WEEK-1

MODULE-2

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

**package** EcommercePlatformSearchFunction;

**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;

}

@Override

**public** String toString() {

**return** "Product{id=" + productId + ", name='" + productName + "', category='" + category + "'}";

}

}

**package** EcommercePlatformSearchFunction;

**import** java.util.Arrays;

**import** java.util.Comparator;

**public** **class** ECommerceSearch {

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

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

**if** (p.getProductId() == id) {

**return** p;

}

}

**return** **null**;

}

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

**int** low = 0;

**int** high = products.length - 1;

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

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

**if** (products[mid].getProductId() == id) {

**return** products[mid];

} **else** **if** (products[mid].getProductId() < id) {

low = mid + 1;

} **else** {

high = mid - 1;

}

}

**return** **null**;

}

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

// Sample Products

Product[] products = {

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

**new** Product(105, "Mouse", "Electronics"),

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

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

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

};

System.***out***.println("\nOriginal Products:");

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

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

}

System.***out***.println("\nPerforming Linear Search for ID 103:");

Product result1 = *linearSearch*(products, 103);

System.***out***.println(result1 != **null** ? "Found: " + result1 : "Not Found");

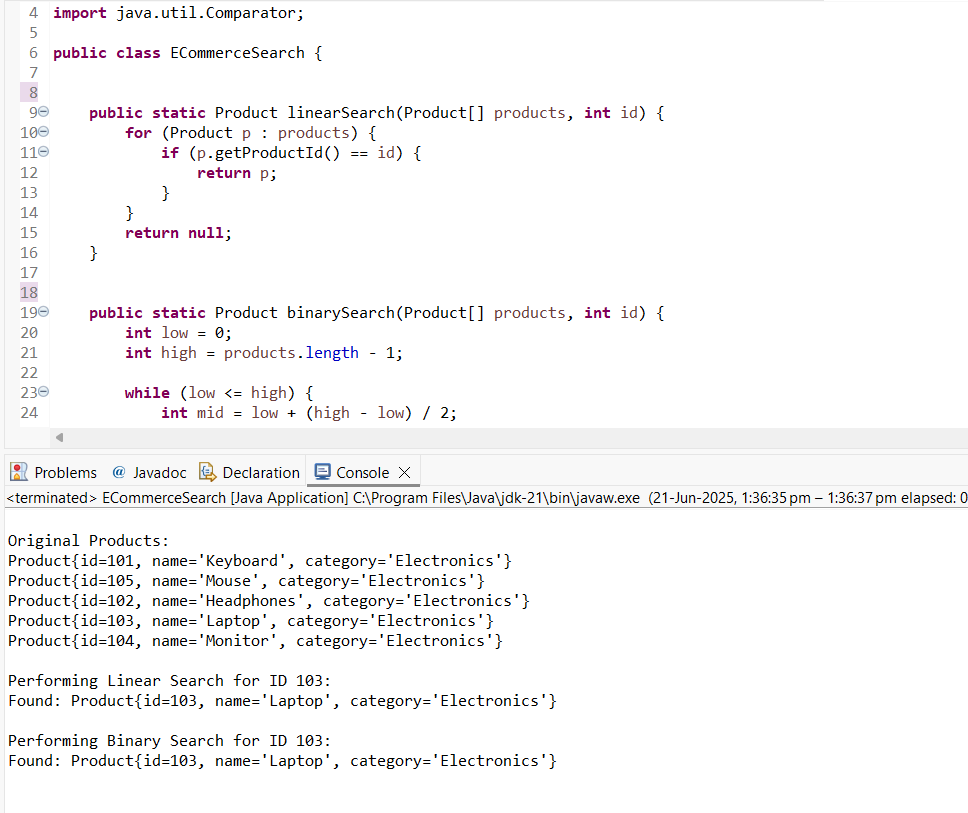
Arrays.*sort*(products, Comparator.*comparingInt*(Product::getProductId));

System.***out***.println("\nPerforming Binary Search for ID 103:");

Product result2 = *binarySearch*(products, 103);

System.***out***.println(result2 != **null** ? "Found: " + result2 : "Not Found");

