**TASK 1: DRAWOUT OBJECT – CLASS PROGRAMMING FROM UML DIAGRAM**

A diagram of a server

Description automatically generated

**SOLUTION:**

**User.java**

**package** pkg1;

**public** **class** User {

**private** **boolean** role;

**private** String loginID;

**private** String password;

**public** **boolean** isRole() {

**return** role;

}

**public** **void** setRole(**boolean** role) {

**this**.role = role;

}

**public** String getLoginID() {

**return** loginID;

}

**public** **void** setLoginID(String loginID) {

**this**.loginID = loginID;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** **void** verifyUser() {

System.***out***.println("The user is verified successfully!!!");

}

}

**Customer.java**

**package** pkg1;

**public** **class** Customer **extends** User{

**private** String id;

**private** String address;

**private** String phone;

**private** String Email;

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** String getPhone() {

**return** phone;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** String getEmail() {

**return** Email;

}

**public** **void** setEmail(String email) {

Email = email;

}

**public** **void** login() {

System.***out***.println("Logged in successfully!");

}

**public** **void** register() {

System.***out***.println("User Registered Successfully!");

}

**public** **void** updateProfile() {

System.***out***.println("Profile updation link opened!");

}

}

**Shopping\_Cart.java**

**package** pkg1;

**import** java.util.Date;

**public** **class** Shopping\_Cart **extends** Customer{

**public** **void** c(){

Customer c = **new** Customer(); //composition

c.login();

}

**private** Date created;

**public** Date getCreated() {

**return** created;

}

**public** **void** setCreated(Date created) {

**this**.created = created;

}

**public** **void** addCartItem() {

System.***out***.println("Item added successfully!");

}

**public** **void** checkOut() {

System.***out***.println("Let's checkout!");

}

**public** **void** viewCartDetails() {

System.***out***.println("Let's view the cart details!");

}

**public** **void** updateQuantity() {

System.***out***.println("Let's update the quantity!");

}

}

**Seller.java**

**package** pkg1;

**public** **class** Seller **extends** User{

**private** String id;

**private** String address;

**private** String phone;

**private** String Email;

**private** **int** sellerRating;

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** String getPhone() {

**return** phone;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** String getEmail() {

**return** Email;

}

**public** **void** setEmail(String email) {

Email = email;

}

**public** **int** getSellerRating() {

**return** sellerRating;

}

**public** **void** setSellerRating(**int** sellerRating) {

**this**.sellerRating = sellerRating;

}

**public** **void** login() {

System.***out***.println("Logged in successfully!");

}

**public** **void** register() {

System.***out***.println("User registered successfully!");

}

**public** **void** updateProfile() {

System.***out***.println("Profile updated successfully!");

}

}

**Product.java**

**package** pkg1;

**import** java.util.Date;

**public** **class** Product **extends** Seller{

**public** **void** l(){

Seller l= **new** Seller(); //composition

l.login();

}

**private** String productID;

**private** String ProductName;

**private** **int** productCost;

**private** String sellerId;

**private** Date postedDate;

**public** String getProductID() {

**return** productID;

}

**public** **void** setProductID(String productID) {

**this**.productID = productID;

}

**public** String getProductName() {

**return** ProductName;

}

**public** **void** setProductName(String productName) {

ProductName = productName;

}

**public** **int** getProductCost() {

**return** productCost;

}

**public** **void** setProductCost(**int** productCost) {

**this**.productCost = productCost;

}

**public** String getSellerId() {

**return** sellerId;

}

**public** **void** setSellerId(String sellerId) {

**this**.sellerId = sellerId;

}

**public** Date getPostedDate() {

**return** postedDate;

}

**public** **void** setPostedDate(Date postedDate) {

**this**.postedDate = postedDate;

}

**public** **void** addToart() {

System.***out***.println("Added to cart successfully!");

}

**public** **void** sellProduct() {

System.***out***.println("The product is being sold!");

}

**public** **void** getProductDetails() {

System.***out***.println("Product details fetched successfully!");

}

**public** **void** buyProduct() {

System.***out***.println("Product bought successfully!");

}

}

**Reviews.java**

**package** pkg1;

**public** **class** Reviews **extends** Customer {

**public** **void** u(){

Customer u = **new** Customer(); //composition

u.updateProfile();

}

**private** String Reviewid;

**private** String customerID;

**private** String reviewContent;

**private** **int** rating;

**private** String parentID;

**private** String productID;

**public** String getReviewid() {

**return** Reviewid;

}

**public** **void** setReviewid(String reviewid) {

Reviewid = reviewid;

}

**public** String getCustomerID() {

**return** customerID;

}

**public** **void** setCustomerID(String customerID) {

**this**.customerID = customerID;

}

**public** String getReviewContent() {

**return** reviewContent;

}

**public** **void** setReviewContent(String reviewContent) {

**this**.reviewContent = reviewContent;

}

**public** **int** getRating() {

**return** rating;

}

**public** **void** setRating(**int** rating) {

**this**.rating = rating;

}

**public** String getParentID() {

**return** parentID;

}

**public** **void** setParentID(String parentID) {

**this**.parentID = parentID;

}

**public** String getProductID() {

**return** productID;

}

**public** **void** setProductID(String productID) {

**this**.productID = productID;

}

**public** **void** addReview() {

System.***out***.println("Row added successfully!");

}

**public** **void** deleteReview() {

System.***out***.println("Review deleted successfully!");

}

**public** **void** editReview() {

System.***out***.println("Row edited successfully!");

}

}

**Payment.java**

**package** pkg1;

**public** **class** Payment **extends** Orders {

**public** **void** u(){

Customer u = **new** Customer(); //composition

u.updateProfile();

}

**private** String id;

**private** String OrderID;

**private** **boolean** paid;

**private** **int** total;

**private** String details;

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getOrderID() {

**return** OrderID;

}

**public** **void** setOrderID(String orderID) {

OrderID = orderID;

}

**public** **boolean** isPaid() {

**return** paid;

}

**public** **void** setPaid(**boolean** paid) {

**this**.paid = paid;

}

**public** **int** getTotal() {

**return** total;

}

**public** **void** setTotal(**int** total) {

**this**.total = total;

}

**public** String getDetails() {

**return** details;

}

**public** **void** setDetails(String details) {

**this**.details = details;

}

**public** **void** sendOTP() {

System.***out***.println("OTP sent successfully!");

}

**public** **void** confirmTransaction() {

System.***out***.println("Transaction Confirmed!");

}

**public** **void** getPaymentDetails() {

System.***out***.println("Payment Deatils Fetched Successfully!");

}

**public** **void** makeTransaction() {

System.***out***.println("Transaction done successfully!");

}

}

**Oders.java**

**package** pkg1;

**import** java.util.Date;

**public** **class** Orders **extends** Customer {

**public** **void** p(){

Payment p = **new** Payment(); //composition

p.placeOrder();

}

**private** String id;

**private** String sellerID;

**private** String customerID;

**private** String productID;

**private** String totalAmount;

**private** Date orderDate;

**private** String address;

**private** Date deliveredDate;

**private** String deliveryStatus;

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getSellerID() {

**return** sellerID;

}

**public** **void** setSellerID(String sellerID) {

**this**.sellerID = sellerID;

}

**public** String getCustomerID() {

**return** customerID;

}

**public** **void** setCustomerID(String customerID) {

**this**.customerID = customerID;

}

**public** String getProductID() {

**return** productID;

}

**public** **void** setProductID(String productID) {

**this**.productID = productID;

}

**public** String getTotalAmount() {

**return** totalAmount;

}

**public** **void** setTotalAmount(String totalAmount) {

**this**.totalAmount = totalAmount;

}

**public** Date getOrderDate() {

**return** orderDate;

}

**public** **void** setOrderDate(Date orderDate) {

**this**.orderDate = orderDate;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** Date getDeliveredDate() {

**return** deliveredDate;

}

**public** **void** setDeliveredDate(Date deliveredDate) {

**this**.deliveredDate = deliveredDate;

}

**public** String getDeliveryStatus() {

**return** deliveryStatus;

}

**public** **void** setDeliveryStatus(String deliveryStatus) {

**this**.deliveryStatus = deliveryStatus;

}

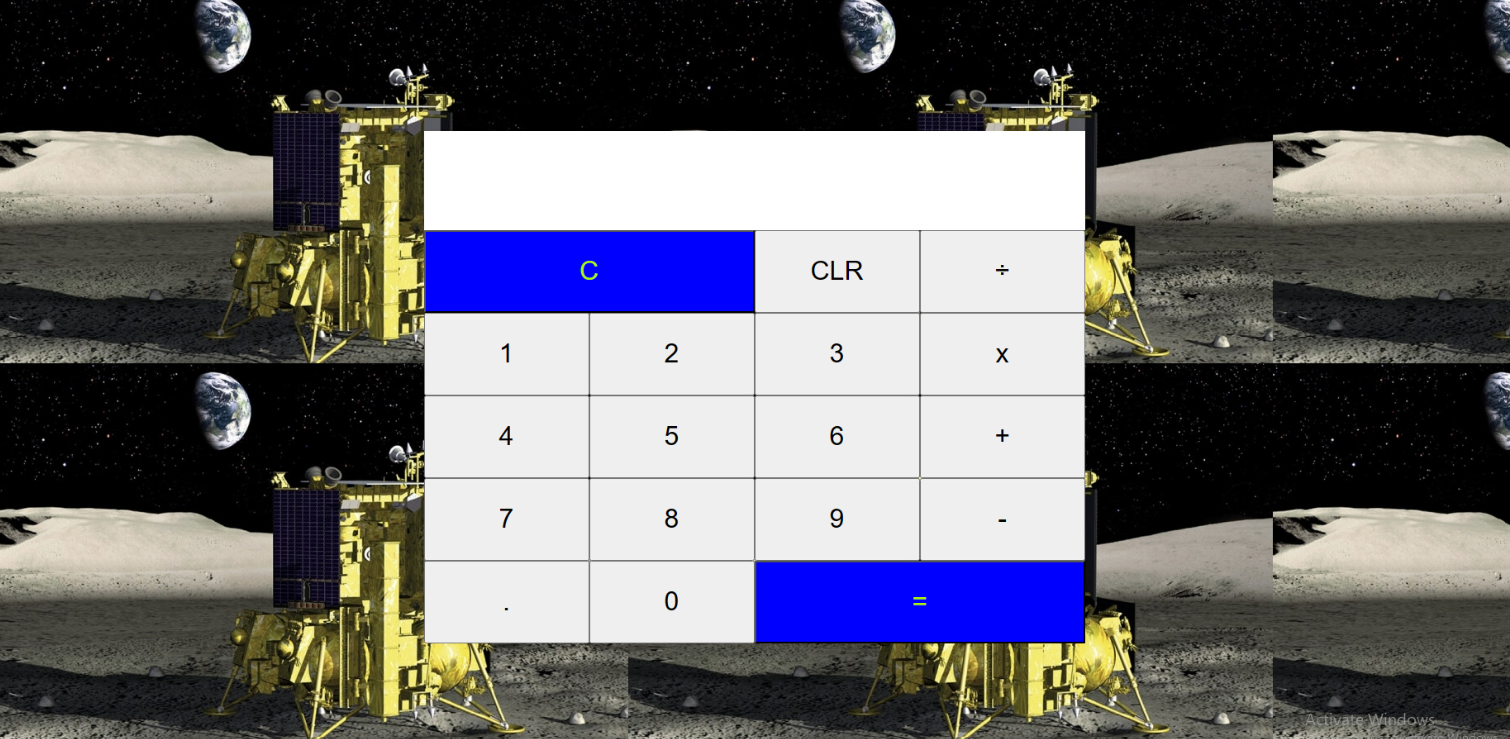
**public** **void** placeOrder() {

System.***out***.println("Order placed successfully!");

