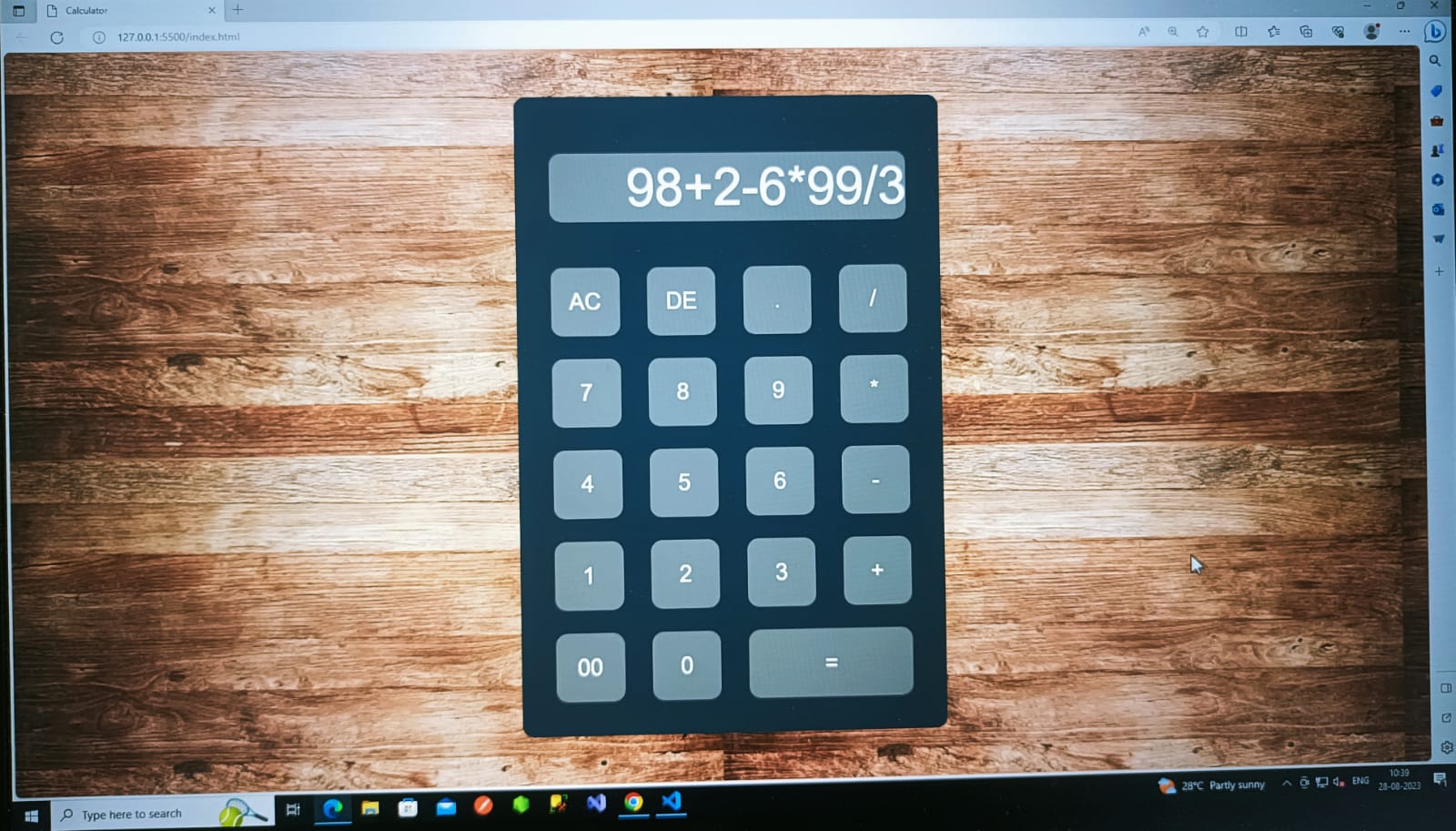
**Assignment on Foundation Boot camp Batch 2023**

