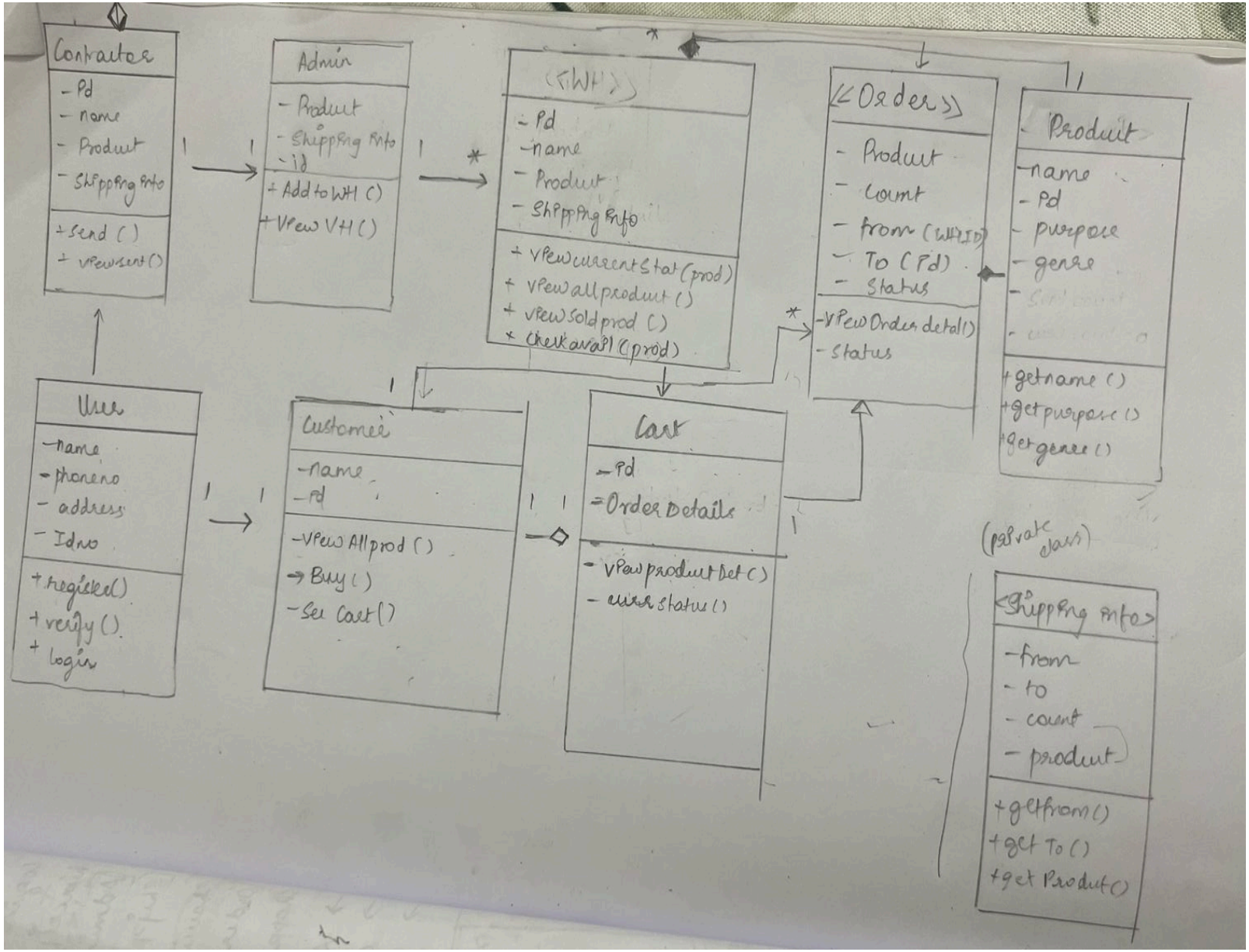
CLASS DIAGRAM



CODE

```
import java.util.*;

class User {
    private String name;
    private String id;
    private String password;
    int count;
    private String Address;
    private String Adhaar_ID;
    private Long contactNo;
    private boolean isVerified;

    public User(String name, String Address, String password, String
Adhaar_ID, Long contactNo, boolean isVerified) {
        this.name = name;
        this.id = name + "_" + count++;
        this.password = password;
        this.Address = Address;
        this.Adhaar_ID = Adhaar_ID;
        this.contactNo = contactNo;
        this.isVerified = isVerified;
    }

    public String getName() { return name; }
    public String getAddress() { return Address; }
    public String getAdhaar_ID() { return Adhaar_ID; }
    public Long getContactNo() { return contactNo; }
    public boolean isVerified() { return isVerified; }
    public String getID() { return id; }
    public String getPassword() { return password; }
    public void setVerified() { this.isVerified = true; }
}

class Customer{

    ArrayList<Product> cart = new ArrayList<>();
    void viewProducts(Warehouse warehouse) {
        warehouse.displayProducts();
```

```
    }
    void buyProduct(Warehouse warehouse, int productIndex) {
        if (productIndex >= 1 && productIndex <=
warehouse.productDetails.size()) {
            Product product = warehouse.productDetails.get(productIndex -
1);
            System.out.println("Successfully purchased: " +
product.getProductName());
        } else {
            System.out.println("Invalid product selection.");
        }
    }

    public void addToCart(String productName) {
        for (Product product : productDetails.keySet()) {
            if (product.getProductName().equalsIgnoreCase(productName)) {
                cart.add(product);
                System.out.println(product.getProductName() + " added to
cart.");
                return;
            }
        }
        System.out.println("Product not found.");
    }

    public void viewCart() {
        if (cart.isEmpty()) {
            System.out.println("Your cart is empty.");
            return;
        }
        for (Product product : cart) {
            System.out.println("----------------------------");
            System.out.println("Product Name: " +
product.getProductName());
            System.out.println("Price: $" + product.getProductCost());
            System.out.println("Description: " +
product.getProductGenre());
            System.out.println("Category: " + product.getProductName());
            System.out.println("----------------------------");
        }
```

```
    }

}

class Administrator {
    private String name;
    private String password;
    private String AdminID;
    Warehouse warehouse ;
    private HashMap<String, Product> receivedProducts = new HashMap<>();
    private List<String> wareHouseID = new ArrayList<>();
    public Administrator(String name, String AdminID, String password,
Warehouse warehouse) {
        this.AdminID = AdminID;
        this.name = name;
        this.warehouse =  warehouse ;
        this.password = password;
    }
    void addProductToWarehouse(Product product, int quantity) {
        warehouse.addProduct(product, quantity);
        System.out.println(product.getProductName() + " added to warehouse
with quantity: " + quantity);
    }
    public void receiveProduct(Product product) {
        receivedProducts.put(product.getProductID(), product);
    }
    public void updateWareHouse(){
        for(String str : receivedProducts.keySet()){

        }
    }
    void removeProductFromWarehouse(String productId) {
        warehouse.removeProduct(productId);
    }

    void updateInventoryInWarehouse(String productId, int quantity) {
        warehouse.updateInventory(productId, quantity);
    }

    void searchProductInWarehouse(String keyword) {
```

```
        warehouse.searchProduct(keyword);
    }

    public String getID() { return AdminID; }
    public String getPassword() { return password; }
    public String getName() { return name; }
}

class Contractor {
    String ID;
    String name;
    String password;
    List<Ship_Product> sent_Products = new ArrayList<>();
    Warehouse warehouse  ;
    public Contractor(String name, String id, String password) {
        this.name = name;
        this.ID = id;
        this.password = password;
    }
    Contractor(Warehouse warehouse){
        this.warehouse =  warehouse ;
    }
    public void sendProducts(Ship_Product product_details) {
        String adminID = product_details.getTo();
        boolean addedToAdmin = false;

        for (Administrator admin : AccountManager.Administrators) {
            if (admin.getID().equals(adminID)) {
                admin.receiveProduct(product_details.getProduct());
                warehouse.addProduct(product_details.getProduct(), 10);
                sent_Products.add(product_details);
                System.out.println("Product successfully added to Admin
and Warehouse.");
                addedToAdmin = true;
                break;
            }
        }

        if (!addedToAdmin) {
```

```
            System.out.println("Administrator ID not found. Product not
sent.");
        }
    }
    void shipProductFromWarehouse(String productId, int quantity) {
        warehouse.shipProduct(productId, quantity);
    }
}

class AccountManager {
    private User user;
    private List<User> registeredUsers = new ArrayList<>();
    public static List<Contractor> Contractors = new ArrayList<>();
    public static List<Administrator> Administrators = new ArrayList<>();
    private List<String> loggedInUsers = new ArrayList<>();

    public AccountManager() {
        Contractor contractor1 = new Contractor("Contractor1",
"contractor_1", "12345");
        Contractor contractor2 = new Contractor("Contractor2",
"contractor_2", "12345");
        Contractors.add(contractor1);
        Contractors.add(contractor2);

        Warehouse warehouse = new Warehouse();
        Administrator administrator1 = new Administrator("Admin1",
"admin1", "12345", warehouse);
        Administrators.add(administrator1);
    }

    public void register(String name, String Address, String password,
String Adhaar_ID, Long contactNo, boolean isVerified) {
        User newUser = new User(name, Address, password, Adhaar_ID,
contactNo, isVerified);
        if (!registeredUsers.contains(newUser))
            System.out.println("Account creation successful! \nYour user
ID is " + newUser.getID());
        else System.out.println("User already exists");
    }
```

```
    public void verifyUser(String id) {
        for (User user : registeredUsers) {
            if (user.getID().equals(id)) {
                if (!user.isVerified()) {
                    user.setVerified();
                    System.out.println("User successfully verified.");
                } else {
                    System.out.println("User already verified.");
                }
                return;
            }
        }
        System.out.println("User ID is invalid");
    }

    public void login(String id, String password) {
        if(id.substring(0,4 ).equals("Admin") ){
            System.out.println("You are Logged In as an ADMIN");
        }
        else if(id.substring(0 , 10).equals("Contractor")) {
        }
        else
        for (User user : registeredUsers) {
            if (user.getID().equals(id) &&
user.getPassword().equals(password)) {
                loggedInUsers.add(id);
                System.out.println("Login successful");
                return;
            }
        }

        System.out.println("Invalid credentials");
    }

    public static List<Administrator> get_Administrator() {
        return Administrators;
    }

}
```

```
class Ship_Product {
    private String ContractorID;
    private String AdministratorID;
    private int count;
    private Product product;

    public Ship_Product(String ContractorID, String AdministratorID,
Product product, int count) {
        this.ContractorID = ContractorID;
        this.AdministratorID = AdministratorID;
        this.product = product;
        this.count = count;
    }

    public int getCount() { return count; }
    public String getTo() { return AdministratorID; }
    public String getFrom() { return ContractorID; }
    public Product getProduct() { return product; }
}

class Product {
    private String id;
    private String name;
    private String purpose;
    private String genre;
    private double cost;
    private int tax;

    public Product(String id, String name, String purpose, String genre,
double mvp) {
        this.id = id;
        this.name = name;
        this.purpose = purpose;
        this.genre = genre;
        this.cost = mvp + (mvp * tax) / 100;
    }

    public String getProductID() { return id; }
    public String getProductName() { return name; }
    public String getProductGenre() { return genre; }
```

```
    public double getProductCost() { return cost; }
}

class Warehouse {
    Map<String, Product> productDetails = new HashMap<>();
    Map<String, Integer> inventory = new HashMap<>(); //currquantity
    // List<Product> products

    void addProduct(Product product, int quantity) {
        productDetails.put(product.getProductID(), product);
        inventory.put(product.getProductID(),
inventory.getOrDefault(product.getProductID(), 0) + quantity);
    }
    void displayProducts() {
        if (productDetails.isEmpty()) {
            System.out.println("No products available.");
        } else {
            for (int i = 0; i < productDetails.size(); i++) {
                System.out.println((i + 1) + ". " +
productDetails.get(i).getProductName() + " - $" +
productDetails.get(i).getProductCost());
            }
        }
    }
    void removeProduct(String productId) {
        if (inventory.containsKey(productId)) {
            inventory.remove(productId);
            productDetails.remove(productId);
            System.out.println("Product with ID " + productId + " removed
from warehouse.");
        } else {
            System.out.println("Product with ID " + productId + " not
found.");
        }
    }

    void updateInventory(String productId, int quantity) {
        if (inventory.containsKey(productId)) {
            inventory.put(productId, quantity);
```

```
            System.out.println("Inventory updated for product ID " +
productId);
        } else {
            System.out.println("Product with ID " + productId + " not
found.");
        }
    }

    void shipProduct(String productId, int quantity) {
        if (inventory.containsKey(productId) && inventory.get(productId)
>= quantity) {
            inventory.put(productId, inventory.get(productId) - quantity);
            System.out.println(quantity + " units of product ID " +