}

}

**TASK 2: CREATE A CALCULATOR AS PER THE FOLLOWING DIAGRAM**

****

**SOLUTION:**

**Index.html**

<!DOCTYPE html>

<html>

<head>

<title>Calculator</title>

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

<script src="script.js" defer></script>

</head>

<body>

<div class="calculator-grid">

<div class="output">

<div data-previous-operand class="previous-operand"></div>

<div data-current-operand class="current-operand"></div>

</div>

<button data-all-clear class="span-two">C</button>

<button data-delete>CLR</button>

<button data-operation>÷</button>

<button data-number>1</button>

<button data-number>2</button>

<button data-number>3</button>

<button data-operation>x</button>

<button data-number>4</button>

<button data-number>5</button>

<button data-number>6</button>

<button data-operation>+</button>

<button data-number>7</button>

<button data-number>8</button>

<button data-number>9</button>

<button data-operation>-</button>

<button data-number>.</button>

<button data-number>0</button>

<button data-equals class="span-two">=</button>

</div>

</body>

</html>

**style.css**

\*, \*::before, \*::after {

    box-sizing: border-box;

    font-family: Gotham Rounded, sans-serif;

    font-weight: normal;

  }

  body {

    margin: 0;

    padding: 0;

    background-image: url("Isro.jpg");

    background-color: antiquewhite;

    }

    .calculator-grid {

      display: grid;

      justify-content: center;

      align-content: center;

      min-height: 100vh;

      grid-template-columns: repeat(4, 200px);

      grid-template-rows: minmax(120px, auto) repeat(5, 100px);

      }

      .calculator-grid > button {

        cursor: pointer;

        font-size: 2rem;

        border: 1px, solid #FFFFFF;

        outline: none;

      }

        .calculator-grid > button:hover {

          background-color: #a9a9a9;

        }

        .span-two {

          grid-column: span 2;

          color: #adf802;

          background-color: blue;

        }

        .output{

          grid-column: 1 / -1;

          background-color: white;

          display: flex;

          align-items: flex-end;

          justify-content: space-around;

          flex-direction: column;

          padding: 10px;

          word-wrap: break-word;

          word-break: break-all;

        }

        .output .previous-operand{

          color: black);

          font-size: 1.5rem;

        }

        .output .current-operand{

          color: black;

          font-size: 2.5rem;

        }

**script.js**

class Calculator {

    constructor(previousOperandTextElement, currentOperandTextElement) {

      this.previousOperandTextElement = previousOperandTextElement

      this.currentOperandTextElement = currentOperandTextElement

      this.clear()

    }

    clear() {

      this.currentOperand = ''

      this.previousOperand = ''

      this.operation = undefined

    }

    delete() {

      this.currentOperand = this.currentOperand.toString().slice(0, -1)

    }

    appendNumber(number) {

      if (number === '.' && this.currentOperand.includes('.')) return

      this.currentOperand = this.currentOperand.toString() + number.toString()

    }

    chooseOperation(operation) {

      if (this.currentOperand === '') return

      if (this.previousOperand !== '') {

        this.compute()

      }

      this.operation = operation

      this.previousOperand = this.currentOperand

      this.currentOperand = ''

    }

    compute() {

      let computation

      const prev = parseFloat(this.previousOperand)

      const current = parseFloat(this.currentOperand)

      if (isNaN(prev) || isNaN(current)) return

      switch (this.operation) {

        case '+':

          computation = prev + current

          break

        case '-':

          computation = prev - current

          break

        case '\*':

          computation = prev \* current

          break

        case '÷':

          computation = prev / current

          break

        default:

          return

      }

      this.currentOperand = computation

      this.operation = undefined

      this.previousOperand = ''