}

}

**Exercise 7: Financial Forecasting**

**package** FinancialForecasting;

**public** **class** FinancialForecast {

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

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

**return** initialValue;

}

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

}

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

**double** initialInvestment = 1000.0;

**double** annualGrowthRate = 0.05; // 5% annual growth

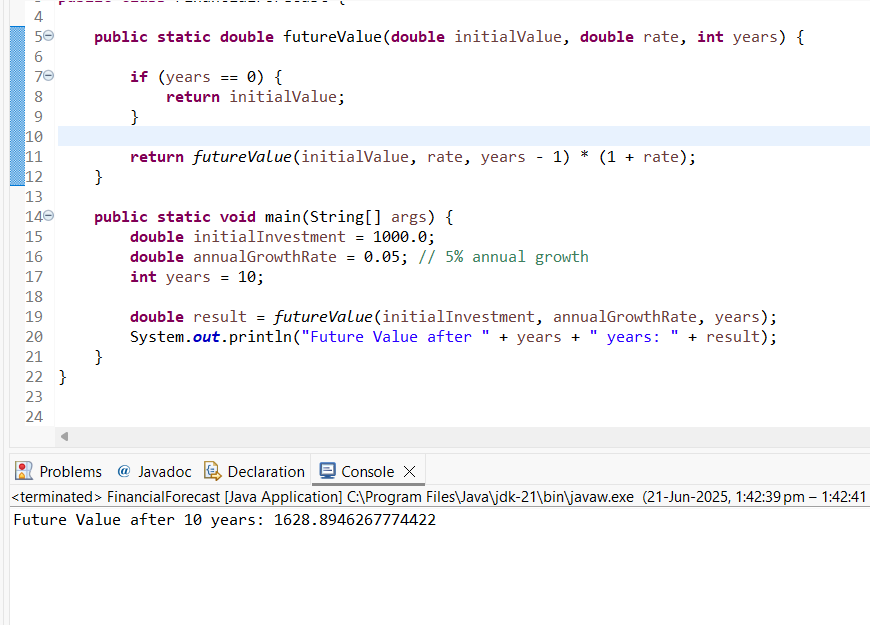
**int** years = 10;

**double** result = *futureValue*(initialInvestment, annualGrowthRate, years);

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

}

}



**Exercise 6: Library Management System**

package LibraryManagementSystem;

public class Book {

private int bookId;

private String title;

private String author;

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

this.bookId = bookId;

this.title = title;

this.author = author;

}

public int getBookId() {

return bookId;

}

public String getTitle() {

return title;

}

public String getAuthor() {

return author;

}

@Override

public String toString() {

return "Book{id=" + bookId + ", title='" + title + "', author='" + author + "'}";

}

}

**package** LibraryManagementSystem;

**import** java.util.Arrays;

**import** java.util.Comparator;

**public** **class** LibrarySearch {

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

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

**if** (book.getTitle().equalsIgnoreCase(title)) {

**return** book;

}

}

**return** **null**;

}

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

**int** low = 0;

**int** high = books.length - 1;

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

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

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

**if** (compareResult == 0) {

**return** books[mid];

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

low = mid + 1;

} **else** {

high = mid - 1;

}

}

**return** **null**;

}

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

Book[] books = {

**new** Book(101, "The Alchemist", "Paulo Coelho"),

**new** Book(102, "1984", "George Orwell"),

**new** Book(103, "Brave New World", "Aldous Huxley"),

**new** Book(104, "To Kill a Mockingbird", "Harper Lee"),

**new** Book(105, "Moby-Dick", "Herman Melville")

};

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

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

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

}

System.***out***.println("\nSearching for '1984' using Linear Search:");

Book result1 = *linearSearch*(books, "1984");

System.***out***.println(result1 != **null** ? "Found: " + result1 : "Not Found");

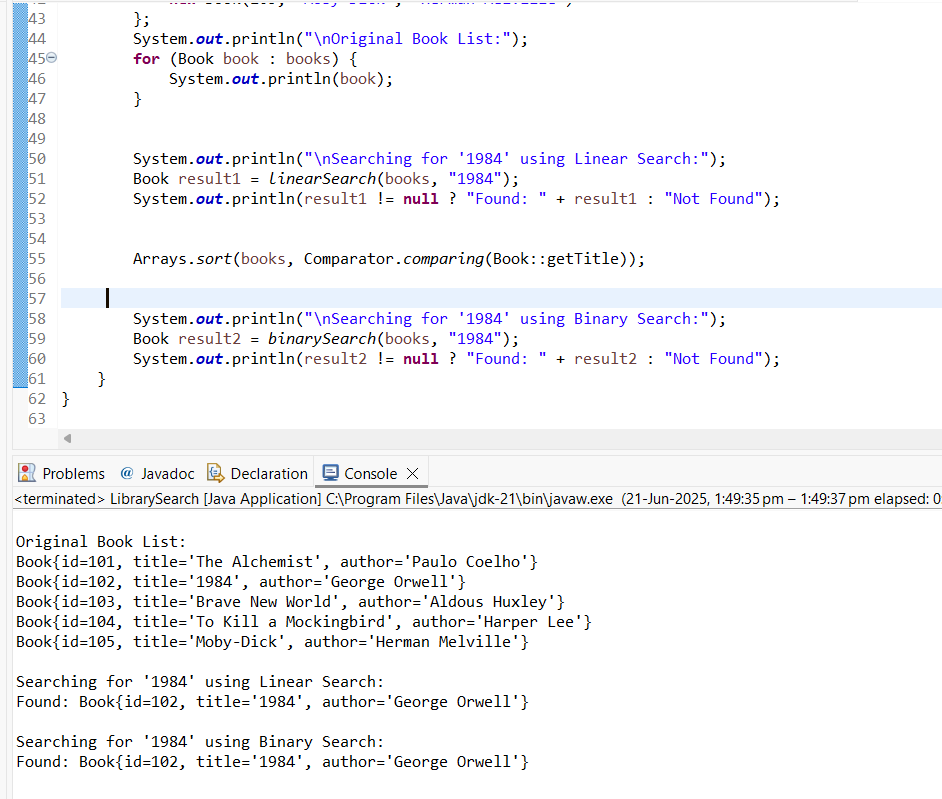
Arrays.*sort*(books, Comparator.*comparing*(Book::getTitle));

System.***out***.println("\nSearching for '1984' using Binary Search:");

Book result2 = *binarySearch*(books, "1984");

System.***out***.println(result2 != **null** ? "Found: " + result2 : "Not Found");

}

}

**Exercise 5: Task Management System**

package TaskManagementSystem;

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;

}

@Override

public String toString() {

return "Task{id=" + taskId + ", name='" + taskName + "', status='" + status + "'}";

}

}

**package** TaskManagementSystem;

**public** **class** TaskLinkedList {

**private** Node head;

**private** **static** **class** Node {

Task task;

Node next;

Node(Task task) {

**this**.task = task;

}

}

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

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

newNode.next = head;

head = newNode;

}

**public** Task search(**int** id) {

Node current = head;

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

**if** (current.task.getTaskId() == id) {

**return** current.task;

}

current = current.next;

}

**return** **null**;

}

**public** **void** traverse() {

Node current = head;

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

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

current = current.next;

}

}

**public** **void** delete(**int** id) {

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

**if** (head.task.getTaskId() == id) {

head = head.next;

**return**;

}

Node current = head;

**while** (current.next != **null** && current.next.task.getTaskId() != id) {

current = current.next;

}

**if** (current.next != **null**) {

current.next = current.next.next;

}

}

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

TaskLinkedList list = **new** TaskLinkedList();

list.add(**new** Task(101, "Design database schema", "Pending"));

list.add(**new** Task(102, "Develop REST APIs", "In Progress"));

list.add(**new** Task(103, "Write unit tests", "Completed"));

list.add(**new** Task(104, "Perform deployment", "Pending"));

System.***out***.println("\nAll Tasks:");

list.traverse();

System.***out***.println("\nSearching for Task ID 102:");

System.***out***.println(list.search(102));

System.***out***.println("\nDeleting Task ID 103");

list.delete(103);

System.***out***.println("\nAll Tasks After Deletion:");

list.traverse();

}

}

