**Collections Assignment**

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

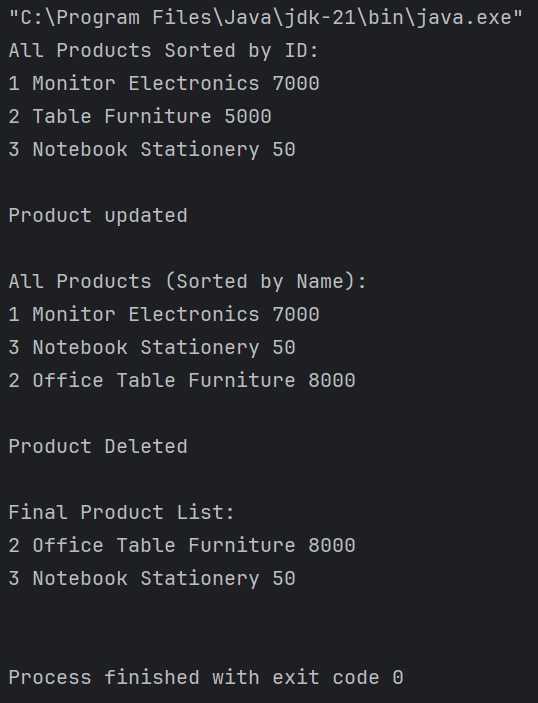
Product Name

Category

Price

package Handling;  
  
import java.util.\*;  
  
class Product{  
 private  
 int prodID;  
 String prodName;  
 String catg;  
 int price;  
 static HashMap<Integer, Product> *mp* = new HashMap<>();  
  
 public Product(int prodID, String prodName, String catg, int price) {  
 this.prodID = prodID;  
 this.prodName = prodName;  
 this.catg = catg;  
 this.price = price;  
 }  
  
 public int getProdID() {  
 return prodID;  
 }  
  
 public String getProdName() {  
 return prodName;  
 }  
  
 public String getCatg() {  
 return catg;  
 }  
  
 public int getPrice() {  
 return price;  
 }  
  
 public void setProdName(String prodName) {  
 this.prodName = prodName;  
 }  
  
 public void setCatg(String catg) {  
 this.catg = catg;  
 }  
  
 public void setPrice(int price) {  
 this.price = price;  
 }  
  
 public static boolean addProd(Product P){  
 if(*mp*.containsKey(P.getProdID())) return false;  
 *mp*.put(P.getProdID(), P);  
 return true;  
 }  
  
 public static Product getProd(int id){  
 return *mp*.get(id);  
 }  
  
 public static boolean updateProd(int id, String name, String catg, int price){  
 Product P = *mp*.get(id);  
 if(P==null) return false;  
 P.setProdName(name);  
 P.setCatg(catg);  
 P.setPrice(price);  
 return true;  
 }  
  
 public static boolean deleteProd(int id){  
 return *mp*.remove(id) != null;  
 }  
  
 public static List<Product> getSortedById() {  
 List<Product> list = new ArrayList<>(*mp*.values());  
 list.sort(Comparator.*comparingInt*(Product::getProdID));  
 return list;  
 }  
  
 public static List<Product> getSortedByName() {  
 List<Product> list = new ArrayList<>(*mp*.values());  
 list.sort(Comparator.*comparing*(Product::getProdName));  
 return list;  
 }  
 public static void display(List<Product> list) {  
 for (Product p : list) {  
 System.*out*.println(p.getProdID() + " " + p.getProdName() + " " + p.getCatg() + " " + p.getPrice());  
 }  
 System.*out*.println();  
 }  
  
}  
  
public class productques {  
 public static void main(String[] args) {  
 Product.*addProd*(new Product(1, "Monitor", "Electronics", 7000));  
 Product.*addProd*(new Product(2, "Table", "Furniture", 5000));  
 Product.*addProd*(new Product(3, "Notebook", "Stationery", 50));  
  
 System.*out*.println("All Products Sorted by ID:");  
 Product.*display*(Product.*getSortedById*());  
  
  
 if(Product.*updateProd*(2, "Office Table", "Furniture", 8000)){  
 System.*out*.println("Product updated");  
 System.*out*.println();  
 }  
  
 System.*out*.println("All Products (Sorted by Name):");  
 Product.*display*(Product.*getSortedByName*());  
  
 if(Product.*deleteProd*(1)){  
 System.*out*.println("Product Deleted");  
 System.*out*.println();  
 }  
  
 System.*out*.println("Final Product List:");  
 Product.*display*(Product.*getSortedById*());  
 }  
}

Output:



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

package Handling;  
  
import java.util.\*;  
  
class Productn {  
 private int prodID;  
 private String prodName;  
 private String catg;  
 private int price;  
 static HashMap<Productn, Integer> *catalogue* = new HashMap<>();  
  
 public Productn(int prodID, String prodName, String catg, int price) {  
 this.prodID = prodID;  
 this.prodName = prodName;  
 this.catg = catg;  
 this.price = price;  
 }  
  
 public int getProdID() { return prodID; }  
 public String getProdName() { return prodName; }  
 public String getCatg() { return catg; }  
 public int getPrice() { return price; }  
  
 public void setProdName(String prodName) { this.prodName = prodName; }  
 public void setCatg(String catg) { this.catg = catg; }  
 public void setPrice(int price) { this.price = price; }  
  
 // Equality based only on prodID  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Productn)) return false;  
 Productn p = (Productn) o;  
 return this.prodID == p.prodID;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(prodID);  
 }  
  
 @Override  
 public String toString() {  
 return prodID + " " + prodName + " " + catg + " " + price;  
 }  
  
 public static boolean addProduct(Productn p, int quantity) {  
 if (*catalogue*.containsKey(p)) return false;  
 *catalogue*.put(p, quantity);  
 return true;  
 }  
  
 public static Productn getProduct(int id) {  
 for (Productn p : *catalogue*.keySet()) {  
 if (p.getProdID() == id) return p;  
 }  
 return null;  
 }  
 public static boolean updateProduct(int id, String name, String catg, int price, int quantity) {  
 Productn target = *getProduct*(id);  
 if (target == null) return false;  
 target.setProdName(name);  
 target.setCatg(catg);  
 target.setPrice(price);  
 *catalogue*.put(target, quantity);  
 return true;  
 }  
  
 public static boolean deleteProduct(int id) {  
 Productn target = *getProduct*(id);  
 if (target == null) return false;  
 *catalogue*.remove(target);  
 return true;  
 }  
  
 public static List<Map.Entry<Productn, Integer>> getSortedById() {  
 List<Map.Entry<Productn, Integer>> list = new ArrayList<>(*catalogue*.entrySet());  
 list.sort(Comparator.*comparing*(entry -> entry.getKey().getProdID()));  
 return list;  
 }  
  
 public static List<Map.Entry<Productn, Integer>> getSortedByName() {  
 List<Map.Entry<Productn, Integer>> list = new ArrayList<>(*catalogue*.entrySet());  
 list.sort(Comparator.*comparing*(entry -> entry.getKey().getProdName()));  
 return list;  
 }  
  
 public static void display(List<Map.Entry<Productn, Integer>> list) {  
 for (Map.Entry<Productn, Integer> entry : list) {  
 System.*out*.println(entry.getKey() + " Qty: " + entry.getValue());  
 }  
 System.*out*.println();  
 }  
}  
  
class productques2 {  
 public static void main(String[] args) {  
 Productn.*addProduct*(new Productn(1, "Monitor", "Electronics", 7000), 10);  
 Productn.*addProduct*(new Productn(2, "Table", "Furniture", 5000), 5);  
 Productn.*addProduct*(new Productn(3, "Notebook", "Stationery", 50), 100);  
  
 System.*out*.println("All Products Sorted by ID:");  
 Productn.*display*(Productn.*getSortedById*());  
  
 if (Productn.*updateProduct*(2, "Office Table", "Furniture", 8000, 7)) {  
 System.*out*.println("Product updated.\n");  
 }  
 System.*out*.println("All Products Sorted by Name:");  
 Productn.*display*(Productn.*getSortedByName*());  
  
 if (Productn.*deleteProduct*(1)) {  
 System.*out*.println("Product deleted.\n");  
 }  
 System.*out*.println("Final Product List:");  
 Productn.*display*(Productn.*getSortedById*());  
 }  
}

Output:

