Collections Assignment

02, June, 2025

- Jahnavi Lakshmi Muttireddy

1) We are looking for a Java-based application that will help us efficiently manage product records using the Collections framework. The system should allow us to:

- Store and manage product data in a structured format.
- Perform key operations such as adding, retrieving, updating, and deleting product records.
- Sort products dynamically based on criteria like product id, product name.
- Prevent duplicate entries to maintain data integrity.

Product entity should contain the following:

Product ID

Product Name

Category

Price

Solution:

ProductDetails

Defines the Product entity with fields for product ID, name, category, and price.

```
public class ProductDetails {  18 usages                                                    ⚠6
    private int productId;  4 usages
    private String productName;  4 usages
    private String category;  4 usages
    private double price;  4 usages

    public ProductDetails(int productId, String productName, String category, double price) {...}

    public int getProductId() { return productId; }

    public void setProductId(int productId) { this.productId = productId; }

    public String getProductName() { return productName; }

    public void setProductName(String productName) { this.productName = productName; }

    public String getCategory() { return category; }

    public void setCategory(String category) { this.category = category; }

    public double getPrice() { return price; }

    public void setPrice(double price) { this.price = price; }
```

ProductManage:

Handles product operations using Collections Framework: add, get, update, delete, and sort.

Adding Products and Handling duplicates:

```java
public void addProduct(ProductDetails product){  5 usages
    if(productIdset.contains(product.getProductId())){
        System.out.println("Duplicate Product ID: "+product.getProductId());
        return;
    }
    productList.add(product);
    productIdset.add(product.getProductId());
}
```

```java
public static void main(String[] args){
    ProductManage pm = new ProductManage();
    pm.addProduct(new ProductDetails( productId: 101,  productName: "Mobile",  category: "Electronics",  price: 10000));
    pm.addProduct(new ProductDetails( productId: 102,  productName: "Paints",  category: "House Hold",  price: 500));
    pm.addProduct(new ProductDetails( productId: 103,  productName: "Pens",  category: "Stationery",  price: 100));
    pm.addProduct(new ProductDetails( productId: 104,  productName: "Laptop",  category: "Electronics",  price: 20000));
    pm.addProduct(new ProductDetails( productId: 105,  productName: "5star",  category: "Food Items",  price: 50));
```

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Users\JahnaviLakshmiMuttir\AppData\Local\JetBrains\Int
101| Mobile| Electronics| 10000.0
102| Paints| House Hold| 500.0
103| Pens| Stationery| 100.0
104| Laptop| Electronics| 20000.0
105| 5star| Food Items| 50.0
```

```java
pm.addProduct(new ProductDetails( productId: 104,  productName: "Charger",  category: "Electronics",  price: 500));
```

```
Duplicate Product ID: 104
```

Getting a product by id:

```java
public ProductDetails getProductById(int id){  no usages
    for(ProductDetails product: productList){
        if(product.getProductId() == id)
            return product;
    }
    return null;
}
```

```java
System.out.println(pm.getProductById(102));
```

```
102| Paints| House Hold| 500.0
```

Delete a product by Id;

```java
public boolean deleteProduct(int id){  no usages
    for(int i=0;i<productList.size();i++){
        if(productList.get(i).getProductId() == id){
            productList.remove(i);
            productIdset.remove(id);
            return true;
        }
    }
    return false;
}
```

Update a product by id:

```java
public boolean updateProduct(ProductDetails product){  no usages
    for(int i=0;i<productList.size();i++){
        if(productList.get(i).getProductId() == product.getProductId()){
            productList.set(i, product);
            return true;
        }
    }
    return false;
}
```

Sort all the product by id:

```java
public List<ProductDetails> sortProductsById() {  no usages
     List<ProductDetails> sortId =  new ArrayList<>(productList);
     sortId.sort(Comparator.comparing(ProductDetails::getProductId));
     return sortId;
}
```

```
101| Mobile| Electronics| 10000.0
103| Notebook| Stationery| 100.0
104| Laptop| Electronics| 20000.0
105| 5star| Food Items| 50.0
```

Sort products by name:

```java
public List<ProductDetails> sortProductsByName(){  no usages
     List<ProductDetails> sortName = new ArrayList<>(productList);
     sortName.sort(Comparator.comparing(ProductDetails::getProductName));
     return sortName;
}
```

```
105| 5star| Food Items| 50.0
104| Laptop| Electronics| 20000.0
101| Mobile| Electronics| 10000.0
103| Notebook| Stationery| 100.0
```

2) Create a product catalogue key as a product and value as quantity:

- Store and manage product data in a structured format.
- Perform key operations such as adding, retrieving, updating, and deleting product records.
- Sort products dynamically based on criteria like product id, product name.
- Prevent duplicate entries to maintain data integrity.

Product entity should contain the following:

Product ID

Product Name

Category Price

Solution:

Add Product to Catalog:

Adds a new product if the productId doesn't already exist.

