Java Fundamentals

Section 6: Creating an Inventory Project Project

Overview

This project will progress with you throughout Sections 4, 5, 6, and 7 of the course. After each section there will be more to add until it
builds into a complete Java application to maintain Inventory. For each part, build upon the last part so that both the old and new
requirements are met. Include all parts in a package called inventory.
Create an inventory program that can be used for a range of different products (cds, dvds, software, etc.).
Topic(s):
• Using loops (Sections 5.2 and 6.1)
• Handling exceptions (Section 6.2)
• Using if statements (Section 5.1)
• Arrays of objects (Section 6.1)
Instructions:
1. Open the inventory program that was updated in Section 5: Creating an inventory Project.
2. Ask the user to enter the number of products they wish to add. Accept a positive integer for the number of products and
handle the value of zero.
a. Create a variable named maxSize that can store integers.
b. Create a prompt at the beginning of your main method that will instruct the user to enter the required value for the
number of products they wish to store:
Enter the number of products you would like to add
Enter 0 (zero) if you do not wish to add products
c. Use a do while loop so that the program will not continue until a valid positive value is entered. If a value less than
zero is entered an error message stating “Incorrect Value entered” should be displayed before the user is re-prompted
to enter a new value. You should not leave the loop until a value of zero or greater is entered.
3. You are now going to add some error handling to deal with run-time errors in your code. Currently your program deals with
numbers entered outside the given range but cannot handle incorrect data type entries.
a. Add a try block that surrounds all of the code inside the do while loop.
b. Add a catch statement above the while that will take an Exception e parameter. The program should use a console
output statement to display the value of e to screen.
c. As you now assign a value for maxSize inside a try statement there is the possibility that maxSize will not have ben
assigned a value when you get to the while clause. To ensure this does not happen assign an initial value of -1 to
maxSize when it is declared.

HINT: Always assign a value that will fail the loop so that your code is forced to assign a correct value before it

continues.
d. Run and test your code by entering a character instead of a number.
e. Add a line of code in your catch statement that will clear out the input buffer so that the prompt will be displayed, and
the system will wait for user input.
f. Take a note of the specific type of Exception produced when you enter a character and create a catch statement just
for that exception. This error should display an Incorrect data type entered! message to the console and
should also clear the input buffer.
g. Run and test your code by entering a vriety of different input values.
4. Modify the ProductTester class to handle multiple products using a single dimensional array if a value greater than zero is
entered.
a. Create an if statement that will display the message “No products required!” to the console if the value of maxSize is
zero.
b. Add an Else statement to deal with any value other than zero. Create a single one-dimension array named products
based on the Product class that will have the number of elements specified by the user in the maxSize variable.
5. You are now going to populate the array, getting the values from the user for each field in a product object.
a. Inside the else statement under where you created the array write a for loop that will iterate through the array from zero
to 1 less than maxSize.
b. As the last input you received from the user was numeric you will need to add a statement that clears the input buffer
as the first line in your for loop.
c. Copy the code that you used to get input from the user for all a products fields into the for loop. This includes the
name, quantity, price and item number.
d. Add a new product object into the array using the index value for the position and the constructer that takes 4
parameters.
6. Use a for each loop to display the information for each individual product in the products array.
7. Remove any unnecessary code that’s not used in this exercise.
8. Save your project.

Step-by-Step Implementation Guide

1. Initial Setup

1. Create a new Java project named InventoryProject.
2. Inside the project, create a package named inventory.
3. Create a Product class inside the inventory package.
4. Create a ProductTester class inside the inventory package.

2. Product Class

Define the Product class to store product details such as name, quantity, price, and item number.

```
package inventory;

public class Product {
    private String name;
    private int quantity;
    private double price;
    private int itemNumber;

    public Product(String name, int quantity, double price, int itemNumber) {
        this.name = name;
        this.quantity = quantity;
        this.price = price;
        this.itemNumber = itemNumber;
    }

    public String getName() {
        return name;
    }

    public int getQuantity() {
        return quantity;
    }

    public double getPrice() {
        return price;
    }

    public int getItemNumber() {
        return itemNumber;
    }

    @Override
    public String toString() {
```

```
        return "Product [Name=" + name + ", Quantity=" + quantity + ", Price=" + price
+ ", Item Number=" + itemNumber + "]";
    }
}
```

3. ProductTester Class

Implement the ProductTester class to handle user input and manage the inventory.

```
package inventory;

import java.util.InputMismatchException;
import java.util.Scanner;

public class ProductTester {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int maxSize = -1;

        // Step 2: Prompt user for the number of products to add
        do {
            try {
                System.out.println("Enter the number of products you would like to add:");
                System.out.println("Enter 0 (zero) if you do not wish to add products");
                maxSize = scanner.nextInt();

                if (maxSize < 0) {
                    System.out.println("Incorrect value entered. Please enter a positive
number or zero.");
                }
            } catch (InputMismatchException e) {
                System.out.println("Incorrect data type entered! Please enter a valid
number.");
                scanner.next(); // clear the input buffer
            }
        } while (maxSize < 0);

        // Step 4: Handle multiple products using an array
        if (maxSize == 0) {
            System.out.println("No products required!");
        } else {
            Product[] products = new Product[maxSize];

            // Step 5: Populate the array with product data
            for (int i = 0; i < maxSize; i++) {
                scanner.nextLine(); // clear the input buffer

                System.out.println("Enter details for product " + (i + 1) + ":");
                System.out.print("Name: ");
                String name = scanner.nextLine();
```

```
            System.out.print("Quantity: ");
            int quantity = scanner.nextInt();
            System.out.print("Price: ");
            double price = scanner.nextDouble();
            System.out.print("Item Number: ");
            int itemNumber = scanner.nextInt();

            products[i] = new Product(name, quantity, price, itemNumber);
        }

        // Step 6: Display the information for each product
        System.out.println("Product details:");
        for (Product product : products) {
            System.out.println(product);
        }
    }

    scanner.close();
  }
}
```

4. Testing the Program

Run the ProductTester class and test the program by entering various input values, including incorrect data types, to ensure that error handling works as expected.

5. Clean Up

Remove any unnecessary code and ensure your project is saved.

By following the above steps, you will create a Java application that allows users to manage an inventory of different products, utilizing loops, exception handling, if statements, and arrays of objects.