**Exercise 1: Inventory Management System**

**Scenario:**

You are developing an inventory management system for a warehouse. Efficient data storage and retrieval are crucial.

**Solution:**

package InventoryManagement;

public class Product {

private int productId;

private String productName;

private int quantity;

private 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 int getProductId() {

return productId;

}

public String getProductName() {

return productName;

}

public int getQuantity() {

return quantity;

}

public double getPrice() {

return price;

}

public void setProductName(String productName) {

this.productName = productName;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public void setPrice(double price) {

this.price = price;

}

public String toString() {

return "Product Id = "+productId+", Product Name = "+productName+", Quantity = "+quantity+", Price Rs: "+price;

}

}

package InventoryManagement;

import java.util.\*;

public class Storing {

private HashMap<Integer, Product> mp=new HashMap<>();

public void add(Product product) {

mp.put(product.getProductId(), product);

}

public void update(int productId,String productName,int quantity,double price) {

Product product = mp.get(productId);

if(product!=null){

product.setProductName(productName);

product.setQuantity(quantity);

product.setPrice(price);

}else {

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

}

}

public void delete(int productId) {

if(mp.remove(productId) == null)

{

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

}

}

public void displayProducts() {

for(Product p: mp.values()) {

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

}

}

}

package InventoryManagement;

public class Main {

public static void main(String[] args) {

Storing s= new Storing();

s.add(new Product(1,"Chip",50,0.25));

s.add(new Product(2,"Battery",100,500));

s.displayProducts();

s.update(1, "Microchip", 50, 0.50);

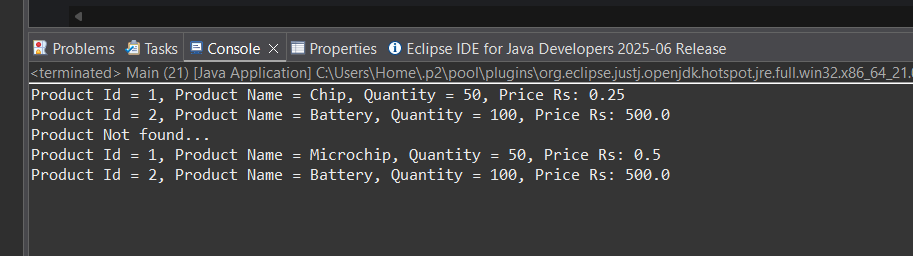
s.delete(3);

s.displayProducts();

}

}

**Output:**



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

**Scenario:**

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**Solution:**

package EcommercePlatform;

public class Product {

private int productId;

private String productName;

private String category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

public int getProductId() {

return productId;

}

public String getProductName() {

return productName;

}

public String getCategory() {

return category;

}

public String toString() {

return "Product ID = "+productId+", Product Name = "+productName+", Category = "+category;

}

}

package EcommercePlatform;

public class Search {

public static Product linersearch(Product[] products, String target) {

for(Product product: products) {

if(product.getProductName().equalsIgnoreCase(target)){

return product;

}

}

return null;

}

public static Product binarysearch(Product[] products, String target) {

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

while(left<=right) {

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

int compare= target.compareToIgnoreCase(products[mid].getProductName());

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

else if(compare < 0) right = mid-1;

else left=mid+1;

}

return null;

}

}

package EcommercePlatform;

import java.util.\*;

public class sortbinary {

public static void sortProductName(Product[] products) {

Arrays.*sort*(products, Comparator.*comparing*(Product:: getProductName, String.***CASE\_INSENSITIVE\_ORDER***));

}

}

package EcommercePlatform;

public class Main {

public static void main(String[] args) {

Product[] products = {

new Product(1,"Laptop","Electronics"),

new Product(2,"Shoes","Footwear"),

new Product(3,"Watch","Accessories"),

new Product(4,"Mobile","Electronics")

};

Product linear= Search.*linersearch*(products, "Laptop");

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

sortbinary.*sortProductName*(products);

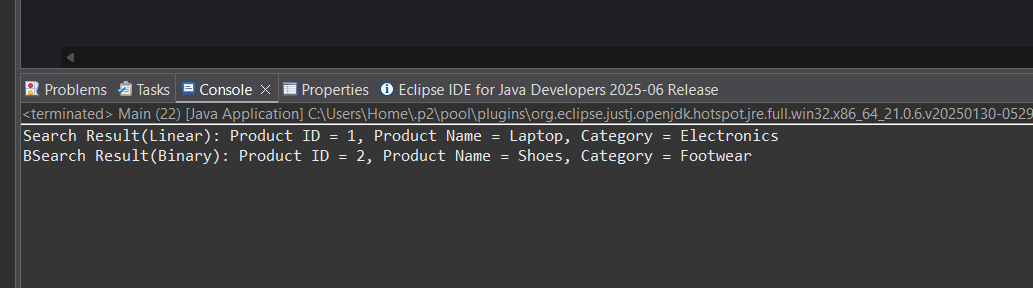
Product binary=Search.*binarysearch*(products, "Shoes");

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

}

}

**Output:**



**Exercise 3: Sorting Customer Orders**

**Scenario:**

You are tasked with sorting customer orders by their total price on an e-commerce platform. This helps in prioritizing high-value orders.