```
public class ProductCatalog {  2 usages
    Map<ProductDetails, Integer> catalog = new HashMap<>();  10 usages

    public void addProduct(ProductDetails product, int quantity){  5 usages
        if(catalog.containsKey(product)){
            System.out.println("Duplicate Product ID: "+ product.getProductId());
            return;
        }
        catalog.put(product, quantity);
    }
```

```
public class CatalogMain {
    public static void main(String[] args) {
        ProductCatalog catalog = new ProductCatalog();

        catalog.addProduct(new ProductDetails( productId: 201,  productName: "Laptop",  category: "Electroni
        catalog.addProduct(new ProductDetails( productId: 202,  productName: "Monitor",  category: "Electron
        catalog.addProduct(new ProductDetails( productId: 203,  productName: "Book",  category: "Stationery"
        catalog.addProduct(new ProductDetails( productId: 204,  productName: "Pen",  category: "Stationery",
        catalog.displayProducts(new ArrayList<>(catalog.getAllProducts().entrySet()));
```

```
Duplicate Product ID: 202
```

```
201| Laptop| Electronics| 75000.0 | Quantity: 5
202| Monitor| Electronics| 15000.0 | Quantity: 3
203| Book| Stationery| 500.0 | Quantity: 20
204| Pen| Stationery| 25.0 | Quantity: 100
```

Get Quantity by Product ID:

Retrieves the quantity of a product by its ID.

```
    public Integer getQuantity(int id){  1 usage
        for(ProductDetails pd : catalog.keySet()){
            if(pd.getProductId() == id){
                return catalog.get(pd);
            }
        }
        return null;
    }
```

```
20
```

Update Product Quantity by ID

Updates the quantity of a product identified by productId.

```
public void updateQuantity(int id, int quantity){  1 usage
    for(ProductDetails pd : catalog.keySet()){
        if(pd.getProductId() == id){
            catalog.put(pd, quantity);
            return;
        }
    }
}
```

```
    catalog.updateQuantity( id: 204,  quantity: 120);
```

Remove Product by ID

Removes the product from the catalog using productid.

```
public void deleteProduct(int id){  1 usage
    for(ProductDetails pd : catalog.keySet()){
        if(pd.getProductId() == id){
            catalog.remove(pd);
            return;
        }
    }
}
```

Get Entire Product Map

Returns a copy of the entire product catalog.

```
public Map<ProductDetails, Integer> getAllProducts(){  1 usage
    return new HashMap<>(catalog);
}
```

```
201| Laptop| Electronics| 75000.0 | Quantity: 5
202| Monitor| Electronics| 15000.0 | Quantity: 3
203| Book| Stationery| 500.0 | Quantity: 20
204| Pen| Stationery| 25.0 | Quantity: 100
```

Sort Products by ID

Returns product entries sorted in ascending order by productId.

```
public List<Map.Entry<ProductDetails, Integer>> sortByProductId(){  1 usage
    List<Map.Entry<ProductDetails, Integer>> sortedList = new ArrayList<>(catalog.entrySet());
    sortedList.sort(Comparator.comparing( Entry<ProductDetails, Integer> e -> e.getKey().getProductId()))
    return sortedList;
}
```

```
 catalog.displayProducts(catalog.sortByProductId());
```

```
202| Monitor| Electronics| 15000.0 | Quantity: 3
203| Book| Stationery| 500.0 | Quantity: 20
204| Pen| Stationery| 25.0 | Quantity: 120
```

Sort Products by Name

Returns product entries sorted alphabetically by productName.

```
public List<Map.Entry<ProductDetails, Integer>> sortByProductName() {  1 usage
    List<Map.Entry<ProductDetails, Integer>> sortedList = new ArrayList<>(catalog.entrySet());
    sortedList.sort(Comparator.comparing( Entry<ProductDetails, Integer> e -> e.getKey().getProductName()
    return sortedList;
}
```

```
 catalog.displayProducts(catalog.sortByProductName());
```

```
203| Book| Stationery| 500.0 | Quantity: 20
202| Monitor| Electronics| 15000.0 | Quantity: 3
204| Pen| Stationery| 25.0 | Quantity: 120
```