

**for** (**int** i = index; i < *count*-1; i++) {

*employees*[i] = *employees*[i+1];

}

*employees*[*count*-1] = **null**;

*count*--;

**return** **true**;

}

**return** **false**;

}

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

*addEmployee*(**new** Employee(101, "Alice", "Developer", 50000));

*addEmployee*(**new** Employee(102, "Bob", "Designer", 45000));

*addEmployee*(**new** Employee(103, "Charlie", "Manager", 80000));

*traverseEmployees*();

Employee found = *searchEmployee*(102);

System.***out***.println("\nSearch result: " + (found != **null** ? found : "Not found"));

*deleteEmployee*(102);

*traverseEmployees*();

}

}

Expected Output:

Current Employees:

ID: 101, Name: Alice, Position: Developer, Salary: 50000.0

ID: 102, Name: Bob, Position: Designer, Salary: 45000.0

ID: 103, Name: Charlie, Position: Manager, Salary: 80000.0

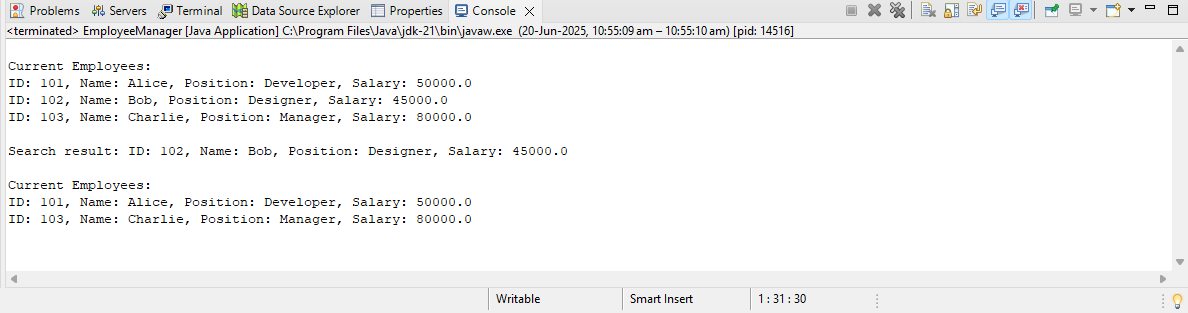
Search result: ID: 102, Name: Bob, Position: Designer, Salary: 45000.0

Current Employees:

ID: 101, Name: Alice, Position: Developer, Salary: 50000.0

ID: 103, Name: Charlie, Position: Manager, Salary: 80000.0

Output:



**Exercise 5: Task Management System**

**Project Name:TaskManagementSystem**

Task.java Code:

**public** **class** Task {

**int** taskId;

String taskName;

String status;

**public** Task(**int** taskId, String taskName, String status) {

**this**.taskId = taskId;

**this**.taskName = taskName;

**this**.status = status;

}

**public** String toString() {

**return** "TaskID: " + taskId + ", Name: " + taskName + ", Status: " + status;

}

}

TaskManager.java Code:

**class** Node {

Task data;

Node next;

**public** Node(Task data) {

**this**.data = data;

**this**.next = **null**;

}

}

**public** **class** TaskManager {

Node head = **null**;

**public** **void** addTask(Task task) {

Node newNode = **new** Node(task);

**if** (head == **null**) {

head = newNode;

} **else** {

Node temp = head;

**while** (temp.next != **null**) {

temp = temp.next;

}

temp.next = newNode;

}

}

**public** Task searchTask(**int** taskId) {

Node temp = head;

**while** (temp != **null**) {

**if** (temp.data.taskId == taskId) {

**return** temp.data;

}

temp = temp.next;

}

**return** **null**;

}

**public** **void** traverseTasks() {

Node temp = head;

System.***out***.println("\nTasks List:");

**while** (temp != **null**) {

System.***out***.println(temp.data);

temp = temp.next;

}

}

**public** **boolean** deleteTask(**int** taskId) {

**if** (head == **null**) **return** **false**;

**if** (head.data.taskId == taskId) {

head = head.next;

**return** **true**;

}

Node curr = head;

Node prev = **null**;

**while** (curr != **null** && curr.data.taskId != taskId) {

prev = curr;

curr = curr.next;

}

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

prev.next = curr.next;

**return** **true**;

}

**return** **false**;

}

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

TaskManager tm = **new** TaskManager();

tm.addTask(**new** Task(1, "Design UI", "Pending"));

tm.addTask(**new** Task(2, "Develop Backend", "In Progress"));

tm.addTask(**new** Task(3, "Test Application", "Completed"));

tm.traverseTasks();

Task found = tm.searchTask(2);

System.***out***.println("\nFound task: " + (found != **null** ? found : "Not Found"));

tm.deleteTask(1);

tm.traverseTasks();

}

}

Expected Output:

Tasks List:

TaskID: 1, Name: Design UI, Status: Pending

TaskID: 2, Name: Develop Backend, Status: In Progress

TaskID: 3, Name: Test Application, Status: Completed

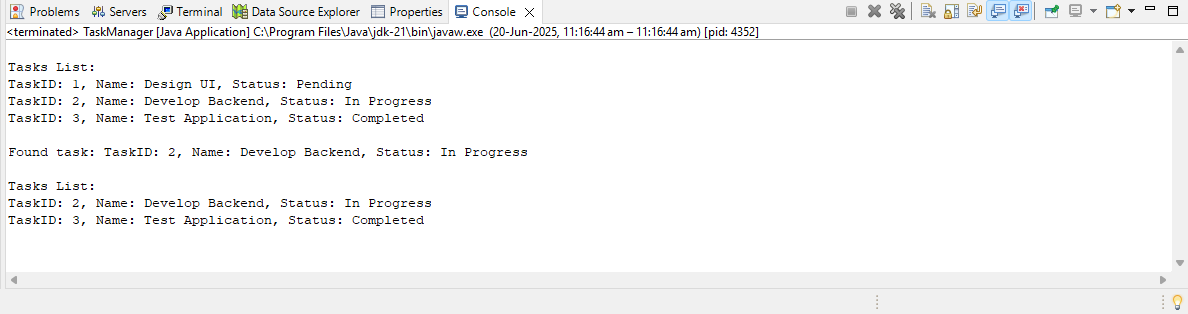
Found task: TaskID: 2, Name: Develop Backend, Status: In Progress

Tasks List:

TaskID: 2, Name: Develop Backend, Status: In Progress

TaskID: 3, Name: Test Application, Status: Completed

Output:



**Exercise 6: Library Management System**

**Project Name:LibraryManagementSystem**

Book.java Code:

**public** **class** Book {

**int** bookId;

String title;

String author;

**public** Book(**int** bookId, String title, String author) {

**this**.bookId = bookId;

**this**.title = title;

**this**.author = author;

}

**public** String toString() {

**return** "ID: " + bookId + ", Title: " + title + ", Author: " + author;

}

}

LibrarySearchDemo.java Code:

**import** java.util.Arrays;

**import** java.util.Comparator;

**public** **class** LibrarySearchDemo {

**public** **static** Book linearSearch(Book[] books, String title) {

**for** (Book b : books) {

**if** (b.title.equalsIgnoreCase(title)) {

**return** b;

}

}

**return** **null**;

}

**public** **static** Book binarySearch(Book[] books, String title) {

**int** low = 0, high = books.length - 1;

**while** (low <= high) {

**int** mid = (low + high) / 2;

**int** cmp = books[mid].title.compareToIgnoreCase(title);

**if** (cmp == 0) {

**return** books[mid];

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

low = mid + 1;

} **else** {

high = mid - 1;

}

}

**return** **null**;

}

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

Book[] books = {

**new** Book(101, "Java Basics", "Alice"),

**new** Book(102, "Data Structures", "Bob"),

**new** Book(103, "Algorithms 101", "Charlie"),

**new** Book(104, "Design Patterns", "Diana")

};

Arrays.*sort*(books, Comparator.*comparing*(b -> b.title));

String titleToFind = "Data Structures";

System.***out***.println("=== Linear Search ===");

Book foundLinear = *linearSearch*(books, titleToFind);

System.***out***.println(foundLinear != **null** ? "Found: " + foundLinear : "Not found");

System.***out***.println("\n=== Binary Search ===");

Book foundBinary = *binarySearch*(books, titleToFind);

System.***out***.println(foundBinary != **null** ? "Found: " + foundBinary : "Not found");

}

}

Excepted Output:

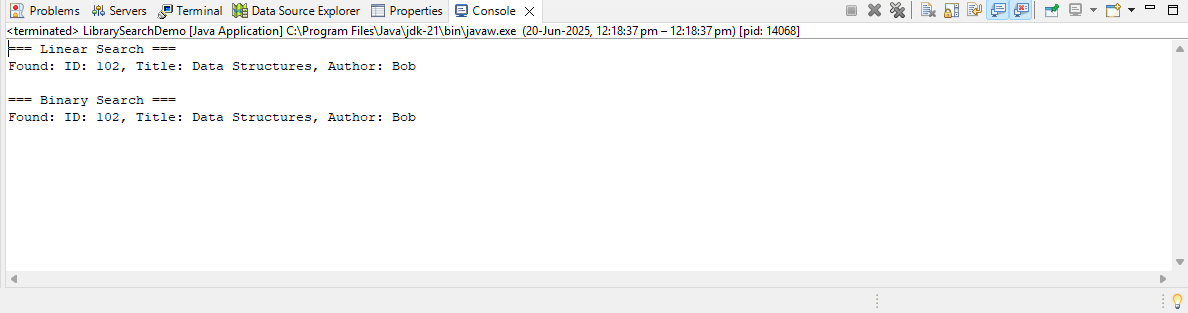
=== Linear Search ===

Found: ID: 102, Title: Data Structures, Author: Bob

=== Binary Search ===

Found: ID: 102, Title: Data Structures, Author: Bob

Output:



**Exercise 7: Financial Forecasting**

**Project Name: FinancialForecastingTool**

ForecastingDemo.java Code:

**public** **class** ForecastingDemo {

**public** **static** **double** futureValue(**double** presentValue, **double** rate, **int** years) {

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

**return** presentValue;

}

**return** *futureValue*(presentValue, rate, years - 1) \* (1 + rate);

}

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

**double** presentValue = 1000;

**double** rate = 0.05;

**int** years = 5;

**double** future = *futureValue*(presentValue, rate, years);

System.***out***.println("Future Value after " + years + " years: $" + future);

}

}

Expected Output:

Future Value after 5 years: $1276.2815625000003

Output:

