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

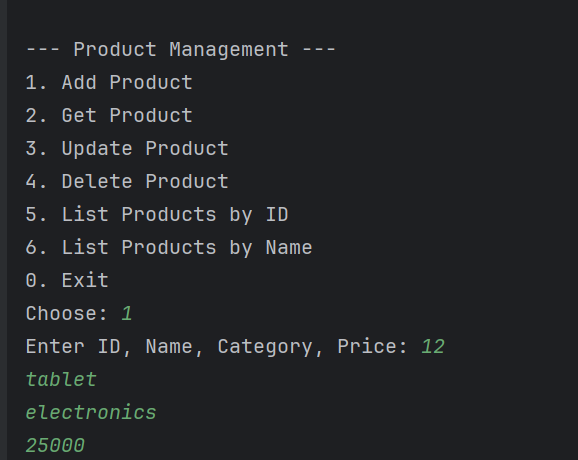
Product Name

Category

Price

import java.util.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 ProductManager manager = new ProductManager();  
  
 while (true) {  
 System.*out*.println("\n--- Product Management ---");  
 System.*out*.println("1. Add Product");  
 System.*out*.println("2. Get Product");  
 System.*out*.println("3. Update Product");  
 System.*out*.println("4. Delete Product");  
 System.*out*.println("5. List Products by ID");  
 System.*out*.println("6. List Products by Name");  
 System.*out*.println("0. Exit");  
 System.*out*.print("Choose: ");  
  
 int choice = sc.nextInt();  
 switch (choice) {  
 case 1:  
 System.*out*.print("Enter ID, Name, Category, Price: ");  
 int id = sc.nextInt();  
 String name = sc.next();  
 String cat = sc.next();  
 double price = sc.nextDouble();  
 boolean added = manager.addProduct(new Product(id, name, cat, price));  
 if (!added) System.*out*.println("Product ID already exists.");  
 break;  
 case 2:  
 System.*out*.print("Enter Product ID: ");  
 id = sc.nextInt();  
 Product p = manager.getProduct(id);  
 if (p != null) System.*out*.println(p);  
 else System.*out*.println("Product not found.");  
 break;  
 case 3:  
 System.*out*.print("Enter ID to update: ");  
 id = sc.nextInt();  
 System.*out*.print("New Name, Category, Price: ");  
 name = sc.next();  
 cat = sc.next();  
 price = sc.nextDouble();  
 boolean updated = manager.updateProduct(id, name, cat, price);  
 if (!updated) System.*out*.println("Product not found.");  
 break;  
 case 4:  
 System.*out*.print("Enter ID to delete: ");  
 id = sc.nextInt();  
 boolean deleted = manager.deleteProduct(id);  
 if (!deleted) System.*out*.println("Product not found.");  
 break;  
 case 5:  
 manager.printProducts(manager.getProductsSortedById());  
 break;  
 case 6:  
 manager.printProducts(manager.getProductsSortedByName());  
 break;  
 case 0:  
 sc.close();  
 return;  
 default:  
 System.*out*.println("Invalid choice.");  
 }  
 }  
 }  
}  
  
import java.util.\*;  
  
public class ProductManager {  
 private Map<Integer, Product> productMap = new HashMap<>();  
  
 public boolean addProduct(Product product) {  
 if (productMap.containsKey(product.getProductId())) {  
 return false;  
 }  
 productMap.put(product.getProductId(), product);  
 return true;  
 }  
  
 public Product getProduct(int productId) {  
 return productMap.get(productId);  
 }  
  
 public boolean updateProduct(int productId, String name, String category, double price) {  
 Product product = productMap.get(productId);  
 if (product == null) return false;  
  
 product.setProductName(name);  
 product.setCategory(category);  
 product.setPrice(price);  
 return true;  
 }  
  
 public boolean deleteProduct(int productId) {  
 return productMap.remove(productId) != null;  
 }  
  
 public List<Product> getProductsSortedById() {  
 List<Product> products = new ArrayList<>(productMap.values());  
 products.sort(Comparator.*comparingInt*(Product::getProductId));  
 return products;  
 }  
  
 public List<Product> getProductsSortedByName() {  
 List<Product> products = new ArrayList<>(productMap.values());  
 products.sort(Comparator.*comparing*(Product::getProductName));  
 return products;  
 }  
  
 public void printProducts(List<Product> products) {  
 for (Product product : products) {  
 System.*out*.println(product);  
 }  
 }  
}

public class Product {  
 private int productId;  
 private String productName;  
 private String category;  
 private double price;  
  
 public Product(int productId, String productName, String category, double price) {  
 this.productId = productId;  
 this.productName = productName;  
 this.category = category;  
 this.price = price;  
 }  
  
 public int getProductId() { return productId; }  
 public String getProductName() { return productName; }  
 public String getCategory() { return category; }  
 public double getPrice() { return price; }  
  
 public void setProductName(String productName) { this.productName = productName; }  
 public void setCategory(String category) { this.category = category; }  
 public void setPrice(double price) { this.price = price; }  
  
 @Override  
 public String toString() {  
 return "Product [ID=" + productId + ", Name=" + productName +  
 ", Category=" + category + ", Price=" + 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

import java.util.\*;  
// Parent class  
import java.util.Objects;  
  
 public class Main {  
 public static void main(String[] args) {  
 ProductCatalogue catalogue = new ProductCatalogue();  
 Scanner sc = new Scanner(System.*in*);  
  
 while (true) {  
 System.*out*.println("\n--- Product Catalogue ---");  
 System.*out*.println("1. Add Product");  
 System.*out*.println("2. Update Product");  
 System.*out*.println("3. Delete Product");  
 System.*out*.println("4. List by ID");  
 System.*out*.println("5. List by Name");  
 System.*out*.println("0. Exit");  
 System.*out*.print("Choose: ");  
  
 int choice = sc.nextInt();  
 switch (choice) {  
 case 1:  
 System.*out*.print("Enter ID, Name, Category, Price, Quantity: ");  
 int id = sc.nextInt();  
 String name = sc.next();  
 String cat = sc.next();  
 double price = sc.nextDouble();  
 int qty = sc.nextInt();  
 catalogue.addProduct(new Product(id, name, cat, price), qty);  
 break;  
 case 2:  
 System.*out*.print("Enter ID to update: ");  
 id = sc.nextInt();  
 System.*out*.print("New Name, Category, Price, Quantity: ");  
 name = sc.next();  
 cat = sc.next();  
 price = sc.nextDouble();  
 qty = sc.nextInt();  
 catalogue.updateProduct(id, name, cat, price, qty);  
 break;  
 case 3:  
 System.*out*.print("Enter ID to delete: ");  
 id = sc.nextInt();  
 catalogue.deleteProduct(id);  
 break;  
 case 4:  
 catalogue.printCatalogue(catalogue.getProductsSortedById());  
 break;  
 case 5:  
 catalogue.printCatalogue(catalogue.getProductsSortedByName());  
 break;  
 case 0:  
 sc.close();  
 return;  
 default:  
 System.*out*.println("Invalid choice.");  
 }  
 }  
 }  
 }

import java.util.Objects;  
  
public class Product {  
 private int productId;  
 private String productName;  
 private String category;  
 private double price;  
  
 public Product(int productId, String productName, String category, double price) {  
 this.productId = productId;  
 this.productName = productName;  
 this.category = category;  
 this.price = price;  
 }  
  
 public int getProductId() { return productId; }  
 public String getProductName() { return productName; }  
 public String getCategory() { return category; }  
 public double getPrice() { return price; }  
  
 public void setProductName(String productName) { this.productName = productName; }  
 public void setCategory(String category) { this.category = category; }  
 public void setPrice(double price) { this.price = price; }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Product)) return false;  
 Product product = (Product) o;  
 return productId == product.productId;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(productId);  
 }  
  
 @Override  
 public String toString() {  
 return "Product [ID=" + productId + ", Name=" + productName + ", Category=" + category + ", Price=" + price + "]";  
 }  
}

import java.util.\*;  
  
public class ProductCatalogue {  
 private Map<Product, Integer> catalogue = new HashMap<>();  
  
 public boolean addProduct(Product product, int quantity) {  
 if (catalogue.containsKey(product)) {  
 System.*out*.println("Product already exists.");  
 return false;  
 }  
 catalogue.put(product, quantity);  
 return true;  
 }  
  
 public Integer getQuantityById(int productId) {  
 for (Product product : catalogue.keySet()) {  
 if (product.getProductId() == productId) {  
 return catalogue.get(product);  
 }  
 }  
 return null;  
 }  
  
 public boolean updateProduct(int productId, String name, String category, double price, int quantity) {  
 Product target = null;  
 for (Product p : catalogue.keySet()) {  
 if (p.getProductId() == productId) {  
 target = p;  
 break;  
 }  
 }  
  
 if (target == null) return false;  
  
 target.setProductName(name);  
 target.setCategory(category);  
 target.setPrice(price);  
 catalogue.put(target, quantity);  
 return true;  
 }  
  
 public boolean deleteProduct(int productId) {  
 Product toRemove = null;  
 for (Product p : catalogue.keySet()) {  
 if (p.getProductId() == productId) {  
 toRemove = p;  
 break;  
 }  
 }  
  
 if (toRemove != null) {  
 catalogue.remove(toRemove);  
 return true;  
 }  
 return false;  
 }  
  
 public List<Map.Entry<Product, Integer>> getProductsSortedById() {  
 List<Map.Entry<Product, Integer>> list = new ArrayList<>(catalogue.entrySet());  
 list.sort(Comparator.*comparing*(entry -> entry.getKey().getProductId()));  
 return list;  
 }  
  
 public List<Map.Entry<Product, Integer>> getProductsSortedByName() {  
 List<Map.Entry<Product, Integer>> list = new ArrayList<>(catalogue.entrySet());  
 list.sort(Comparator.*comparing*(entry -> entry.getKey().getProductName()));  
 return list;  
 }  
  
 public void printCatalogue(List<Map.Entry<Product, Integer>> list) {  
 for (Map.Entry<Product, Integer> entry : list) {  
 System.*out*.println(entry.getKey() + " | Quantity: " + entry.getValue());  
 }  
 }  
}

