**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

CODE:

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

package ManageProduct;  
import java.util.\*;  
public class ProductManager {  
 private Map<Integer,Product> map=new HashMap<>();  
  
 //addProduct prevent duplicates in first entry  
 boolean addProduct(Product prod){  
 if(map.containsKey(prod.getProductId())){  
 System.*out*.println("Duplicate Product ID is Not Allowed");  
 return false;  
 }  
 map.put(prod.getProductId(),prod);  
 return true;  
 }  
  
 //get product  
 Product getProduct(int productId){  
 return map.get(productId);  
 }  
  
 //update product  
 boolean updateProduct(Product prod){  
 if(!map.containsKey(prod.getProductId())){  
 System.*out*.println("Product Not Found");  
 return false;  
 }  
 map.put(prod.getProductId(),prod);  
 return true;  
 }  
  
 //delete product  
 boolean deleteProduct(int productId){  
 return map.remove(productId)!=null;  
 }  
  
 //list all product  
 List<Product> getAllProduct(){  
 return new ArrayList<>(map.values());  
 }  
  
 //sort by id  
 List<Product> sortById(){  
 List<Product> list=getAllProduct();  
 list.sort(Comparator.*comparing*(Product::getProductId));  
 return list;  
 }  
  
 //sort by product name  
 List<Product> sortByName(){  
 List<Product> list=getAllProduct();  
 list.sort(Comparator.*comparing*(Product::getProductName));  
 return list;  
 }  
  
}

package ManageProduct;  
  
public class ProductMain {  
 public static void main(String[] args) {  
 //Add Products  
 ProductManager pm=new ProductManager();  
 pm.addProduct(new Product(101,"Phone","Electronics",23000.0));  
 pm.addProduct(new Product(102,"Laptop","Electronics",67000.0));  
 pm.addProduct(new Product(103,"Watch","Electronics",9000.0));  
  
 //Display Product  
 System.*out*.println("---------------All Products-----------");  
 pm.getAllProduct().forEach(System.*out*::println);  
  
 //Sort By Name  
 System.*out*.println("------Products Sorted By Name-------");  
 pm.sortByName().forEach(System.*out*::println);  
  
 //update Product  
 pm.updateProduct(new Product(102,"Laptop","E;lectronics",89000.0));  
  
 // delete product  
 pm.deleteProduct(103);  
 }  
}

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

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

CODE:

package ManageQuantity;  
  
import java.util.Objects;  
  
public class Product {  
 private int productId;  
 private String productName;  
 private String category;  
 private double price;  
  
 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+"]";  
 }  
  
 @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);  
 }  
}

package ManageQuantity;  
  
import java.util.\*;  
  
public class ProductCatalog {  
 private Map<Product,Integer> map=new HashMap<>();  
  
 //add or update quantity  
 public boolean addProduct(Product prod,int quantity){  
 if(map.containsKey(prod)){  
 System.*out*.println("Product Already Exist");  
 return false;  
 }  
 map.put(prod,quantity);  
 return true;  
 }  
  
 //retrive product  
 public Product getProductIdBy(int productId){  
 for(Product p:map.keySet()){  
 if(p.getProductId()==productId) return p;  
 }  
 return null;  
 }  
  
 //update  
 public boolean updateQuantity(int productId,int newQuantity){  
 Product existing=getProductIdBy(productId);  
 if(existing == null){  
 map.put(existing,newQuantity);  
 return true;  
 }  
 return false;  
 }  
  
 //delete product  
 public boolean deleteProduct(int productId){  
 Product p=getProductIdBy(productId);  
 if(p!=null){  
 map.remove(p);  
 return true;  
 }  
 return false;  
 }  
  
 //all product with quantity  
 public void displayAll(){  
 for(Map.Entry<Product,Integer> entry: map.entrySet()){  
 System.*out*.println(entry.getKey()+", Quantity: "+entry.getValue());  
 }  
 }  
  
 //sort by productid  
 List<Map.Entry<Product,Integer>> sortById(){  
 List<Map.Entry<Product,Integer>> list=new ArrayList<>(map.entrySet());  
 list.sort(Comparator.*comparing*(entry->entry.getKey().getProductId()));  
 return list;  
 }  
  
 //sort by product name  
 List<Map.Entry<Product,Integer>> sortByName(){  
 List<Map.Entry<Product,Integer>> list=new ArrayList<>(map.entrySet());  
 list.sort(Comparator.*comparing*(entry->entry.getKey().getProductName()));  
 return list;  
 }  
  
  
}

package ManageQuantity;  
  
public class Main {  
 public static void main(String[] args) {  
 ProductCatalog pc=new ProductCatalog();  
 pc.addProduct(new Product(101,"Laptop","Electronic",78000),45);  
 pc.addProduct(new Product(102,"Phone","Electronic",32000),60);  
 pc.addProduct(new Product(103,"Adapter","Electronic",1500),70);  
  
 System.*out*.println("---------Catalog---------");  
 pc.displayAll();  
  
 System.*out*.println("-----Sorted By Name------");  
 pc.sortByName().forEach(entry->System.*out*.println(entry.getKey()+" Quantity: "+entry.getValue()));  
  
 pc.updateQuantity(103,34);  
 pc.deleteProduct(102);  
  
  
 }  
}

OUTPUT:

A screen shot of a computer

AI-generated content may be incorrect.