

**REPORT**  
of  
**E-commerce ConsoleApp**  
for  
**OOP Lesson' s Assignment**

1.Intoduction	-
1.a Required informations	-
1.b Scratch ideas	-
1.c Explanation of App (Superficial)	-
2. System of Application	-
1.a UML Diagrams	-
1.b Methods	-
1.c Consol results screenshots	-
3.Outcome	-
1.a What i learned?	-

Eren Çetiner  
200315070  
CBU Computer Engineering Student 1st Edu.  
01/06/2021

# 1.Intoduction

## 1.a Required informations

In this assignment, you are expected to benefit from OOP principles for an e-commerce application.

Within the application;

- There must be a User class that holds the username, name, surname, date of birth, password, email

address, home and work addresses, products ordered, favorite products, and credit card objects

belonging to the user. In this class, there should also be two methods that perform the user's product

ordering and product favorites.

- There must be a CreditCard class that contains the credit card number, credit card user, security code

and expiration date of the credit cards.

- There must be a Product class that holds the product name, product color, product category, product

stock information, product weight, product description information. In addition, when a product is

ordered by users, a method should be written that reduces the number of stocks as much as the number

of products purchased and controls the stock number.

- For order transactions, there must be an Order class that holds the ordering user object, the ordered

product object, and the credit card objects to which the payment is made. In addition, a method must be

used here that accesses the method that controls the product stock information of the Product class and

accesses the purchasing method of the User class.

- You are also expected to develop a test class that checks all the operations you do. (For adding users,

adding credit cards, adding products, purchasing products, favoring products)

- Getter and setter methods of variables in all classes should be written..

You need to write a homework report detailing all classes and methods of the application you have developed. It is very important that your project report is organized and your report should be in PDF format. You have to compress the project folder and report and upload the assignment until 1st June at 00:00

### 1.b Strach ideas

Customer could do;

- \*Register with name, mail and password

- \*Login

- \*Exit App

- \*Buy an Item

- \*Return an Item

- \*Log out.

### 1.c Explanation of App (Superficial)

- \*Application is active on console.

- \*It has a simple interface.

- \*There are few products

- \*You can buy or return an any item.

- \*Login&out with mail and password

- \*Basic commands(1, 2, 3, y, n, enter)

## 2. System of Application

### 2.a UML Diagrams

e\_commerce\_app\_oop\_homework\_200315070\_erencetiner.src



e\_commerce\_app\_oop\_homework\_200315070\_erencetiner.src



e\_commerce\_app\_oop\_homework\_200315070\_erencetiner.src

EcommerceController.java /

**public class** EcommerceController{ }

EcommerceController( )

displayMainMenu( )

displayCustMenu( )

register( )

showItems( )

showInvoice( )

buyItem( )

ReplaceItem( )

StartBrowsing( )

## 2.b Methods

```
public static void main( String[] args ){  
  
    try{  
  
        EcommerceController ecommerceController = new EcommerceController();  
        ecommerceController.startBrowsing();  
  
    }  
  
    catch(Exception e){  
  
        System.out.println("The ecommerce platform is having issues loading");  
        System.out.println("Try again later!");  
  
    }  
  
}
```

Main method

```
/**  
 * constructor for controller object and to add test data into in-memory  
 * collections  
 */  
public EcommerceController() {  
  
    customers = new HashMap<String, Customer>();  
    items = new HashMap<Long, Item>();  
    dataGeneratorUtil.generateCustomers(customers);  
    dataGeneratorUtil.generateItems(items);  
    invoices = new HashMap<Long, Invoice>();  
    itemCodes = new HashMap<String, Item>();  
    dataGeneratorUtil.generateInvoices(invoices, itemCodes);  
  
}
```

