Advanced Product Management System using Arrays in Core Java (with Sorting, File Handling, and Category Filters)

Problem Statement

Enhance the Product Management System so that:
- Products are stored in a fixed-size array (max 100).
- Each Product has: id, name, price, quantity, category.
- Support operations: Add, View, Search by ID, Update, Delete.
- Additional features: Sort by price, Sort by name, Filter by category.
- Data should be persisted across runs using a text file (save & load).

Requirements

1) Language: Core Java (no Collections for core storage; use array of Product).
2) Class Product: int id, String name, double price, int quantity, String category.
3) Menu-driven console UI.
4) File persistence: products.txt (CSV format per line).
5) Sorting limited to the active range [0..count).
6) Filtering should be case-insensitive on category.

UML (ASCII Diagram)

```
+------------------------------+
|            Product           |
+------------------------------+
| - id: int                    |
| - name: String               |
| - price: double              |
| - quantity: int              |
| - category: String           |
+------------------------------+
| + display(): void            |
| + toFileString(): String     |
| + fromFileString(String):    |
|          static Product      |
+------------------------------+

+---------------------------------------------+
|      AdvancedProductManagementSystem        |
+---------------------------------------------+
| - products: Product[100]                    |
| - count: int                                |
| - FILE_NAME: String                         |
+---------------------------------------------+
| + addProduct(): void                        |
| + viewProducts(): void                      |
| + searchProduct(): void                     |
| + updateProduct(): void                     |
| + deleteProduct(): void                     |
| + sortByPrice(): void                       |
| + sortByName(): void                        |
| + filterByCategory(): void                  |
| + saveToFile(): void                        |
| + loadFromFile(): void                      |
+---------------------------------------------+
```

