**Exercise 1: Inventory Management System**

java

CopyEdit

import java.util.\*;

class Product {

int productId;

String productName;

int quantity;

double price;

Product(int id, String name, int qty, double pr) {

this.productId = id;

this.productName = name;

this.quantity = qty;

this.price = pr;

}

}

class InventorySystem {

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

void addProduct(Product p) {

inventory.put(p.productId, p);

}

void updateProduct(int id, int qty, double price) {

if (inventory.containsKey(id)) {

Product p = inventory.get(id);

p.quantity = qty;

p.price = price;

}

}

void deleteProduct(int id) {

inventory.remove(id);

}

}

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

java

CopyEdit

class ProductSearch {

int productId;

String productName;

String category;

ProductSearch(int id, String name, String cat) {

this.productId = id;

this.productName = name;

this.category = cat;

}

static int linearSearch(ProductSearch[] arr, String name) {

for (int i = 0; i < arr.length; i++) {

if (arr[i].productName.equals(name)) return i;

}

return -1;

}

static int binarySearch(ProductSearch[] arr, String name) {

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

while (low <= high) {

int mid = (low + high) / 2;

int cmp = arr[mid].productName.compareTo(name);

if (cmp == 0) return mid;

else if (cmp < 0) low = mid + 1;

else high = mid - 1;

}

return -1;

}

}

**Exercise 3: Sorting Customer Orders**

java

CopyEdit

class Order {

int orderId;

String customerName;

double totalPrice;

Order(int id, String name, double price) {

this.orderId = id;

this.customerName = name;

this.totalPrice = price;

}

}

class OrderSorting {

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;

}

}

}

}

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);

}

}

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;

}

}

**Exercise 4: Employee Management System**

java

CopyEdit

class Employee {

int employeeId;

String name;

String position;

double salary;

Employee(int id, String name, String pos, double sal) {

this.employeeId = id;

this.name = name;

this.position = pos;

this.salary = sal;

}

}

class EmployeeSystem {

Employee[] employees = new Employee[100];

int count = 0;

void add(Employee e) {

employees[count++] = e;

}

int search(int id) {

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

if (employees[i].employeeId == id) return i;

}

return -1;

}

void traverse() {

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

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

}

}

void delete(int id) {

int idx = search(id);

if (idx != -1) {

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

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

}

count--;

}

}

}

**Exercise 5: Task Management System**

java

CopyEdit

class Task {

int taskId;

String taskName;

String status;

Task next;

Task(int id, String name, String stat) {

this.taskId = id;

this.taskName = name;

this.status = stat;

this.next = null;

}

}

class TaskList {

Task head;

void add(Task t) {

if (head == null) head = t;

else {

Task temp = head;

while (temp.next != null) temp = temp.next;

temp.next = t;

}

}

Task search(int id) {

Task temp = head;

while (temp != null) {

if (temp.taskId == id) return temp;

temp = temp.next;

}

return null;

}

void traverse() {

Task temp = head;

while (temp != null) {

System.out.println(temp.taskName);

temp = temp.next;

}

}

void delete(int id) {

if (head == null) return;

if (head.taskId == id) {

head = head.next;

return;

}

Task temp = head;

while (temp.next != null && temp.next.taskId != id) {

temp = temp.next;

}

if (temp.next != null) {

temp.next = temp.next.next;

}

}

}

**Exercise 6:**

**Library Management System**

java

CopyEdit

class Book {

int bookId;

String title;

String author;

Book(int id, String title, String author) {

this.bookId = id;

this.title = title;

this.author = author;

}

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

for (int i = 0; i < books.length; i++) {

if (books[i].title.equals(title)) return i;

}

return -1;

}

static int 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.compareTo(title);

if (cmp == 0) return mid;

else if (cmp < 0) low = mid + 1;

else high = mid - 1;

}

return -1;

}

}

**Exercise 7: Financial Forecasting**

java

CopyEdit

class FinancialForecast {

static double predict(double initialValue, double growthRate, int years) {

if (years == 0) return initialValue;

return predict(initialValue \* (1 + growthRate), growthRate, years - 1);

}

}