Constructor for controller object and to add test data into in-memory collections.

```

/**
 * Main entry point menu for console app for startBrowsing method that returns
 * an int for user choice.
 *
 * @return selected - User menu prompt selection of type int. returns a choice
 * or exception if invalid input
 */

private int displayMainMenu() {

    String choiceEntered;
    int selected;
    System.out.println("=====+");
    System.out.println("| Welcome to ErenCetinerOOP Ecommerce! |");
    System.out.println("|1. Register |");
    System.out.println("|2. Login |");
    System.out.println("|3. Exit app. |");
    System.out.println("=====+");

    while (true) {

        System.out.println("\nEnter Choice (1, 2, 3) : ");
        choiceEntered = consoleScan.nextLine();

        try {
            selected = Integer.parseInt(choiceEntered);
            break;
        } catch (Exception exception) {

            System.out.println("Enter a valid choice (1, 2, 3). Try Again!");

        }

    }

    System.out.println();
    return selected;
}

```

Main entry point menu for console app for startBrowsing method that returns an int for user choice. User menu prompt selection of type int. returns a choice or exception if invalid input

```

/**
 * Display a submenu for customer after a successful login that returns an int
 *
 * @return selected - User menu prompt selection of type int. returns a choice
 * or exception if invalid input
 */

private int displayCustMenu() {
    String inputToParse;
    int selected;

    System.out.println("=====Customer Menu=====");
    System.out.println("|1. Buy an Item          |");
    System.out.println("|2. Return an Item       |");
    System.out.println("|3. Log out              |"); // return user to to main menu
    System.out.println("=====");

    while (true) {

        System.out.println("Enter a number (1,2,3) from Customer menu : ");
        inputToParse = consoleScan.nextLine();

        try {
            selected = Integer.parseInt(inputToParse);
            break;
        }
        catch (Exception e) {

            System.out.println("Something happened with the try!");
            System.out.println("Please enter a valid choice. Try again!");
        }

    }

    return selected;
}

```

Display a submenu for customer after a successful login that returns an int

User menu prompt selection of type int. returns a choice or exception if invalid input

```

// Methods

// private void login() {}

/**
 * method for creating a customer account
 */

private void register() {

    Customer customer;
    String email;
    String password;
    String confirmPassword;
    String name;
    System.out.println("Enter your details for a new customer account");
    System.out.println("Name: ");
    name = consoleScan.nextLine();
    System.out.println("Email: ");
    email = consoleScan.nextLine();

    while (customers.containsKey(email)) {

        System.out.println("Email already exists, Try again..");
        email = consoleScan.nextLine();
    }

    System.out.println("Password: ");
    password = consoleScan.nextLine();
    System.out.println("Confirm password: ");
    confirmPassword = consoleScan.nextLine();

    // password check for matching strings.

    try {

        if (password.equals(confirmPassword)) {
            customer = new Customer(name, password, email);
            customers.put(email, customer);
            System.out.println("Registration Successful");
        }
        else {
            System.out.println("Password does not match. try again");
        }

    }
    catch (Exception e) {
        System.out.println("something happened, try again");
    }

}

```

Method for creating a customer account. Password check for matching strings.



```

/**
 * Method to display all items for sale from datasource or testdata
 */
private void showItems() {
    Set set = items.entrySet();
    Iterator iterator = set.iterator();
    System.out.println("\n+=====Items in Stock=====+");
    System.out.print("|ProductId \t Name \t\t Item Code \t Price |\t \n");

    if (items.isEmpty()) {
        System.out.println("No purchases have been made on this account ");
    }

    while (iterator.hasNext()) {
        Map.Entry mentry = (Map.Entry) iterator.next();
        System.out.println("|" + mentry.getKey() + ". " + mentry.getValue() + "|");
    }

    System.out.println("+=====+");
}

```

Method to display all items for sale from datasource or testdata.

```

private void showInvoices() {

    Set set = invoices.entrySet();
    Iterator iterator = set.iterator();
    System.out.println("\n+=====+");

    while (iterator.hasNext()) {
        Map.Entry mentry = (Map.Entry) iterator.next();
        System.out.println("|" + mentry.getValue() + "|");
    }

    System.out.println("+=====+");
}

```

```

/**
 * Method for choosing an item to buy by its key and making it into an invoice
 * by looking for its key value in hashmap.
 */

private void buyItem(Customer c) {

    // Show list of items
    String choice;
    Long selected;
    Item item; // add values of item to Invoice

    showItems();

    // select an item to buy and search for it by its key in the hashmap
    System.out.println("Enter the product Id of the item you want to purchase");

    // choice
    choice = consoleScan.nextLine();
    selected = Long.parseLong(choice);

    // Extra credit Idea/challenge: ask for how much of an idea to buy/quantity and
    // get the invoice to show the total of all item elements in item list
    // Extra // ask if a user wants to purchase a different item and prompt if they
    // are finished.

    try {

        if (items.containsKey(selected)) {

            item = items.get(selected);
            long min = 1L;
            long max = 100L;
            long invoiceNo = min + (long) (Math.random() * (max - min));

            Invoice = new Invoice(invoiceNo, c, item, item.getItemTotal());
            invoices.put(invoiceNo, Invoice);
            itemCodes.put(item.getItemCode(), item);
            Invoice.showInvoiceDetails();
            Invoice.toString();

            System.out.println(Invoice.showInvoiceDetails());
            System.out.println(Invoice.toString());
            System.out.println("Press Enter to continue");
            System.in.read();

        }

    } catch (Exception e) {

        System.out.println("Please enter the product number from the catalog.");

    }

}

```

Method for choosing an item to buy by its key and making it into an invoice by looking for its key value in hashmap.

Show list of items

Select an item to buy and search for it by its key in the hashmap

Extra credit Idea/challenge: ask for how much of an idea to buy

quantity and get the invoice to show the total of all item elements in item list

Extra

ask if a user wants to purchase a different item and prompt if they are finished.

```

private void ReplaceItem(Customer c) {
    String input;
    String choice;
    Item itemToReplace;
    Long invoiceNo;
    Invoice enteredInvoice;
    System.out.println("Welcome " + c.getEmail() + "!! Your invoice details are:");
    showInvoices();

    try {
        System.out.println("Enter the invoice number.");
        input = consoleScan.nextLine();
        invoiceNo = Long.parseLong(input);

        if (invoices.containsKey(invoiceNo)) {
            System.out.println("Invoice found");
            enteredInvoice = invoices.get(invoiceNo);
            System.out.println("Enter the itemcode of the item to replace.");
            input = consoleScan.nextLine();

            if (itemCodes.containsKey(input)) {
                itemToReplace = itemCodes.get(input);
                long validDate = ChronoUnit.DAYS.between(enteredInvoice.getPurchaseDate(), LocalDate.now());

                if (validDate >= -15) {
                    System.out.println(
                        "Yes, you can return your purchase. Would you like to proceed (enter y for yes or n for no.)");
                    choice = consoleScan.nextLine();

                    if (choice.equals("y")) {
                        System.out.println("Return successful, Your replaced item is" + itemToReplace.toString());
                        System.out.println("Press Enter to continue");
                        System.in.read();
                    } else if (choice.equals("n")) {
                        System.out.println("Returning back to customer menu");
                        displayCurrMenu();
                    } else {
                        System.out.println("Not valid input");
                    }
                } else {
                    System.out.println(
                        "Your purchase is past the 15 day period. You are not able to return or replace your item(s)");
                    System.out.println("Press Enter to continue");
                    System.in.read();
                }
            } else {
                System.out.println("You dont have this item in your invoice");
            }
        }
    } catch (Exception e) {
        System.out.println("Not a valid choice");
    }
}

```

Method for checking a purchase by invoice and purchase date and returning an item within a 15 day return policy

```

public void startBrowsing() {

    try {
        Customer customer;
        String cEmail;
        String cPassword;
        int input; // local function scope
        mainloop: while (true) {
            input = displayMainMenu();

            switch (input) {
                case 1:
                    register();
                    break;
                case 2:
                    System.out.println("Enter Email and Password:");
                    System.out.println("Email : ");
                    cEmail = console.readLine();
                    System.out.println("Password : ");
                    cPassword = console.readLine();
                    System.out.println();

                    if (customers.containsKey(cEmail)) {
                        customer = customers.get(cEmail);
                        System.out.println("\t-----Login Successful!-----");
                        System.out.println("Welcome " + customer.getName());

                        if (customer.getPassword().equals(cPassword)) {
                            while (true) {
                                showItems();
                                input = displayCustMenu();

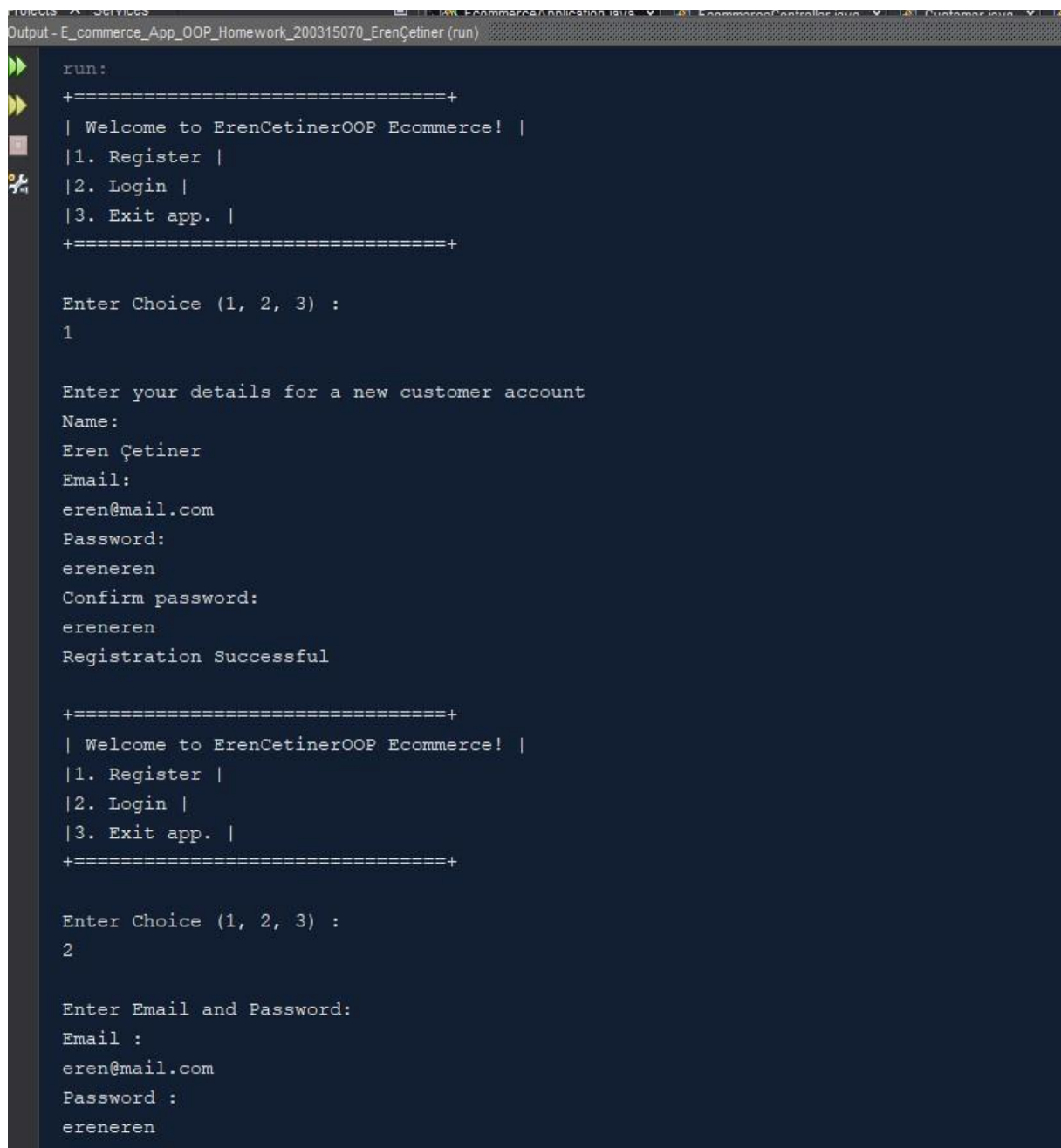
                                switch (input) {
                                    case 1:
                                        buyItem(customer);
                                        System.in.read();
                                        break;
                                    case 2:
                                        replaceItem(customer);
                                        System.in.read();
                                        break;
                                    case 3: {
                                        System.out.println("Signed out successfully");
                                        continue mainloop;
                                    }
                                    default:
                                        System.out.println("Enter a valid choice");
                                }
                            }
                        } else {
                            System.out.println("Invalid Credentials..Try again");
                        }
                    } else {
                        System.out.println("Invalid Credentials..Try again");
                    }
                    break;
                case 3:
                    System.out.println("Thank you for using the ErenCetinerOOP Ecommerce");
                    System.exit(1);
                    break;
                default:
                    System.out.println("Enter a Valid Option!");
                    System.out.println();
            }
        }
    } catch (Exception exception) {
        System.out.println("Enter a valid choice. Try Again!");
        System.out.println(exception);
    }
}