    }

    getDisplayNumber(number) {

      const stringNumber = number.toString()

      const integerDigits = parseFloat(stringNumber.split('.')[0])

      const decimalDigits = stringNumber.split('.')[1]

      let integerDisplay

      if (isNaN(integerDigits)) {

        integerDisplay = ''

      } else {

        integerDisplay = integerDigits.toLocaleString('en', { maximumFractionDigits: 0 })

      }

      if (decimalDigits != null) {

        return `${integerDisplay}.${decimalDigits}`

      } else {

        return integerDisplay

      }

    }

    updateDisplay() {

      this.currentOperandTextElement.innerText =

        this.getDisplayNumber(this.currentOperand)

      if (this.operation != null) {

        this.previousOperandTextElement.innerText =

          `${this.getDisplayNumber(this.previousOperand)} ${this.operation}`

      } else {

        this.previousOperandTextElement.innerText = ''

      }

    }

  }

  const numberButtons = document.querySelectorAll('[data-number]')

  const operationButtons = document.querySelectorAll('[data-operation]')

  const equalsButton = document.querySelector('[data-equals]')

  const deleteButton = document.querySelector('[data-delete]')

  const allClearButton = document.querySelector('[data-all-clear]')

  const previousOperandTextElement = document.querySelector('[data-previous-operand]')

  const currentOperandTextElement = document.querySelector('[data-current-operand]')

  const calculator = new Calculator(previousOperandTextElement, currentOperandTextElement)

  numberButtons.forEach(button => {

    button.addEventListener('click', () => {

      calculator.appendNumber(button.innerText)

      calculator.updateDisplay()

    })

  })

  operationButtons.forEach(button => {

    button.addEventListener('click', () => {

      calculator.chooseOperation(button.innerText)

      calculator.updateDisplay()

    })

  })

  equalsButton.addEventListener('click', button => {

    calculator.compute()

    calculator.updateDisplay()

  })

  allClearButton.addEventListener('click', button => {

    calculator.clear()

    calculator.updateDisplay()

  })

  deleteButton.addEventListener('click', button => {

    calculator.delete()

    calculator.updateDisplay()

  })

  //events are shown as below

  document.addEventListener('keydown', function (event) {

    let patternForNumbers = /[0-9]/g;

    let patternForOperators = /[+\-\*\/]/g

    if (event.key.match(patternForNumbers)) {

      event.preventDefault();

      calculator.appendNumber(event.key)

      calculator.updateDisplay()

    }

    if (event.key === '.') {

      event.preventDefault();

      calculator.appendNumber(event.key)

      calculator.updateDisplay()

    }

    if (event.key.match(patternForOperators)) {

      event.preventDefault();

      calculator.chooseOperation(event.key)

      calculator.updateDisplay()

    }

    if (event.key === 'Enter' || event.key === '=') {

      event.preventDefault();

      calculator.compute()

      calculator.updateDisplay()

    }

    if (event.key === "Backspace") {

      event.preventDefault();

      calculator.delete()

      calculator.updateDisplay()

    }

    if (event.key == 'Delete') {

      event.preventDefault();

      calculator.clear()

      calculator.updateDisplay()

    }

  });

**Developed By:**

**Sagar Singha Roy**

PwC | Associate | Advisory

Mobile: +91- 6009350230

Email: [sagar.singha.roy@pwc.com](mailto:sagar.singha.roy@pwc.com)

PricewaterhouseCoopers Private Limited  
56 & 57, Block DN, Sector V, Salt Lake, Kolkata – 700091

PS: Significant amount of research work has been done while solving this exercise from the opensource websites, thus it might seem some logical similarity with some other codes. Please beg your pardon for that.