**Name :- Poulami Deb**

**Emploee\_ID- 101439993**

1. **Make a fully operational calculator using HTML , CSS , JavaScript**



**JavaScript Code**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Calculator</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<div class="container">

<div class="calculator">

<form>

<div class="display">

<input type="text" name="display">

</div>

<div>

<input type="button" value="AC" onclick="display.value = '' ">

<input type="button" value="DE" onclick="display.value = display.value.toString().slice(0,-1)">

<input type="button" value="." onclick="display.value += '.' ">

<input type="button" value="/" onclick="display.value += '/' ">

</div>

<div>

<input type="button" value="7" onclick="display.value += '7' ">

<input type="button" value="8" onclick="display.value += '8' ">

<input type="button" value="9" onclick="display.value += '9' ">

<input type="button" value="\*" onclick="display.value += '\*' ">

</div>

<div>

<input type="button" value="4" onclick="display.value += '4' ">

<input type="button" value="5" onclick="display.value += '5' ">

<input type="button" value="6" onclick="display.value += '6' ">

<input type="button" value="-" onclick="display.value += '-' ">

</div>

<div>

<input type="button" value="1" onclick="display.value += '1' ">

<input type="button" value="2" onclick="display.value += '2' ">

<input type="button" value="3" onclick="display.value += '3' ">

<input type="button" value="+" onclick="display.value += '+' ">

</div>

<div>

<input type="button" value="00" onclick="display.value += '00' ">

<input type="button" value="0" onclick="display.value += '0' ">

<input type="button" value="=" onclick="display.value =eval(display.value)"" class="equal">

</div>

</form>

</div>

</div>

</body>

</html>

**CSS Code**

\*{

margin: 0;

padding:0;

font-family:'Poppins',sans-serif;

box-sizing: border-box;

}

.container{

width: 100%;

height: 100vh;

background-color: rgb(221, 132, 24);

background-image: url("https://media.istockphoto.com/id/624697496/photo/distressed-wooden-boards.jpg?s=612x612&w=0&k=20&c=Y4j6HgrHByyJeTl235u1m89gRhDCpat8I2z57YkHgSM=");

box-shadow: 5px 10px #888888;

display: flex;

align-items: center;

justify-content: center;

}

.calculator{

background: #3a4452;

padding: 20px;

border-radius: 10px;

}

.calculator form input{

border: 0px;

outline: 0;

width:60px;

height:60px;

border-radius: 10px;

box-shadow: whitesmoke;

background: grey;

font-size: 20px;

color: #fff;

cursor:pointer;

margin:10px;

}

form .display {

display:flex;

justify-content:flex-end;

margin:20px 0;

}

form .display input{

text-align: right;

flex:1;

font-size: 45px;

box-shadow:none;

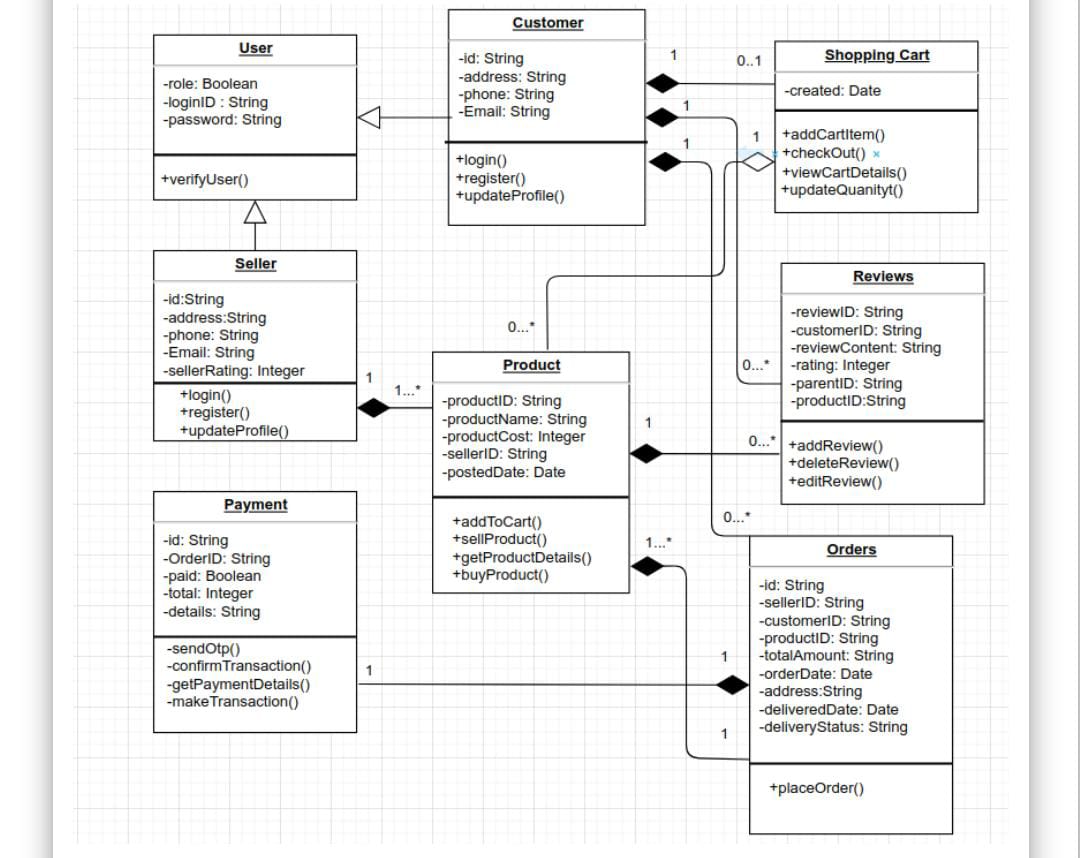
}

form input.equal{

width:145px;

}

1. **Using Java give the working flow (code) of the following UML diagram.**



//User class

class User {

private boolean role;

private String loginID;

private String password;

public User(boolean role, String loginID, String password) {

this.role = role;

this.loginID = loginID;

this.password = password;

}

public boolean verifyUser(String providedLoginID, String providedPassword) {

// Compare provided login ID and password with stored values

return loginID.equals(providedLoginID) && password.equals(providedPassword);

}

public static void main(String[] args) {

User user = new User(false, "Poulami\_Deb", "secretpass");

String providedLoginID = "Poulami\_Deb";

String providedPassword = "secretpass";

if (user.verifyUser(providedLoginID, providedPassword)) {

System.out.println("User verified successfully.");

} else {

System.out.println("User verification failed.");

}

}

}

//Customer Class

class Customer extends User {

private int id;

private String address;

private String phone;

private String email;

private ShoppingCart cart;

private List<Review> reviews;

private List<Order> orders;

public Customer(boolean role, String loginID, String password, int id, String address, String phone, String email) {

super(role, loginID, password);

this.id = id;

this.address = address;

this.phone = phone;

this.email = email;

this.cart = new ShoppingCart();

this.reviews = new ArrayList<>();

this.orders = new ArrayList<>();

}

public void login() {

if (verifyUser()) {

System.out.println("Login successful");

} else {

System.out.println("Login failed");

}

}

public void register() {

System.out.println("Customer registered");

}

public void updateProfile(String newAddress, String newPhone, String newEmail) {

this.address = newAddress;

this.phone = newPhone;

this.email = newEmail;

System.out.println("Profile updated");

}

}

//Seller Class

class Seller extends User {

private int id;

private String address;

private String phone;

private String email;

private int sellerRating;

public Seller(boolean role, String loginID, String password, int id, String address, String phone, String email) {

super(role, loginID, password);

this.id = id;

this.address = address;

this.phone = phone;

this.email = email;

this.sellerRating = 0;

}

public void login() {

if (verifyUser()) {

System.out.println("Login successful");

} else {

System.out.println("Login failed");

}

}

public void register() {

System.out.println("Seller registered");

}

public void updateProfile(String newAddress, String newPhone, String newEmail) {

this.address = newAddress;

this.phone = newPhone;

this.email = newEmail;

System.out.println("Profile updated");

}

}

//Shopping cart

class ShoppingCart {

private Date created;

private List<Product> items;

public ShoppingCart() {

this.created = new Date();

this.items = new ArrayList<>();

}

public void addCartItem(Product product) {

items.add(product);

System.out.println("Product added to cart");

}

public void checkOut() {

double totalAmount = calculateTotalAmount();

Order order = createOrder(totalAmount);

Payment payment = initiatePayment(order);

System.out.println("Checkout completed");

}

public void viewCartDetails() {

for (Product item : items) {

System.out.println("Product: " + item.getProductName() + ", Price: " + item.getProductCost());

}

}

public void updateQuantity(Product product, int newQuantity) {

for (Product item : items) {

if (item.equals(product)) {

item.setQuantity(newQuantity);

System.out.println("Quantity updated");

return;

}

}

System.out.println("Product not found in cart");

}

}

}

//Review

class Review {

private int reviewID;

private int customerID;

private String reviewContent;

private int rating;

private int parentID; // If it's a reply to another review

private int productID;

public Review(int reviewID, int customerID, String reviewContent, int rating, int parentID, int productID) {

this.reviewID = reviewID;

this.customerID = customerID;

this.reviewContent = reviewContent;

this.rating = rating;

this.parentID = parentID;

this.productID = productID;

}

public void addReview() {

Review newReview = new Review(reviewID, customerID, reviewContent, rating, parentID, productID);

product.addReview(newReview);

System.out.println("Review added");

}

public void deleteReview() {

product.removeReview(this);

System.out.println("Review deleted");

}

public void editReview(String newContent, int newRating) {

this.reviewContent = newContent;

this.rating = newRating;

System.out.println("Review edited");

}

}

//Product Class

class Product {

private int productID;

private String productName;

private double productCost;

private int sellerID;

private Date postedDate;

private List<Review> reviews;

public Product(int productID, String productName, double productCost, int sellerID, Date postedDate) {

this.productID = productID;

this.productName = productName;

this.productCost = productCost;

this.sellerID = sellerID;

this.postedDate = postedDate;

this.reviews = new ArrayList<>();

}

public void addToCart(ShoppingCart cart) {

cart.addCartItem(this);

System.out.println("Product added to cart");

}

public void getProductDetails() {

System.out.println("Product Name: " + productName);

System.out.println("Price: " + productCost);

System.out.println("Seller: " + sellerID);

}

public void buyProduct() {

ShoppingCart cart = customer.getCart();

addToCart(cart);

cart.checkOut();

System.out.println("Product purchased");

}

}

//Order Class

class Order {

private int orderID;

private int sellerID;

private int customerID;

private List<Product> products;

private double totalAmount;

private Date orderDate;

private String address;

private Date deliveryDate;

private String deliveryStatus;

public Order(int orderID, int sellerID, int customerID, List<Product> products, double totalAmount, Date orderDate, String address, Date deliveryDate, String deliveryStatus) {

this.orderID = orderID;

this.sellerID = sellerID;

this.customerID = customerID;

this.products = products;

this.totalAmount = totalAmount;

this.orderDate = orderDate;

this.address = address;

this.deliveryDate = deliveryDate;

this.deliveryStatus = deliveryStatus;

}

public void placeOrder() {

Order newOrder = new Order(orderID, sellerID, customerID, products, totalAmount,orderDate, address, deliveryDate, deliveryStatus);

customer.addOrder(newOrder);

seller.addOrder(newOrder);

System.out.println("Order placed");

}

}

//Payment

class Payment {

private int paymentID;

private int orderID;

private boolean paid;

private double total;

private String details;

public Payment(int paymentID, int orderID, double total) {

this.paymentID = paymentID;

this.orderID = orderID;

this.total = total;

this.paid = false;

this.details = "";

}

public void sendOTP() {

String otp = generateOTP();

System.out.println("OTP sent: " + otp);

}

public void confirmTransaction(String otp) {

if (validateOTP(otp)) {

paid = true;

System.out.println("Transaction confirmed");

} else {

System.out.println("Invalid OTP");

}

}

public void makeTransaction() {

if (paid) {

System.out.println("Transaction successful");

} else {

System.out.println("Transaction not paid");

}

}

}