JAVA COLLECTION ASSIGNMENT

2nd June 2025

Abhik Chakraborty (Abhik.Chakraborty@bounteous.com)

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

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:

1. Stores product info (ID, name, category, price)
2. Add, update, delete, and view products
3. Prevents duplicate products
4. Can **sort products by ID or name**
5. Maintains a **product catalogue** where each product has a quantity

1. Product.java -> A **POJO** is a simple class.

```java
package june2nd.assignment;

import java.util.Objects;

public class Product {  2 usages
    int productId;  7 usages
    String productName;  7 usages
    String category;  7 usages
    double price;  7 usages

    Product(){  no usages
    }
    Product(int productId, String productName, String category, double price){
        this.productId = productId;
        this.productName = productName;
        this.category = category;
        this.price = price;
    }
```

```java
    public int getProductId() {  no usages
        return productId;
    }
    public void setProductId(int productId) {  no usages
        this.productId = productId;
    }

    public String getProductName() {  no usages
        return productName;
    }
    public void setProductName(String productName) {  no usages
        this.productName = productName;
    }

    public String getCategory() {  no usages
        return category;
    }
```

```java
    public void setCategory(String category) {  no usages
        this.category = category;
    }

    public double getPrice() {  no usages
        return price;
    }
    public void setPrice(double price) {  no usages
        this.price = price;
    }

    @Override
    public String toString() {
        return productId + " | " + productName + " | " + category + " | $" + price;
    }

```

```java
    @Override
    public boolean equals(Object o) {
        if (o == null || getClass() != o.getClass()) return false;
        Product product = (Product) o;
        return productId == product.productId && Double.compare(price, product.price) == 0 && Objects.equals(productName, product.productName)
                && Objects.equals(category, product.category);
    }

    @Override
    public int hashCode() {
        return Objects.hash(productId, productName, category, price);
    }
}
```

2. Manage Products (ProductManagerService)

```java
package june2nd.assignment;
import java.util.*;
public class ProductManagerService {  2 usages
    private Set<Product> productSet = new HashSet<>();  7 usages

    public boolean addProduct(Product product) {  2 usages
        return productSet.add(product);
    }

    public Product getProductById(int id) {  no usages
        for (Product p : productSet) {
            if (p.getProductId() == id) return p;
        }
        return null;
    }

```

```java
    public boolean updateProduct(Product updatedProduct) {  1 usage
        for (Product p : productSet) {
            if (p.getProductId() == updatedProduct.getProductId()) {
                p.setProductName(updatedProduct.getProductName());
                p.setCategory(updatedProduct.getCategory());
                p.setPrice(updatedProduct.getPrice());
                return true;
            }
        }
        return false;
    }

    public boolean deleteProduct(int id) {  no usages
        return productSet.removeIf( Product p -> p.getProductId() == id);
    }
```

```java
    public List<Product> sortById() {  no usages
        List<Product> list = new ArrayList<>(productSet);
        list.sort(Comparator.comparingInt(Product::getProductId));
        return list;
    }

    public List<Product> sortByName() {  1 usage
        List<Product> list = new ArrayList<>(productSet);
        list.sort(Comparator.comparing(Product::getProductName));
        return list;
    }

    public Set<Product> getAllProducts() {  no usages
        return productSet;
    }
}
```

3. Create Product Catalogue (ProductCatalogueService)

```java
package june2nd.assignment;
import java.util.*;
public class ProductCatalogueService {  no usages
    private Map<Product, Integer> catalogue = new HashMap<>();  9 usages

    public boolean addProduct(Product product, int quantity) {  no usages
        if (!catalogue.containsKey(product)) {
            catalogue.put(product, quantity);
            return true;
        }
        return false; // Prevent duplicates
    }

    public boolean updateQuantity(Product product, int quantity) {  no usages
        if (catalogue.containsKey(product)) {
            catalogue.put(product, quantity);
            return true;
        }
        return false;
    }
```

```java
    public boolean deleteProduct(Product product) {  no usages
        return catalogue.remove(product) != null;
    }

    public int getQuantity(Product product) {  no usages
        return catalogue.getOrDefault(product,  defaultValue: 0);
    }

    public List<Map.Entry<Product, Integer>> sortById() {  no usages
        List<Map.Entry<Product, Integer>> list = new ArrayList<>(catalogue.entrySet());
        list.sort(Comparator.comparing( Entry<Product, Integer> entry -> entry.getKey().getProductId()));
        return list;
    }

    public List<Map.Entry<Product, Integer>> sortByName() {  no usages
        List<Map.Entry<Product, Integer>> list = new ArrayList<>(catalogue.entrySet());
        list.sort(Comparator.comparing( Entry<Product, Integer> entry -> entry.getKey().getProductName()));
        return list;
    }

    public Map<Product, Integer> getAllProducts() {  no usages
        return catalogue;
    }
}
```

The Main Function:

```java
package june2nd.assignment;

import java.util.Map;

public class Main {
    public static void main(String[] args) {
        Product p1 = new Product( productId: 1,  productName: "Laptop",  category: "Electronics",  price: 70000);
        Product p2 = new Product( productId: 2,  productName: "Phone",  category: "Electronics",  price: 30000);
        Product p3 = new Product( productId: 3,  productName: "Mouse",  category: "Accessories",  price: 500);

        ProductManagerService manager = new ProductManagerService();
        manager.addProduct(p1);
        manager.addProduct(p2);
        manager.updateProduct(new Product( productId: 2,  productName: "Smartphone",  category: "Electronics",  price: 35000));

        System.out.println("Sorted Products by Name:");
        for (Product p : manager.sortByName()) {
            System.out.println(p);
        }
```

```java
        ProductCatalogueService catalogue = new ProductCatalogueService();
        catalogue.addProduct(p1, quantity: 10);
        catalogue.addProduct(p3, quantity: 50);
        catalogue.updateQuantity(p1, quantity: 15);

        System.out.println("\nCatalogue Sorted by Product ID:");
        for (Map.Entry<Product, Integer> entry : catalogue.sortById()) {
            System.out.println(entry.getKey() + " | Quantity: " + entry.getValue());
        }
    }
}
```

THE OUTPUT:

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent
Sorted Products by Name:
1 | Laptop | Electronics | $70000.0
2 | Smartphone | Electronics | $35000.0

Catalogue Sorted by Product ID:
1 | Laptop | Electronics | $70000.0 | Quantity: 15
3 | Mouse | Accessories | $500.0 | Quantity: 50
```

POJO	To store product info cleanly
HashSet	To avoid duplicate products
HashMap	To link products with their quantities
equals/hashCode	To treat products with same ID as same
Comparator	To sort products by ID or name