**Solution:**

package CustomerOrders;

public class Product {

private int productId;

private String customerName;

private double totalPrice;

public Product(int productId, String customerName, double totalPrice) {

this.productId = productId;

this.customerName = customerName;

this.totalPrice = totalPrice;

}

public int getProductId() {

return productId;

}

public String getCustomerName() {

return customerName;

}

public double getTotalPrice() {

return totalPrice;

}

public String toString() {

return "Product ID = " + productId + ", Name = " + customerName + ", Total = ₹" + totalPrice;

}

}

package CustomerOrders;

public class Sorting {

public static void bubble(Product[] product) {

int n=product.length;

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

boolean swap=false;

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

if(product[j].getTotalPrice() < product[j+1].getTotalPrice()) {

Product temp=product[j];

product[j]=product[j+1];

product[j+1]=temp;

swap=true;

}

}

if(!swap) break;

}

}

public static void quick(Product[] product,int low,int high) {

if(low < high) {

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

*quick*(product,low,pi-1);

*quick*(product,pi+1,high);

}

}

private static int partition(Product[] product,int low,int high) {

double pivot= product[high].getTotalPrice();

int i=low-1;

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

if(product[j].getTotalPrice() > pivot) {

i++;

Product temp = product[i];

product[i] = product[j];

product[j] = temp;

}

}

Product temp = product[i + 1];

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

product[high] = temp;

return i + 1;

}

}

package CustomerOrders;

public class Main {

public static void printProducts(Product[] products) {

for (Product p : products) {

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

}

}

public static void main(String[] args) {

Product[] products1 = {

new Product(101, "Tarun", 750.0),

new Product(102, "Viswa", 1200.0),

new Product(103, "Vignesh", 500.0),

new Product(104, "Sankar", 950.0)

};

System.***out***.println("Bubble Sort(Descending by Price):");

Sorting.*bubble*(products1);

*printProducts*(products1);

Product[] products2 = {

new Product(101, "Tarun", 750.0),

new Product(102, "Viswa", 1200.0),

new Product(103, "Vignesh", 500.0),

new Product(104, "Sankar", 950.0)

};

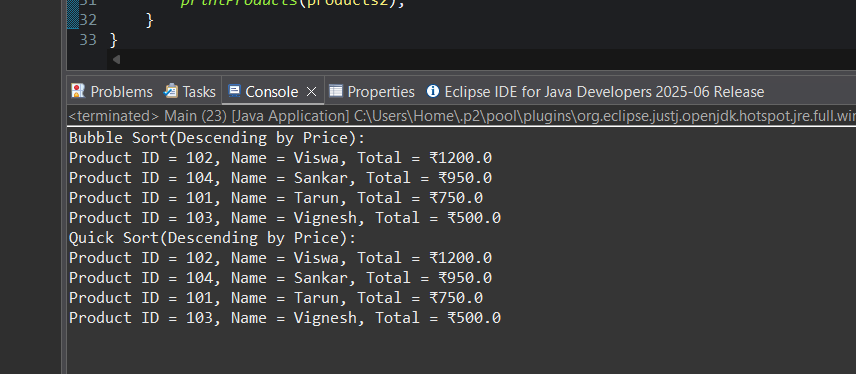
System.***out***.println("Quick Sort(Descending by Price):");

Sorting.*quick*(products2, 0, products2.length - 1);

*printProducts*(products2);

}

}

**Output:**

**Exercise 4: Employee Management System**

**Scenario:**

You are developing an employee management system for a company. Efficiently managing employee records is crucial.