```

Method for starting up ecommerce menu for Customer registration Customer

Login Buying an item Replacing an item if broken Exit the program

## 2.c Console results screenshots



```
run:
+=====+
| Welcome to ErenCetinerOOP Ecommerce! |
|1. Register |
|2. Login |
|3. Exit app. |
+=====+

Enter Choice (1, 2, 3) :
1

Enter your details for a new customer account
Name:
Eren Çetiner
Email:
eren@mail.com
Password:
ereeneren
Confirm password:
ereeneren
Registration Successful

+=====+
| Welcome to ErenCetinerOOP Ecommerce! |
|1. Register |
|2. Login |
|3. Exit app. |
+=====+

Enter Choice (1, 2, 3) :
2

Enter Email and Password:
Email :
eren@mail.com
Password :
ereeneren
```

```

+=====Login Successful!=====+
Welcome Eren Detiner

+=====Items in Stock=====+
|ProductId      Name                Item Code      Price |
|1. Product [ItemNo=1, Item name=MSI GS66 Stealth, itemCode=MSI01, itemPrice=29999.99, quantity=1, itemTotal=29999.99]|
|2. Product [ItemNo=2, Item name=Adidas Futbol Topu, itemCode=D01, itemPrice=299.99, quantity=1, itemTotal=29999.99]|
|3. Product [ItemNo=3, Item name=GeForce RTX 3090, itemCode=GPU01, itemPrice=39999.99, quantity=1, itemTotal=39999.99]|
|4. Product [ItemNo=4, Item name=Ds Damat Slim Fit Mavi Gömlek, itemCode=MG01, itemPrice=199.99, quantity=1, itemTotal=199.99]|
+=====+
+=====Customer Menu=====+
|1. Buy an Item          |
|2. Return an Item       |
|3. Log out              |
+=====+
Enter a number (1,2,3) from Customer menu :
1

+=====Items in Stock=====+
|ProductId      Name                Item Code      Price |
|1. Product [ItemNo=1, Item name=MSI GS66 Stealth, itemCode=MSI01, itemPrice=29999.99, quantity=1, itemTotal=29999.99]|
|2. Product [ItemNo=2, Item name=Adidas Futbol Topu, itemCode=D01, itemPrice=299.99, quantity=1, itemTotal=29999.99]|
|3. Product [ItemNo=3, Item name=GeForce RTX 3090, itemCode=GPU01, itemPrice=39999.99, quantity=1, itemTotal=39999.99]|
|4. Product [ItemNo=4, Item name=Ds Damat Slim Fit Mavi Gömlek, itemCode=MG01, itemPrice=199.99, quantity=1, itemTotal=199.99]|
+=====+
Enter the product Id of the item you want to purchase

Enter a valid choice. Try Again!
java.lang.NumberFormatException: For input string: ""
BUILD SUCCESSFUL (total time: 1 minute 20 seconds)

```

## 3.Outcome

### 3.a What i learned?

- \*Responsibility
- \*New coding skills
- \*Systematic work