Java Source Code

```
import java.io.*;
import java.util.*;
class Product {
```

```
    int id;
    String name;
    double price;
    int quantity;
    String category;

    public Product(int id, String name, double price, int quantity, String category) {
        this.id = id;
        this.name = name;
        this.price = price;
        this.quantity = quantity;
        this.category = category;
    }

    public void display() {
        System.out.printf("%-10d %-15s %-10.2f %-10d %-15s\n", id, name, price, quantity, category);
    }

    public String toFileString() {
        return id + "," + name + "," + price + "," + quantity + "," + category;
    }

    public static Product fromFileString(String line) {
        String[] parts = line.split(",");
        return new Product(
                Integer.parseInt(parts[0]),
                parts[1],
                Double.parseDouble(parts[2]),
                Integer.parseInt(parts[3]),
                parts[4]
        );
    }
}

public class AdvancedProductManagementSystem {
    static Product[] products = new Product[100];
    static int count = 0;
    static Scanner sc = new Scanner(System.in);
    static final String FILE_NAME = "products.txt";

    public static void main(String[] args) {
        loadFromFile();

        int choice;
        do {
            System.out.println("\n=== Advanced Product Management System ===");
            System.out.println("1. Add Product");
            System.out.println("2. View All Products");
            System.out.println("3. Search Product by ID");
            System.out.println("4. Update Product");
            System.out.println("5. Delete Product");
            System.out.println("6. Sort Products by Price");
            System.out.println("7. Sort Products by Name");
            System.out.println("8. Filter Products by Category");
            System.out.println("9. Save & Exit");
            System.out.print("Enter choice: ");
            choice = sc.nextInt();

            switch (choice) {
                case 1 -> addProduct();
                case 2 -> viewProducts();
                case 3 -> searchProduct();
                case 4 -> updateProduct();
                case 5 -> deleteProduct();
                case 6 -> sortByPrice();
                case 7 -> sortByName();
                case 8 -> filterByCategory();
                case 9 -> {
                    saveToFile();
                    System.out.println("Data saved. Exiting...");
                }
                default -> System.out.println("Invalid choice! Try again.");
            }
        } while (choice != 9);
    }
```

```
// 1. Add Product
public static void addProduct() {
    if (count >= products.length) {
        System.out.println("Product list is full!");
        return;
    }
    System.out.print("Enter Product ID: ");
    int id = sc.nextInt();
    sc.nextLine();
    System.out.print("Enter Product Name: ");
    String name = sc.nextLine();
    System.out.print("Enter Price: ");
    double price = sc.nextDouble();
    System.out.print("Enter Quantity: ");
    int qty = sc.nextInt();
    sc.nextLine();
    System.out.print("Enter Category (Electronics/Clothing/Grocery): ");
    String category = sc.nextLine();

    products[count++] = new Product(id, name, price, qty, category);
    System.out.println("Product added successfully!");
}

// 2. View Products
public static void viewProducts() {
```

```
if (count == 0) {
            System.out.println("No products available!");
            return;
        }
        System.out.printf("%-10s %-15s %-10s %-10s %-15s\n", "ID", "Name", "Price", "Quantity", "Category");
        System.out.println("-------------------------------------------------------------------------");
        for (int i = 0; i < count; i++) {
            products[i].display();
        }
    }

    // 3. Search Product
    public static void searchProduct() {
        System.out.print("Enter Product ID to search: ");
        int id = sc.nextInt();
        for (int i = 0; i < count; i++) {
            if (products[i].id == id) {
                System.out.println("Product found:");
                products[i].display();
                return;
            }
        }
        System.out.println("Product not found!");
    }

    // 4. Update Product
    public static void updateProduct() {
        System.out.print("Enter Product ID to update: ");
        int id = sc.nextInt();
        for (int i = 0; i < count; i++) {
            if (products[i].id == id) {
                sc.nextLine();
                System.out.print("Enter New Name: ");
                products[i].name = sc.nextLine();
                System.out.print("Enter New Price: ");
                products[i].price = sc.nextDouble();
                System.out.print("Enter New Quantity: ");
                products[i].quantity = sc.nextInt();
                sc.nextLine();
                System.out.print("Enter New Category: ");
                products[i].category = sc.nextLine();
                System.out.println("Product updated successfully!");
                return;
            }
        }
        System.out.println("Product not found!");
    }

    // 5. Delete Product
    public static void deleteProduct() {
        System.out.print("Enter Product ID to delete: ");
        int id = sc.nextInt();
        for (int i = 0; i < count; i++) {
            if (products[i].id == id) {
                for (int j = i; j < count - 1; j++) {
                    products[j] = products[j + 1];
                }
                products[--count] = null;
                System.out.println("Product deleted successfully!");
                return;
            }
        }
        System.out.println("Product not found!");
    }

    // 6. Sort by Price
    public static void sortByPrice() {
        Arrays.sort(products, 0, count, Comparator.comparingDouble(p -> p.price));
        System.out.println("Products sorted by price!");
    }

    // 7. Sort by Name
    public static void sortByName() {
        Arrays.sort(products, 0, count, Comparator.comparing(p -> p.name.toLowerCase()));
        System.out.println("Products sorted by name!");
```

```
}

// 8. Filter by Category
public static void filterByCategory() {
    sc.nextLine();
    System.out.print("Enter Category to filter: ");
    String cat = sc.nextLine();
    boolean found = false;
    System.out.printf("%-10s %-15s %-10s %-10s %-15s\n", "ID", "Name", "Price", "Quantity", "Category");
    System.out.println("-------------------------------------------------------------------------");
    for (int i = 0; i < count; i++) {
        if (products[i].category.equalsIgnoreCase(cat)) {
            products[i].display();
            found = true;
        }
    }
    if (!found) {
        System.out.println("No products found in this category!");
    }
}

// File Ha
```

```
ndling
    public static void saveToFile() {
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(FILE_NAME))) {
            for (int i = 0; i < count; i++) {
                bw.write(products[i].toFileString());
                bw.newLine();
            }
        } catch (IOException e) {
            System.out.println("Error saving file: " + e.getMessage());
        }
    }

    public static void loadFromFile() {
        File file = new File(FILE_NAME);
        if (!file.exists()) return;
        try (BufferedReader br = new BufferedReader(new FileReader(FILE_NAME))) {
            String line;
            while ((line = br.readLine()) != null) {
                products[count++] = Product.fromFileString(line);
            }
        } catch (IOException e) {
            System.out.println("Error loading file: " + e.getMessage());
        }
    }
}
```

Test Cases

#	Action / Input	Expected Output
1	Add Product: (101, Laptop, 55000, 10, Electronics)	Product added successfully!
2	View Products	Shows table with 101 Laptop 55000.00 10 Electronics
3	Search Product (101)	Product found
4	Update Product (101 -> Gaming Laptop, 60000, 8, Electronics)	Product updated successfully!
5	Delete Product (101)	Product deleted successfully!
6	Delete same Product again (101)	Product not found!
7	Sort by Price after adding multiple products	Products sorted by price!
8	Sort by Name after adding multiple products	Products sorted by name!
9	Filter by Category (Electronics)	Only electronics shown or 'No products found'
10	Save & Exit, then restart and View Products	Previously saved data is loaded