**Solution:**

package EmployeeManagement;

public class Employee {

private int EmployeeId;

private String name;

private String position;

private 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 int getEmployeeId() {

return EmployeeId;

}

public String getName() {

return name;

}

public String getPosition() {

return position;

}

public double getSalary() {

return salary;

}

public String toString() {

return "EmployeeId = "+EmployeeId+", Name = "+name+", Position = "+position+", Salary = ₹"+salary;

}

}

package EmployeeManagement;

public class EmployeeOperations {

private Employee[] employees;

private int count;

public EmployeeOperations(int size) {

employees= new Employee[size];

int count=0;

}

public boolean add(Employee e) {

if(count >= employees.length) return false;

employees[count++] = e;

return true;

}

public boolean delete(int id) {

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

if(employees[i].getEmployeeId() == id) {

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

employees[j]=employees[j+1];

}

employees[--count]=null;

return true;

}

}

return false;

}

public Employee search(int id) {

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

if(employees[i].getEmployeeId() == id)

return employees[i];

}

return null;

}

public void display() {

if(count==0) {

System.***out***.println("No employees"); return;

}

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

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

}

}

}

package EmployeeManagement;

public class Main {

public static void main(String[] args) {

EmployeeOperations eo=new EmployeeOperations(5);

eo.add(new Employee(1,"Tarun","Engineer",7500000));

eo.add(new Employee(2,"Raja","Manager",3500000));

eo.add(new Employee(3,"Kishore","HR",6000000));

eo.display();

Employee found= eo.search(2);

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

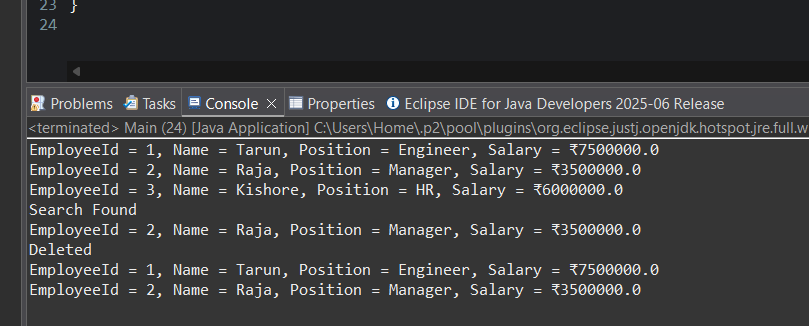
System.***out***.println(eo.delete(3) ? "Deleted" : "Not Found");

eo.display();

}

}

**Output:**

****

**Exercise 5: Task Management System**

**Scenario:**

You are developing a task management system where tasks need to be added, deleted, and traversed efficiently.

**Solution:**

package TaskManagement;

public class Task {

private int taskId;

private String taskName;

private String status;

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

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

}

public int getTaskId() {

return taskId;

}

public String getTaskName() {

return taskName;

}

public String getStatus() {

return status;

}

public String toString() {

return "Task [taskId=" + taskId + ", taskName=" + taskName + ", status=" + status + "]";

}

}

package TaskManagement;

public class TaskLinkedList {

private Node head;

private class Node{

Task task;

Node next;

Node(Task task){

this.task=task;

this.next=null;

}

}

public void add(Task task) {

Node newnode=new Node(task);

if(head==null) head= newnode;

else {

Node curr=head;

while(curr.next!=null) {

curr=curr.next;

}

curr.next=newnode;

}

}

public Task search(int taskId) {

Node curr=head;

while(curr!=null) {

if(curr.task.getTaskId() == taskId) {

return curr.task;

}

curr=curr.next;

}

return null;

}

public boolean delete(int taskId) {

if(head==null) return false;

if(head.task.getTaskId() == taskId) {

head=head.next;

return true;

}

Node curr=head;

while(curr.next!=null && curr.next.task.getTaskId()!=taskId) {

curr=curr.next;

}

if(curr.next!=null) {

curr.next=curr.next.next;

return true;

}

return false;

}

public void display() {

if(head==null) {

System.***out***.println("No Tasks available..."); return;

}

Node curr=head;

while(curr!=null) {

System.***out***.println(curr.task);

curr=curr.next;

}

}

}

package TaskManagement;

public class Main {

public static void main(String[] args) {

TaskLinkedList tl=new TaskLinkedList();

tl.add(new Task(1,"Design Frontend","Pending"));

tl.add(new Task(2,"Backend Connectivity","In Progess"));

tl.add(new Task(3,"Design Flows","Completed"));

tl.display();

Task found = tl.search(2);

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

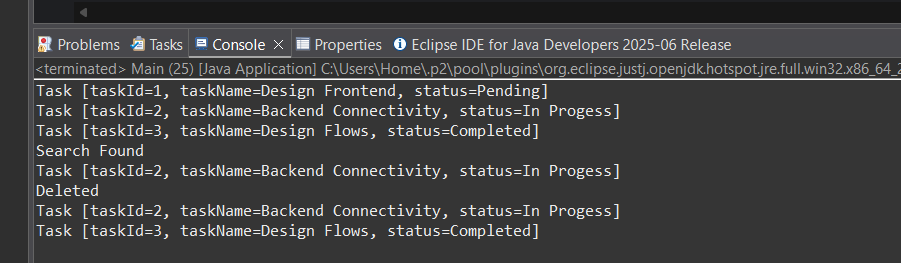
System.***out***.println(tl.delete(1) ? "Deleted" : "Not Found");

tl.display();

}

}

**Output:**

****

**Exercise 6: Library Management System**

**Scenario:**

You are developing a library management system where users can search for books by title or author.

**Solution:**

package LibraryManagement;

public class Book {

private int bookId;

private String title;

private String authorName;

public Book(int bookId, String title, String authorName) {

this.bookId = bookId;

this.title = title;

this.authorName = authorName;

}

public int getBookId() {

return bookId;

}

public String getTitle() {

return title;

}

public String getAuthorName() {

return authorName;

}

public String toString() {

return "Book [bookId=" + bookId + ", title=" + title + ", authorName=" + authorName + "]";

}

}

package LibraryManagement;

import java.util.\*;

public class Search {

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

for(Book b: books) {

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

return b;

}

}

return null;

}

public static void sortTitle(Book[] books) {

Arrays.*sort*(books,Comparator.*comparing*(Book::getTitle,String.***CASE\_INSENSITIVE\_ORDER***));

}

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

int left=0,right=books.length-1;

while(left <= right) {

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

int compare=title.compareToIgnoreCase(books[mid].getTitle());

if(compare == 0) return books[mid];

else if(compare <0) right=mid-1;

else left=mid+1;

}

return null;

}

}

package LibraryManagement;

public class Main {

public static void main(String[] args) {

Book[] books = {

new Book(1,"Wings of Fire","Dr.A.P.J Abdul kalam"),

new Book(2,"How to Win Friends and Influence People","Dale Carnegie"),

new Book(3,"The Alchemist","Paulo Coelho"),

new Book(4,"Who Will Cry When you Die?","Robin Sharma")

};

Book linearfound = Search.*linear*(books, "The Alchemist");

System.***out***.println(linearfound!=null ? "Search Found\n"+linearfound : "Not found");

Search.*sortTitle*(books);

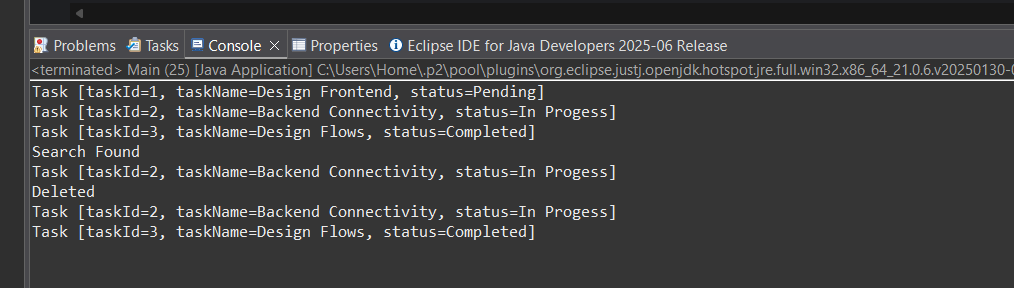
Book binaryfound = Search.*linear*(books, "Wings of Fire");

System.***out***.println(binaryfound!=null ? "Search Found\n"+binaryfound : "Not Found");

}

}

**Output:**

****

**Exercise 7: Financial Forecasting**

**Scenario:**

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

**Solution:**

package FinancialForecasting;

public class Financial {

public static double Value(double principal,double rate,int years) {

if(years == 0) {

return principal;

}

return (1+rate)\**Value*(principal,rate,years-1);

}

public static double Memo(double principal,double rate,int years,Double[] memo) {

if(years == 0) return principal;

if(memo[years]!=null) return memo[years];

memo[years] = (1+rate)\* *Memo*(principal,rate,years-1,memo);

return memo[years];

}

}

package FinancialForecasting;

public class Main {

public static void main(String[] args) {

double principal = 850000;

double rate = 0.08;

int years = 5;

double future = Financial.*Value*(principal, rate, years);

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

Double[] memo = new Double[years+1];

double ValueMemo= Financial.*Memo*(principal, rate, years, memo);

System.***out***.printf("Memoized Future Value: ₹%.2f\n", ValueMemo);

}

}

**Output:**

