|  |
| --- |
| Exp: 11 Date: 18/10/2023  **JAVA PROGRAMMING**  **CS6308** |

Wrapper, Exception Handling and Collections

Name: VIJAI SURIA M

Reg No.: 2021503568

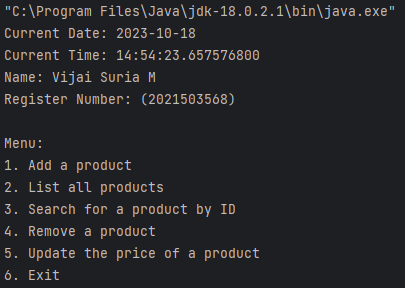
1. Write a Product class with the following attributes: id (int), name (String), price (double). Create an ArrayList to store Product objects. Ask the user to add products to the list. For each product, prompt the user for the id, name, and price and create a Product object. Add the Product object to the ArrayList. After adding products, display a menu to the user with the following options: List all products, Search for a product by ID, remove the product and update the price of the product.

**CODE:**

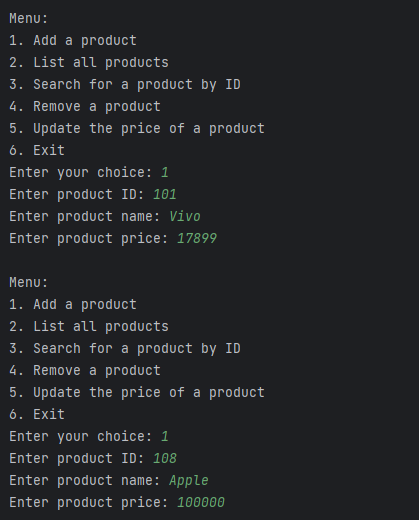
|  |
| --- |
| package LAB12\_1810;  import java.time.LocalDate; import java.time.LocalTime; import java.util.ArrayList; import java.util.List; import java.util.Scanner;  class Product {  int id;  String name;  Double price;   Product(int id, String name, Double price) {  this.id = id;  this.name = name;  this.price = price;  }   @Override  public String toString() {  return "ID: " + id + ", Name: " + name + ", Price: " + price;  } }  public class ProductList3568 {  public static void main(String args[]) {  System.*out*.println("Current Date: " + LocalDate.*now*());  System.*out*.println("Current Time: " + LocalTime.*now*());  System.*out*.println("Name: Vijai Suria M \nRegister Number: (2021503568)");   List<Product> products = new ArrayList<Product>();  Scanner scanner = new Scanner(System.*in*);   while (true) {  System.*out*.println("\nMenu:");  System.*out*.println("1. Add a product");  System.*out*.println("2. List all products");  System.*out*.println("3. Search for a product by ID");  System.*out*.println("4. Remove a product");  System.*out*.println("5. Update the price of a product");  System.*out*.println("6. Exit");  System.*out*.print("Enter your choice: ");  int choice = scanner.nextInt();   switch (choice) {  case 1:  // Add a product  System.*out*.print("Enter product ID: ");  int id = scanner.nextInt();  scanner.nextLine(); // Consume newline  System.*out*.print("Enter product name: ");  String name = scanner.nextLine();  System.*out*.print("Enter product price: ");  double price = scanner.nextDouble();  Product product = new Product(id, name, price);  products.add(product);  break;   case 2:  // List all products  for (Product p : products) {  System.*out*.println(p);  }  break;   case 3:  // Search for a product by ID  System.*out*.print("Enter product ID to search: ");  int searchId = scanner.nextInt();  for (Product p : products) {  if (p.id == searchId) {  System.*out*.println("Found: " + p);  break;  }  }  break;   case 4:  // Remove a product by ID  System.*out*.print("Enter product ID to remove: ");  int removeId = scanner.nextInt();  products.removeIf(p -> p.id == removeId);  break;   case 5:  // Update the price of a product by ID  System.*out*.print("Enter product ID to update price: ");  int updateId = scanner.nextInt();  System.*out*.print("Enter the new price: ");  double newPrice = scanner.nextDouble();  for (Product p : products) {  if (p.id == updateId) {  p.price = newPrice;  System.*out*.println("Price updated.");  break;  }  }  break;   case 6:  // Exit the program  scanner.close();  System.*exit*(0);   default:  System.*out*.println("Invalid choice. Please try again.");  break;  }  }  } } |

**OUTPUT:**

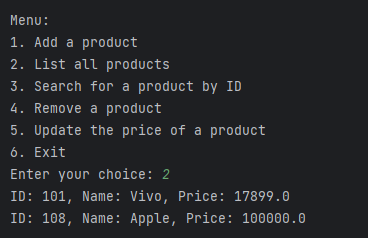
**MAIN MENU**



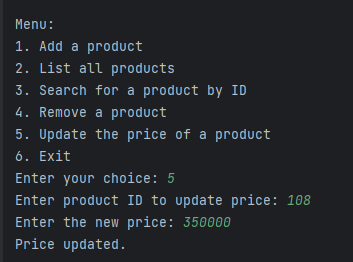
**ADD PRODUCTS:**

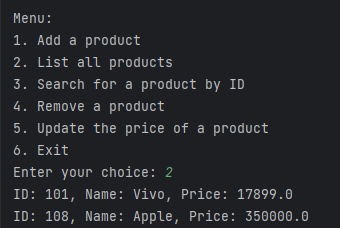


**LIST ALL PRODUCTS:**

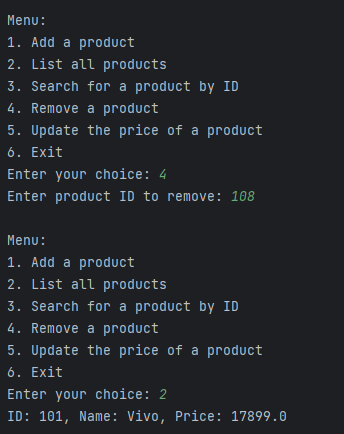


**UPDATE THE PRIZE OF A PRODUCT:**





**REMOVING THE PRODUCT:**

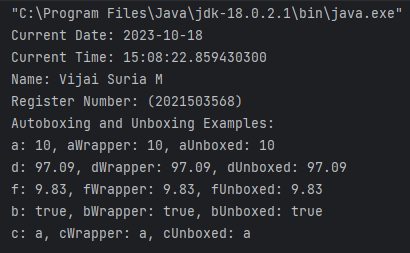


1. Write a BoxAndUnbox class. Write a method to perform Autoboxing and unboxing of all primitive types in java

**CODE:**

|  |
| --- |
| package LAB12\_1810;  import java.time.LocalDate;  import java.time.LocalTime;  public class Wrapper3568 {  public static void main(String[] args) {  System.out.println("Current Date: " + LocalDate.now());  System.out.println("Current Time: " + LocalTime.now());  System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");  int a = 10;  Integer aWrapper = a; // Autoboxing (int to Integer)  int aUnboxed = aWrapper; // Unboxing (Integer to int)  double d = 97.09D;  Double dWrapper = d; // Autoboxing (double to Double)  double dUnboxed = dWrapper; // Unboxing (Double to double)  float f = 9.83F;  Float fWrapper = f; // Autoboxing (float to Float)  float fUnboxed = fWrapper; // Unboxing (Float to float)  boolean b = true;  Boolean bWrapper = b; // Autoboxing (boolean to Boolean)  boolean bUnboxed = bWrapper; // Unboxing (Boolean to boolean)  char c = 'a';  Character cWrapper = c;  char cUnboxed = cWrapper;  System.out.println("Autoboxing and Unboxing Examples:");  System.out.println("a: " + a + ", aWrapper: " + aWrapper + ", aUnboxed: " + aUnboxed);  System.out.println("d: " + d + ", dWrapper: " + dWrapper + ", dUnboxed: " + dUnboxed);  System.out.println("f: " + f + ", fWrapper: " + fWrapper + ", fUnboxed: " + fUnboxed);  System.out.println("b: " + b + ", bWrapper: " + bWrapper + ", bUnboxed: " + bUnboxed);  System.out.println("c: " + c + ", cWrapper: " + cWrapper + ", cUnboxed: " + cUnboxed);  }  } |

**OUTPUT:**



1. **Write a Java program for an Amazonstore to handle the customer orders. The application has Product, Amazonstore, and Main classes with specific method's purpose and functionality**:

**1.Product class:**

Attributes are String name, double price, int quantity

**Product(String name, double price, int quantity):** initialize a new product with the given name, price, and quantity.

**getName():** This method returns the name of the product.

**getPrice():** This method returns the price of the product.

**getQuantity():** This method returns the quantity of the product.

**2. AmazonStore class**:

**AmazonStore():**initializes an e-commerce store with empty product storage (a map), an empty set to track product names, and an empty shopping cart (a list).

class AmazonStore {

private Map<String, Product> productMap;

private Set<String> productNames;

private List<Product> cart;

public AmazonStore() {

productMap = new HashMap<>();

productNames = new HashSet<>();

cart = new ArrayList<>();

}

**addProduct(String name, double price, int quantity) throws ProductAlreadyExistsException**: This method takes the name, price, and quantity of the product as parameters and checks if a product with the same name already exists in the store (productNames.contains(name))and throws a ProductAlreadyExistsException if so. If not, it adds the product to the map and updates the set of product names.

Product product = new Product(name, price, quantity);

productMap.put(name, product);

productNames.add(name);

**addToCart(String name, int quantity) throws ProductNotFoundException, InsufficientQuantityException**: This method allows a user to add a product to their shopping cart. It takes the name of the product and the desired quantity as parameters. It checks if the product exists in the store and throws a ProductNotFoundException if not. It also checks if there's a sufficient quantity in stock and throws an InsufficientQuantityException if not. If both checks pass, the product is added to the shopping cart.

**listProducts():** This method lists all the products available in the store.

**viewCart():** This method displays the contents of the user's shopping cart.

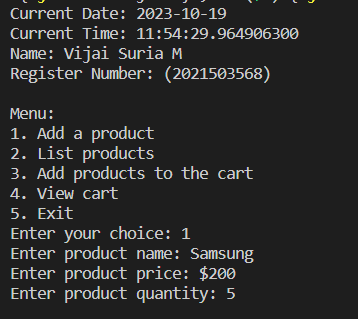
**3. Main class:**

Sets up a menu-driven interface for the user to interact with the store, and handles user input. It allows the user to **add products, list products, add products to the cart, view the cart, and exit** the program. It also handles exceptions and displays appropriate error messages if they occur.

**Code:**

|  |
| --- |
| import java.time.LocalDate;  import java.time.LocalTime;  import java.util.\*;  class ProductAlreadyExistsException extends Exception {  public ProductAlreadyExistsException(String message) {  super(message);  }  }  class ProductNotFoundException extends Exception {  public ProductNotFoundException(String message) {  super(message);  }  }  class InsufficientQuantityException extends Exception {  public InsufficientQuantityException(String message) {  super(message);  }  }  class Product {  public String name;  public double price;  public int quantity;  public Product(String name, double price, int quantity) {  this.name = name;  this.price = price;  this.quantity = quantity;  }  public String getName() {  return name;  }  public double getPrice() {  return price;  }  public int getQuantity() {  return quantity;  }  }  class AmazonStore {  private Map<String, Product> productMap;  private Set<String> productNames;  private List<Product> cart;  public AmazonStore() {  productMap = new HashMap<>();  productNames = new HashSet<>();  cart = new ArrayList<>();  }  public void addProduct(String name, double price, int quantity) throws ProductAlreadyExistsException {  if (productNames.contains(name)) {  throw new ProductAlreadyExistsException("Product with the same name already exists.");  }  Product product = new Product(name, price, quantity);  productMap.put(name, product);  productNames.add(name);  }  public void addToCart(String name, int quantity) throws ProductNotFoundException, InsufficientQuantityException {  if (!productNames.contains(name)) {  throw new ProductNotFoundException("Product not found.");  }  Product product = productMap.get(name);  if (product.getQuantity() < quantity) {  throw new InsufficientQuantityException("Insufficient quantity of " + name + " in stock.");  }  cart.add(new Product(name, product.getPrice(), quantity));  product.quantity -= quantity;  }  public void listProducts() {  System.out.println("Available Products:");  for (Product product : productMap.values()) {  System.out.println(  product.getName() + " - Price: $" + product.getPrice() + " - Quantity: " + product.getQuantity());  }  }  public void viewCart() {  System.out.println("Shopping Cart:");  double total = 0;  for (Product product : cart) {  System.out.println(  product.getName() + " - Price: $" + product.getPrice() + " - Quantity: " + product.getQuantity());  total += product.getPrice() \* product.getQuantity();  }  System.out.println("Total Price: $" + total);  }  }  public class Main {  public static void main(String[] args) {  AmazonStore store = new AmazonStore();  Scanner scanner = new Scanner(System.in);  System.out.println("Current Date: " + LocalDate.now());  System.out.println("Current Time: " + LocalTime.now());  System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");1  while (true) {  System.out.println("\nMenu:");  System.out.println("1. Add a product");  System.out.println("2. List products");  System.out.println("3. Add products to the cart");  System.out.println("4. View cart");  System.out.println("5. Exit");  System.out.print("Enter your choice: ");  int choice = scanner.nextInt();  switch (choice) {  case 1:  // Add a product  scanner.nextLine(); // Consume newline  System.out.print("Enter product name: ");  String name = scanner.nextLine();  System.out.print("Enter product price: $");  double price = scanner.nextDouble();  System.out.print("Enter product quantity: ");  int quantity = scanner.nextInt();  try {  store.addProduct(name, price, quantity);  } catch (ProductAlreadyExistsException e) {  System.out.println("Product already exists.");  break;  }  break;  case 2:  // List available products  store.listProducts();  break;  case 3:  // Add products to the cart  scanner.nextLine(); // Consume newline  System.out.print("Enter product name to add to cart: ");  String cartName = scanner.nextLine();  System.out.print("Enter quantity: ");  int cartQuantity = scanner.nextInt();  try {  store.addToCart(cartName, cartQuantity);  } catch (ProductNotFoundException e) {  System.out.println("Product not found.");  break;  } catch (InsufficientQuantityException e) {  System.out.println("Insufficient quantity of " + cartName + " in stock.");  break;  }  break;  case 4:  // View the shopping cart  store.viewCart();  break;  case 5:  // Exit the program  scanner.close();  System.exit(0);  default:  System.out.println("Invalid choice. Please try again.");  break;  }  }  }  } |

**OUTPUT:**



A screen shot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

