**Exercise 1: Inventory Management System**

**Scenario:**

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

**Program:**

import java.util.\*;

class Product {

String productId;

String productName;

int quantity;

double price;

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

this.productId = id;

this.productName = name;

this.quantity = qty;

this.price = price;

}

public String toString() {

return productId + " - " + productName + " - Qty: " + quantity + " - Price: ₹" + price;

}

}

public class Main {

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

static void addProduct(Product p) {

inventory.put(p.productId, p);

}

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

if (inventory.containsKey(id)) {

Product p = inventory.get(id);

p.quantity = qty;

p.price = price;

}

}

static void deleteProduct(String id) {

inventory.remove(id);

}

public static void main(String[] args) {

addProduct(new Product("P101", "Mouse", 50, 299.99));

addProduct(new Product("P102", "Keyboard", 30, 799.49));

updateProduct("P101", 60, 289.99);

deleteProduct("P102");

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

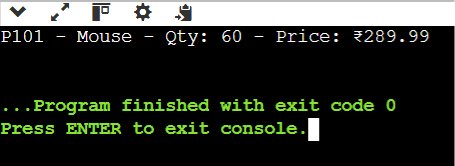
System.out.println(p);

}

}

}

**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.

**Program:**

import java.util.Arrays;

import java.util.Comparator;

class Main {

static class Product {

int productId;

String name;

String category;

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

this.productId = id;

this.name = name;

this.category = cat;

}

}

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

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

if (products[i].name.equals(name)) return i;

}

return -1;

}

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

if (low > high) return -1;

int mid = (low + high) / 2;

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

if (cmp == 0) return mid;

else if (cmp > 0) return binarySearch(products, name, low, mid - 1);

else return binarySearch(products, name, mid + 1, high);

}

public static void main(String[] args) {

Product[] products = {

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

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

new Product(3, "Laptop", "Electronics")

};

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

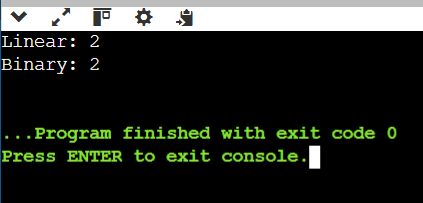
Arrays.sort(products, Comparator.comparing(p -> p.name));

System.out.println("Binary: " + binarySearch(products, "Laptop", 0, products.length - 1));

}

}

**Output:**

****