Exercise 1: Inventory Management System

package dsa1;

import java.util.\*;

class Product {

int productId;

String productName;

int quantity;

double price;

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

this.productId = id;

this.productName = name;

this.quantity = qty;

this.price = price;

}

}

public class InventorySystem {

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

static void addProduct(Product p) {

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

}

static void updateProduct(int id, int newQty, double newPrice) {

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

if (p != null) {

p.quantity = newQty;

p.price = newPrice;

}

}

static void deleteProduct(int id) {

*inventory*.remove(id);

}

public static void main(String[] args) {

*addProduct*(new Product(1, "Pen", 100, 10.5));

*addProduct*(new Product(2, "Notebook", 50, 40));

*updateProduct*(1, 150, 11.0);

*deleteProduct*(2);

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

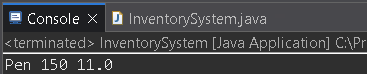
System.***out***.println(p.productName + " " + p.quantity + " " + p.price);

}

}

}

Output:



Exercise 2: E-commerce Platform Search Function

package dsa2;

class Product {

int productId;

String productName;

String category;

Product(int id, String name, String category) {

this.productId = id;

this.productName = name;

this.category = category;

}

}

public class ProductSearch {

static int linearSearch(Product[] products, String name) {

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

if (products[i].productName.equals(name)) {

return i;

}

}

return -1;

}

static int binarySearch(Product[] products, String name) {

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

while (low <= high) {

int mid = (low + high) / 2;

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

if (cmp == 0) return mid;

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

else high = mid - 1;

}

return -1;

}

public static void main(String[] args) {

Product[] products = {

new Product(1, "Book", "Education"),

new Product(2, "Laptop", "Electronics"),

new Product(3, "Pen", "Stationery")

};

Product[] sortedProducts = {products[0], products[2], products[1]};

java.util.Arrays.*sort*(sortedProducts, (a, b) -> a.productName.compareTo(b.productName));

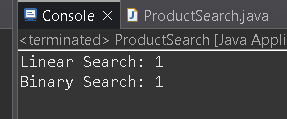
System.***out***.println("Linear Search: " + *linearSearch*(products, "Laptop"));

System.***out***.println("Binary Search: " + *binarySearch*(sortedProducts, "Laptop"));

}

}

Output:



Exercise 3: Sorting Customer Orders

package dsa3;

class Order {

int orderId;

String customerName;

double totalPrice;

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

this.orderId = id;

this.customerName = name;

this.totalPrice = price;

}

}

public class OrderSorting {

static void bubbleSort(Order[] orders) {

for (int i = 0; i < orders.length - 1; i++) {

for (int j = 0; j < orders.length - 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;

}

public static void main(String[] args) {

Order[] orders = {

new Order(1, "Alice", 2000),

new Order(2, "Bob", 5000),

new Order(3, "Charlie", 1000)

};

*bubbleSort*(orders);

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

for (Order o : orders) {

System.***out***.println(o.customerName + " " + o.totalPrice);

}

Order[] orders2 = {

new Order(1, "Alice", 2000),

new Order(2, "Bob", 5000),

new Order(3, "Charlie", 1000)

};

*quickSort*(orders2, 0, orders2.length - 1);

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

for (Order o : orders2) {

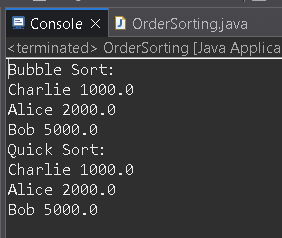
System.***out***.println(o.customerName + " " + o.totalPrice);

}

}

}

Output:



Exercise 4: Employee Management System

package dsa4;

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;

}

}

public class EmployeeSystem {

static Employee[] *employees* = new Employee[100];

static int *count* = 0;

static void add(Employee e) {

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

}

static void search(int id) {

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

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

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

return;

}

}

}

static void traverse() {

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

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

}

}

static void delete(int id) {

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

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

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

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

}

*count*--;

break;

}

}

}

public static void main(String[] args) {

*add*(new Employee(1, "John", "Manager", 50000));

*add*(new Employee(2, "Jane", "Engineer", 40000));

*search*(1);

*traverse*();

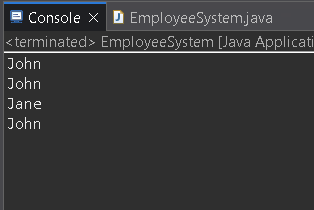
*delete*(2);

*traverse*();

}

}

Output:



Exercise 5: Task Management System

package dsa5;

class Task {

int taskId;

String taskName;

String status;

Task next;

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

this.taskId = id;

this.taskName = name;

this.status = status;

this.next = null;

}

}

public class TaskList {

Task head = null;

void add(Task t) {

if (head == null) head = t;

else {

Task temp = head;

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

temp.next = t;

}

}

void search(int id) {

Task temp = head;

while (temp != null) {

if (temp.taskId == id) {

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

return;

}

temp = temp.next;

}

}

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;

}

public static void main(String[] args) {

TaskList list = new TaskList();

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

list.add(new Task(2, "Develop", "In Progress"));

list.search(1);

list.traverse();

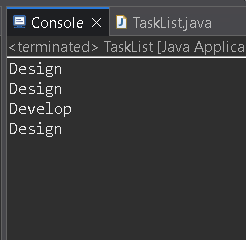
list.delete(2);

list.traverse();

}

}

Output:



Exercise 6: Library Management System

package dsa6;

class Book {

int bookId;

String title;

String author;

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

this.bookId = id;

this.title = title;

this.author = author;

}

}

public class LibrarySearch {

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;

}

public static void main(String[] args) {

Book[] books = {

new Book(1, "Java", "James"),

new Book(2, "Python", "Guido"),

new Book(3, "C++", "Bjarne")

};

Book[] sortedBooks = {books[2], books[0], books[1]};

java.util.Arrays.*sort*(sortedBooks, (a, b) -> a.title.compareTo(b.title));

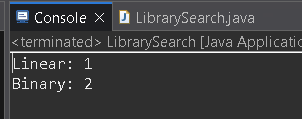
System.***out***.println("Linear: " + *linearSearch*(books, "Python"));

System.***out***.println("Binary: " + *binarySearch*(sortedBooks, "Python"));

}

}

Output:



Exercise 7: Financial Forecasting

package dsa7;

public class FinancialForecast {

static double forecast(double value, double rate, int years) {

if (years == 0) return value;

return *forecast*(value \* (1 + rate), rate, years - 1);

}

public static void main(String[] args) {

double result = *forecast*(1000, 0.10, 3);

System.***out***.println("Forecast: " + result);

}

}

Output:

