

Joe-Turner-Adv_Soft-Dev-Assignment

by Joe Turner

Submission date: 24-Jan-2024 02:40AM (UTC+0000)

Submission ID: 222992211

File name: 32679_Joe_Turner_Joe-Turner-Adv_Soft-Dev-Assignment_532657_165509854.docx (2.92M)

Word count: 1385

Character count: 6974

Advanced Software Development

Joe Turner

2115463

Contents

Brief.....	2
Breakdown of Features	2
Concept Breakdown	3
Testing	4
Code	5

Brief

For this assignment, I have been tasked with creating an object oriented program that demonstrates the use of classes, attributes and methods, as well as inheritance and polymorphism. Following this criteria, I decided to create a shop inventory management system. The primary features of this application were an account system, where users log into the account, as well as a basic shopping basket system, where users can access products from different categories and add a particular amount to a basket, which will be added up in the checkout to give a total price.



Breakdown of Features

When making the program, I had to first decide on what the program would need to achieve to be feature complete. Several arrays would be created in order to store details efficiently, while a basket function would also need to be added, so a customer could add objects they wish to purchase.

To start the program, I implemented a simple login system; when the user starts the application, a prompt appears to press 1 to login and 2 to exit. Pressing 2 would immediately close the program, while pressing 1 would move onto the login screen. The login used an array to store multiple usernames and passwords, which if the user input matched, would allow the user to continue onto the main section of the program.

The main section of the program contains a list of different product categories customers navigate, sorting items by their type, such as dairy products and canned products. Users could navigate these sections by typing into a prompt ranging from 1 to 4, showing items in that category and adding items by typing the number corresponding to the item they wish to visit.

The user could then input how many of the item they wanted, which would then use a function called "multiply" to multiply the number of items entered by the price to find the total price (i.e. 3 cans of tomato soup would be multiplied by their price, 2, to display an overall total of 6), before being returned to the category selection screen. This total would then be stored in a "total" variable.

Invalid instructions would be ignored using a loop to ensure that the program would not break. I also wanted to implement a search feature that would display a product if their name was entered into a search box. However, I decided the feature would be too difficult and time consuming to implement in comparison to the project scope.

If the user pressed 5, they would be taken to a checkout, where each item was added together to calculate the values of each "total" variable using another function, addUp, which added two values together. This process was repeated until every item from each category was added to find the total price of the basket.

Finally, a payment section was added at the end of the checkout, which used additional arrays to store card details to be used for payment. Similarly to the login section, a user input would be compared to the values in the arrays, and if each value matched one in the arrays, the program would output a message confirming the details were correct, then end.

Additionally, an exit feature was added, which would simply close the program when entered. This was used at the beginning of the program before the login section, and in the main product section containing each product category and the checkout.

Concept Breakdown

Classes

During this project, I created two classes, `bankingDetails`, and `itemInfo`, in order to store variables and values for use in other parts of my code.

`BankingDetails` was created in order to store variables relating to the checkout section of the application, which were used to create the arrays that stored banking data.

`ItemInfo` was created as a simple class, with the intention of being used to create several other derived classes; when developing the basket, I initially used several different classes to store individual values for each item as well as a calculation multiplying the item value by the quantity that was in the basket; this was incredibly inefficient and time consuming.

Eventually I changed this to use a single class, `itemInfo` that determined just the value and quantity of an item and created derived classes for categories and items based on this class. The derived classes would then also be used for specific information for a product in that category, for example the derived class `bakery` took `itemInfo`'s value and quantity variables but also included `bakeDate`, a string used to store when the product was made.

Functions

In this program, I used two different functions, "multiply" and "addUp" in order to make my code faster and more efficient.

I created `multiply` to take two values and multiply them together. This was used after a user inputs the quantity of an item they wanted. `Multiply` would then multiply this quantity by the cost of the item to determine the overall price.

`AddUp` was created in order to speed up the process of adding up item totals to find the final basket price, for example `addUp` was used when adding the prices of milk and cheese, then cream cheese and sliced cheese before adding up both of those totals to find the total price for the dairy section.

Arrays

Arrays were used in the login and banking sections of the application in order to store details and values. In the login section, they stored usernames and passwords, which, if the user entered, would change the bool `userFound` to true, and allow the user to continue onto the rest of the program.

The banking section used arrays in a similar way, storing different details of a bank account that, if a user inputs those values, would change the `detailsFound` bool to true and end the program.

Inheritance

When creating classes, I created `itemInfo` to make use of inheritance by using `itemInfo` as a base for several derived classes. These classes would make use of `itemInfo`'s quantity and value variables, while containing their own separate variables. These were used for additional details relevant to different item categories, such as the dairy derived class using quantity and value, but also introducing a new variable, `milkQuantity`, to show the percentage of milk in each item in the dairy category.

Testing

Test 1: Normal functionality

Description: Run the program normally, without incorrect inputs.

Expected result: The application should run normally and finish without errors. Program output should return 0.

Result: Program ended normally. Program outputs correct in output log.



Test 2: Exit

Description: Run normally, exit when prompted (startup and main)

Expected result: Program should close without issue.

Result: Program exits normally.

Test 3: Invalid input (Prompt)

Description: When prompted for input, input wrong value (int 4 instead of int 1)

Expected result: Program outputs message, loops and prompts to try again.

Result: Error in main; resolved by adding userInputMain = 0; resetting the loop. If input is invalid in main, program does not close automatically after finishing checkout. Exit command, however, still functions as intended.

Test 4: Invalid input (Array)

Description: When prompted for input, input wrong value (value not in array)

Expected result: program outputs message, loops and prompts to try again.

Result: Program outputs message, loops and prompts to try again. Program does not close automatically after checkout.

Test 5: Invalid input (Wrong variable type)

Description: When prompted for input, input wrong value (string aaa instead of int 1)

Expected result: program outputs message, loops and prompts to try again.

Result: Program infinitely prints, potential solution is using getline instead of cin. The same error occurs when using special characters such as ~.

Test 6: Invalid input (high values and negative values)

Description: When prompted for input, input wrong value (10000000, -5)

Expected result: program outputs message, loops and prompts to try again.

Result: Program breaks, infinitely prints after inputting a value higher than 10 digits. Program accepts negative values; this should not happen, program should display an error message if a negative value is entered, potential solution is creating an if value < 0 catch.

Code

Below this are screenshots of the code used to create the program.

```
#include <string>
#include <fstream>
#include <iostream>

using namespace std;

int multiply(int a, int b) { // Multiplies items for the shopping basket
    return a * b;
}

int addUp(int a, int b) { // Adds items for the shopping basket
    return a + b;
}

int main() {
    const int userMaxCount = 10; // Defines the maximum amount of users that can use the program at once.
    while (userMaxCount > 11) { // While over the limit, forces users to exit
        cout << "We're sorry, we are currently experiencing high levels of traffic. Please try again later.";
        exit();
    };
    int i = 0;

    class bankingDetails {
    public:
        bankingDetails(int& username, int& password, int& accountID, string& price) {
            int bankNum = 1111222233334444;
            int serCode = 55556666; // User banking details
            int secNum = 666;
            string cardName = "John Johnson";
        }
    };

    int login(); // startup

    int startup;
    while (true) {
        cout << "Welcome to the shopping app. Press 1 to login, or 2 to exit the program. " << endl;
        cin >> startup; // User enters input
        if (startup == 1) { // If input = 1 break the loop
            cout << "Loading login..." << endl;
            break;
        }
        else if (startup == 2) { // If input = 2 exit the program
            cout << "Exiting program..." << endl;
            exit();
            return 0;
        }
    }
}
```

```

        else { // If input does not = 1 or 2, loop
            cout << "That input is invalid. Please try again. " << endl;
        }
    }
}

int userData(); {
    // Login

    string username[userMaxCount]; // Creating username array

    username[0] = { "1" };
    username[1] = { "username2" }; // Usernames
    username[2] = { "username" };

    string password[userMaxCount]; // Creating password array

    password[0] = { "1" };
    password[1] = { "password2" }; // Passwords
    password[2] = { "password" };

    string enteredUsername; // Stores entered username and password
    string enteredPassword;

    bool userFound = false; // Checking if username and password are correct; default set to false
    while (userFound == false) {
        cout << "Enter your username now: " << endl;
        cin >> enteredUsername; // User enters username

        cout << "Enter your password now: " << endl;
        cin >> enteredPassword; // User enters password

        if (username[1] == enteredUsername && password[1] == enteredPassword) { // If both username and password are correct
            userFound = true; // The user has been found; userFound is true
            cout << "You have successfully logged in." << endl;
            break;
        }
        else {
            cout << "User not found. Please try again." << endl; // User is not found; loops
        }
    }
}

```

```

int basket(); { // Creating the basket

    // main

    cout << "Welcome to the shop interface. Please select an option to continue." << endl;

    int userInput = 0; // User inputs for section screens
    int userInputMain = 0; // User input for main screen
    int userInputItem = 0; // User inputs for individual items
    int userInputAmount; // for how many of an item they want
    bool programDone = false;
    bool userExit = false; // For exit

    // Creating variables for totals; this is for the checkout later
    int allTotal = 0;
    int cheeseTotal = 0;
    int cCheeseTotal = 0;
    int sCheeseTotal = 0;

    int mBreadTotal = 0;
    int mBreadTotal = 0;
    int doughnutTotal = 0;
    int cookieTotal = 0;

    int tShopTotal = 0;
    int spaTotal = 0;
    int cFunaTotal = 0;

    int tShirtTotal = 0;
    int trousersTotal = 0;
    int coatTotal = 0;
    int shoeTotal = 0;

    class ItemInfo { // Basic class for item details; used by derived classes as a base to build from
    public:
        int value;
        int quantity;
    };

    while (userInputMain == 0) {

        cout << "1. Dairy" << endl << "2. Bakery" << endl << "3. Canned Goods" << endl << "4. Clothing products" << endl << "5. Checkout" << endl << "6. Exit" << endl;
        cin >> userInputMain;

        while (userInputMain == 1) { // Go to Dairy products

            class Dairy : public ItemInfo { // Creating derived class Dairy from ItemInfo

```

```

public:
    string milkQuantity;
};

cout << "Welcome to the dairy section." << endl << "1. Milk - £2" << endl << "2. Cheese - £2" << endl << "3. Cream Cheese - £2" << endl << "4. Sliced Cheese - £1" << endl;

cin >> userInput;

if (userInput == 1) {
    dairy milk; // Defines milk using ItemInfo class, creating a milk object
    milk.value = 2; // Price of item
    milk.quantity = 0; // Item wanted by customer default to 0
    milk.milkQuantity = "milk"; // % of milk in item
    cout << "Milk - " << milk.milkQuantity << " Milk - £2" << endl << "How many do you want? " << endl;
    userInput = 0; // Resets user input
    cin >> userInput;
    milk.quantity = userInput; // Sets the quantity to user entered amount
    milkTotal = multiply(milk.value, milk.quantity); // Multiply function to multiply milk cost by quantity
    cout << "That comes to £" << milkTotal << endl;
}

if (userInput == 2) {
    dairy cheese; // Defines cheese using ItemInfo class, creating a cheese object
    cheese.value = 2; // Price of item
    cheese.quantity = 0; // Item wanted by customer default to 0
    cheese.milkQuantity = "milk"; // % of milk in item
    cout << "Cheese - " << cheese.milkQuantity << " Milk - £2" << endl << "How many do you want? " << endl;
    userInput = 0; // Resets user input
    cin >> userInput;
    cheese.quantity = userInput; // Sets the quantity to user entered amount
    cheeseTotal = multiply(cheese.value, cheese.quantity); // Multiply function to multiply cheese cost by quantity
    cout << "That comes to £" << cheeseTotal << endl;
}

if (userInput == 3) {
    dairy cCheese; // Defines cream cheese using ItemInfo class, creating a cCheese object
    cCheese.value = 2; // Price of item
    cCheese.quantity = 0; // Item wanted by customer default to 0
    cCheese.milkQuantity = "milk"; // % of milk in item
    cout << "Cream Cheese - " << cCheese.milkQuantity << " Milk - £2" << endl << "How many do you want? " << endl;
    userInput = 0; // Resets user input
    cin >> userInput;
    cCheese.quantity = userInput; // Sets the quantity to user entered amount
    cCheeseTotal = multiply(cCheese.value, cCheese.quantity); // Multiply function to multiply cream cheese cost by quantity
    cout << "That comes to £" << cCheeseTotal << endl;
}

```

```

if (userInput == 4) {
    dairy sCheese; // Defines sliced cheese using ItemInfo class, creating a sCheese object
    sCheese.value = 1; // Price of item
    sCheese.quantity = 0; // Item wanted by customer default to 0
    sCheese.milkQuantity = "milk"; // % of milk in item
    cout << "Sliced Cheese - " << sCheese.milkQuantity << " Milk - £1" << endl << "How many do you want? " << endl;
    userInput = 0; // Resets user input
    cin >> userInput;
    sCheese.quantity = userInput; // Sets the quantity to user entered amount
    sCheeseTotal = multiply(sCheese.value, sCheese.quantity); // Multiply function to multiply cheese cost by quantity
    cout << "That comes to £" << sCheeseTotal << endl;
}

userInputMain = 0; // Sends user back to section selection screen
}

while (userInputMain == 2) { // Go to Bakery products
    class bakery : public ItemInfo { // Creating derived class bakery from ItemInfo
    public:
        string bakeDate;
    };

    cout << "Welcome to the bakery section." << endl << "1. White Bread - £2" << endl << "2. Brown Bread - £2" << endl << "3. Doughnuts - £3" << endl << "4. Cookies - £2" << endl;

    // userInput = 0; // Resetting userInput to 0
    cin >> userInput;

    if (userInput == 1) {
        bakery wBread; // Defines white bread using ItemInfo class, creating a wBread object
        wBread.value = 2; // Price of item
        wBread.quantity = 0; // Item wanted by customer default to 0
        wBread.bakeDate = "21/01/2024"; // Date item was made in bakery
        cout << "White Bread - Baked on " << wBread.bakeDate << " - £2" << endl << "How many do you want? " << endl;
        userInput = 0; // Resets user input
        cin >> userInput;
        wBread.quantity = userInput; // Sets the quantity to user entered amount
        wBreadTotal = multiply(wBread.value, wBread.quantity); // Multiply function to multiply bread cost by quantity
        cout << "That comes to £" << wBreadTotal << endl;
    }

    if (userInput == 2) {
        bakery bBread; // Defines brown bread using ItemInfo class, creating a bBread object
        bBread.value = 2; // Price of item
        bBread.quantity = 0; // Item wanted by customer default to 0
        bBread.bakeDate = "21/01/2024"; // Date item was made in bakery
        cout << "Brown Bread - Baked on " << bBread.bakeDate << " - £2" << endl << "How many do you want? " << endl;
        userInput = 0; // Resets user input
        cin >> userInput;
        bBread.quantity = userInput; // Sets the quantity to user entered amount
        bBreadTotal = multiply(bBread.value, bBread.quantity); // Multiply function to multiply bread cost by quantity
        cout << "That comes to £" << bBreadTotal << endl;
    }
}

userInputMain = 0; // Sends user back to section selection screen
}

if (userInputMain == 3) { // Go to Canned products
    class canned : public ItemInfo { // Creating derived class canned from ItemInfo
    public:
        string expirDate;
    };

    cout << "Welcome to the canned section." << endl << "1. Tomato Soup - £2" << endl << "2. Spas - £1" << endl << "3. Canned Tuna - £2" << endl;

    userInput = 0; // Resetting userInput to 0
    cin >> userInput;

    if (userInput == 1) {

```

```

        cin >> userInput;
        bBread.quantity = userInput; // Sets the quantity to user entered amount
        bBreadTotal = multiply(bBread.value, bBread.quantity); // Multiply function to multiply bread cost by quantity
        cout << "That comes to £" << bBreadTotal << endl;
    }

    if (userInput == 3) {
        bakery doughnut; // Defines doughnut using ItemInfo class, creating a doughnut object
        doughnut.value = 3; // Price of item
        doughnut.quantity = 0; // Item wanted by customer default to 0
        doughnut.bakeDate = "21/01/2024"; // Date item was made in bakery
        cout << "Doughnuts - Baked on " << doughnut.bakeDate << " - £3" << endl << "How many do you want? " << endl;
        userInput = 0; // Resets user input
        cin >> userInput;
        doughnut.quantity = userInput; // Sets the quantity to user entered amount
        doughnutTotal = multiply(doughnut.value, doughnut.quantity); // Multiply function to multiply doughnut cost by quantity
        cout << "That comes to £" << doughnutTotal << endl;
    }

    if (userInput == 4) {
        bakery cookie; // Defines cookie using ItemInfo class, creating a cookie object
        cookie.value = 2; // Price of item
        cookie.quantity = 0; // Item wanted by customer default to 0
        cookie.bakeDate = "21/01/2024"; // Date item was made in bakery
        cout << "Cookies - Baked on " << cookie.bakeDate << " - £2" << endl << "How many do you want? " << endl;
        userInput = 0; // Resets user input
        cin >> userInput;
        cookie.quantity = userInput; // Sets the quantity to user entered amount
        cookieTotal = multiply(cookie.value, cookie.quantity); // Multiply function to multiply cookie cost by quantity
        cout << "That comes to £" << cookieTotal << endl;
    }

    userInputMain = 0; // Sends user back to section selection screen
}

if (userInputMain == 3) { // Go to Canned products
    class canned : public ItemInfo { // Creating derived class canned from ItemInfo
    public:
        string expirDate;
    };

    cout << "Welcome to the canned section." << endl << "1. Tomato Soup - £2" << endl << "2. Spas - £1" << endl << "3. Canned Tuna - £2" << endl;

    userInput = 0; // Resetting userInput to 0
    cin >> userInput;

    if (userInput == 1) {

```

```

        canned tSoup; // Defines tomato soup using ItemInfo class; creating a tSoup object
        tSoup.value = 2; // Price of item
        tSoup.quantity = 0; // Items wanted by customer default to 0
        tSoup.expiresOn = "26/06/22"; // Date item will expire
        cout << "Tomato Soup - Expires on " << tSoup.expiresOn << " - £2" << endl << "How many do you want? " << endl;
        userInputItem = 0; // Resets user input
        cin >> userInputItem;
        tSoup.quantity = userInputItem; // Sets the quantity to user entered amount
        tSoupTotal = multiply(tSoup.value, tSoup.quantity); // Multiply function to multiply soup cost by quantity
        cout << "That comes to £" << tSoupTotal << endl;
    }

    if (userInput == 2) {
        canned sSoup; // Defines spam using ItemInfo class; creating a sSoup object
        sSoup.value = 1; // Price of item
        sSoup.quantity = 0; // Items wanted by customer default to 0
        sSoup.expiresOn = "26/06/22"; // Date item will expire
        cout << "Tomato Soup - Expires on " << sSoup.expiresOn << " - £1" << endl << "How many do you want? " << endl;
        userInputItem = 0; // Resets user input
        cin >> userInputItem;
        sSoup.quantity = userInputItem; // Sets the quantity to user entered amount
        sSoupTotal = multiply(sSoup.value, sSoup.quantity); // Multiply function to multiply spam cost by quantity
        cout << "That comes to £" << sSoupTotal << endl;
    }

    if (userInput == 3) {
        canned cTuna; // Defines canned tuna using ItemInfo class; creating a cTuna object
        cTuna.value = 2; // Price of item
        cTuna.quantity = 0; // Items wanted by customer default to 0
        cTuna.expiresOn = "25/12/24"; // Date item will expire
        cout << "Canned Tuna - Expires on " << cTuna.expiresOn << " - £3" << endl << "How many do you want? " << endl;
        userInputItem = 0; // Resets user input
        cin >> userInputItem;
        cTuna.quantity = userInputItem; // Sets the quantity to user entered amount
        cTunaTotal = multiply(cTuna.value, cTuna.quantity); // Multiply function to multiply tuna cost by quantity
        cout << "That comes to £" << cTunaTotal << endl;
    }

    userInputMain = 0; // Sends user back to section selection screen
}

if (userInputMain == 4) { // Go to Clothing products

    class clothing : public ItemInfo { // Creating derived class clothing from ItemInfo
    public:
        string percentCotton;
    };
}

```

```

cout << "Welcome to the clothing section. " << endl << "1. T-Shirt - £5" << endl << "2. Trousers - £8" << endl << "3. Coat  

userInput = 0; // Resetting userInput to 0
cin >> userInput;

if (userInput == 1) {
    clothing tShirt; // Defines tshirt using ItemInfo class; creating a tShirt object
    tShirt.value = 5; // Price of item
    tShirt.quantity = 0; // Items wanted by customer default to 0
    tShirt.percentCotton = "70%"; // Percentage of cotton in item
    cout << "T-Shirt - " << tShirt.percentCotton << " Cotton - £5" << endl << "How many do you want? " << endl;
    userInputItem = 0; // Resets user input
    cin >> userInputItem;
    tShirt.quantity = userInputItem; // Sets the quantity to user entered amount
    tShirtTotal = multiply(tShirt.value, tShirt.quantity); // Multiply function to multiply tshirt cost by quantity
    cout << "That comes to £" << tShirtTotal << endl;
}

if (userInput == 2) {
    clothing trousers; // Defines trousers using ItemInfo class; creating a trousers object
    trousers.value = 8; // Price of item
    trousers.quantity = 0; // Items wanted by customer default to 0
    trousers.percentCotton = "80%"; // Percentage of cotton in item
    cout << "Trousers - " << trousers.percentCotton << " Cotton - £8" << endl << "How many do you want? " << endl;
    userInputItem = 0; // Resets user input
    cin >> userInputItem;
    trousers.quantity = userInputItem; // Sets the quantity to user entered amount
    trousersTotal = multiply(trousers.value, trousers.quantity); // Multiply function to multiply trouser cost by quantity
    cout << "That comes to £" << trousersTotal << endl;
}

if (userInput == 3) {
    clothing coat; // Defines coat using ItemInfo class; creating a coat object
    coat.value = 28; // Price of item
    coat.quantity = 0; // Items wanted by customer default to 0
    coat.percentCotton = "20%"; // Percentage of cotton in item
    cout << "Coat - " << coat.percentCotton << " Cotton - £28" << endl << "How many do you want? " << endl;
    userInputItem = 0; // Resets user input
    cin >> userInputItem;
    coat.quantity = userInputItem; // Sets the quantity to user entered amount
    coatTotal = multiply(coat.value, coat.quantity); // Multiply function to multiply coat cost by quantity
    cout << "That comes to £" << coatTotal << endl;
}

if (userInput == 4) {
    clothing shoes; // Defines shoes using ItemInfo class; creating a shoes object
    shoes.value = 18; // Price of item
    shoes.quantity = 0; // Items wanted by customer default to 0
}

```

```

shoes.percentCotton = "10%"; // Percentage of cotton in item
cout << "Shoes - " << shoes.percentCotton << " Cotton - $10" << endl << "How many do you want? " << endl;
userInputItem = 0; // Resets user input
cin >> userInputItem;
shoes.quantity = userInputItem; // Sets the quantity to user entered amount
shoesTotal = multiply(shoes.value, shoes.quantity); // Multiply function to multiply shoes cost by quantity
cout << "That comes to: $" << shoesTotal << endl;
}

userInputMain = 0; // Sends user back to section selection screen
}

if (userInputMain == 5) { // Go to Checkout
    // checkout
    // Adds up every item to get the overall cost
    int dairyTotal = addip(cellaTotal, cheeseTotal);
    int dairyTotal2 = addip(cCheeseTotal, sCheeseTotal);
    int dairyTotal = addip(dairyTotal, dairyTotal2);

    int bakeryTotal = addip(breadTotal, breadTotal);
    int bakeryTotal2 = addip(doughnutTotal, cookieTotal);
    int bakeryTotal = addip(bakeryTotal, bakeryTotal2);

    int cannedTotal = addip(soupTotal, spamTotal);
    int cannedTotal = addip(cannedTotal, cFunaTotal);

    int clothingTotal = addip(tshirtTotal, trousersTotal);
    int clothingTotal2 = addip(costTotal, shoesTotal);
    int clothingTotal = addip(clothingTotal, clothingTotal2);

    int totalTotal = addip(dairyTotal, bakeryTotal);
    int totalTotal2 = addip(cannedTotal, clothingTotal);
    int totalTotal = addip(totalTotal, totalTotal2);

    cout << "Your order comes to: $" << totalTotal << endl;

    // banking
    // Getting card details
    string bankNum[userMaxCount]; // Creating card name array
    bankNum[0] = {"1"};
    bankNum[1] = {"123456789"}; // Bank Number array
    bankNum[2] = {"9181674891830927"};

    string sortCode[userMaxCount]; // Creating sort code array

```

```

sortCode[0] = {"1"};
sortCode[1] = {"123456789"}; // Sorting code array
sortCode[2] = {"888888888"};

string secNum[userMaxCount]; // Creating security number array
secNum[0] = {"1"};
secNum[1] = {"1234"}; // Security number array
secNum[2] = {"4567"};

string cardName[userMaxCount]; // Creating card name array
cardName[0] = {"1"};
cardName[1] = {"John"}; // Card name array
cardName[2] = {"James"};

string enteredBankNum; // Variables for user entered values
string enteredSort;
string enteredSec;
string enteredName;

bool detailsFound = false; // Checking if banking details are correct; default set to false
while (detailsFound == false) {
    cout << "Enter your bank card number now: " << endl;
    cin >> enteredBankNum; // User enters card number

    cout << "Enter your sort number now: " << endl;
    cin >> enteredSort; // User enters sort number

    cout << "Enter your security number now: " << endl;
    cin >> enteredSec; // User enters security number

    cout << "Enter your card name now: " << endl;
    cin >> enteredName; // User enters card name

    if (bankNum[i] == enteredBankNum && sortCode[i] == enteredSort) { // If both bankNum and sortCode are correct,
        if (secNum[i] == enteredSec && cardName[i] == enteredName) { // Then if secNum and cardName are also correct,
            detailsFound = true; // The details have been confirmed; detailsFound is true
            cout << "Bank details confirmed. Have a nice day!" << endl;
            programDone = true; // The program is done
            exit(); // Redundancy
        }
        else {
            cout << "Invalid details. Please try again." << endl; // User is not found; loops
        }
    }
    else {

```

```

    }
}

else {
    cout << "Invalid details. Please try again." << endl; // User is not found; loops
}
}

if (userInputMain == 6) { // Exiting program
    exit;
}

else {
    "Invalid input, please try again."; // Invalid value entered, loops
}
}
}

```

Joe-Turner-Adv_Soft-Dev-Assignment

GRADEMARK REPORT

FINAL GRADE

40/100

GENERAL COMMENTS

Hi Joe,

Well attempted. In-depth feedback is provided within the document, and a summary is presented here. Scores for individual sections are accessible through the rubric.

Areas of appreciation:

The breakdown of features provides a clear and logical explanation of how the program functions, from the login system to the checkout process.

The use of functions like "multiply" and "addUp" contributes to the efficiency of the code, improving readability and maintainability.

areas of improvement:

adding more robust validation for user inputs, especially in scenarios where unexpected input types (e.g., strings for integers) can cause the program to behave unexpectedly could have been considered.

Reconsider the decision to omit the search feature, as it can enhance the user experience by allowing users to locate specific products more efficiently.

Completion of the course from CISCO to a possible extent and a video demonstration could have added credits



Comment 1

good start mentioning a brief introduction about the planned application

PAGE 3



Comment 2

screen shots of particular sections from program code could have been added

PAGE 4



Comment 3

what type of inputs presented and its corresponding outputs should have been presented

PAGE 5

PAGE 6

PAGE 7



Comment 4

screenshots could have been clearly labelled to help in easy navigation

PAGE 8

PAGE 9



Comment 5

a conclusion could better enhance the report

GRADING FORM: DC-ASD

JOE TURNER

40

LITERATURE REVIEW

with IEEE referencing



3

KNOWLEDGE&UNDERSTANDING SKILLS

use of OOPs features



9

COGNITIVE&INTELLUCTUAL SKILLS

Testing &Error handling



12

PRACTICAL SKILLS

Working program Demo video



13

TRANSFERRABLE SKILLS

User Manual, document style



3

CISCO CERTIFICATE



0