productId + " shipped.");
        } else {
            System.out.println("Insufficient stock or product not
found.");
        }
    }

    void searchProduct(String keyword) {
        boolean found = false;
        for (Product product : productDetails.values()) {
            if
(product.getProductName().toLowerCase().contains(keyword.toLowerCase()) ||

product.getProductGenre().toLowerCase().contains(keyword.toLowerCase())) {
                System.out.println(product + " | Quantity: " +
inventory.get(product.getProductID()));
                found = true;
            }
        }
        if (!found) {
            System.out.println("No products found matching the keyword: "
+ keyword);
        }
    }

    void displayInventory() {
        productDetails.values().stream()
```

```
.sorted(Comparator.comparing(Product::getProductCost).thenComparing(Produc
t::getProductName))
            .forEach(product -> System.out.println(product + " | Quantity:
" + inventory.get(product.getProductID())));
    }
}


public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Warehouse warehouse = new Warehouse();
        Administrator admin = new Administrator("" , "" , "", warehouse);
        Contractor contractor = new Contractor(warehouse);
        AccountManager accountManager = new AccountManager();
        Customer customer = new Customer();

        admin.addProductToWarehouse(new Product("P001", "Laptop",
"Electronics", "Tech", 50000), 10);
        admin.addProductToWarehouse(new Product("P002", "Phone",
"Electronics", "Tech", 30000), 20);

        System.out.println("\nCurrent Inventory:");
        warehouse.displayInventory();

        admin.removeProductFromWarehouse("P001");
        System.out.println("\nUpdated Inventory:");
        warehouse.displayInventory();

        admin.updateInventoryInWarehouse("P002", 15);
        System.out.println("\nInventory After Update:");
        warehouse.displayInventory();

        System.out.println("\nShipping 5 units of Phone:");
        contractor.shipProductFromWarehouse("P002", 5);
        warehouse.displayInventory();

        System.out.println("\nSearch Results for 'Phone':");
        admin.searchProductInWarehouse("Phone");
```

```
        accountManager.register("Alice", "123 Main St", "password123",
"123456789012", 1234567890L, false);
        accountManager.verifyUser("Alice_0");

=        accountManager.login("Alice_0", "password123");

        customer.viewProducts(warehouse);
        customer.addToCart("Phone");
        customer.viewCart();
        customer.buyProduct(warehouse, 1);

    }

}
```

OP

```
--- Admin Operations ---
Laptop added to warehouse with quantity: 10
Phone added to warehouse with quantity: 20
Product{id='P001', name='Laptop', purpose='Electronics', genre='Tech',
cost=50000.0} | Quantity: 10
Product{id='P002', name='Phone', purpose='Electronics', genre='Tech',
cost=30000.0} | Quantity: 20
Product with ID P001 removed from warehouse.
Product{id='P002', name='Phone', purpose='Electronics', genre='Tech',
cost=30000.0} | Quantity: 20
Inventory updated for product ID P002
Product{id='P002', name='Phone', purpose='Electronics', genre='Tech',
cost=30000.0} | Quantity: 15
Product{id='P002', name='Phone', purpose='Electronics', genre='Tech',
cost=30000.0} | Quantity: 15

--- Contractor Operations ---
5 units of product ID P002 shipped.
```

```
Product{id='P002', name='Phone', purpose='Electronics', genre='Tech',
cost=30000.0} | Quantity: 10
Product successfully added to Admin and Warehouse.
Product{id='P002', name='Phone', purpose='Electronics', genre='Tech',
cost=30000.0} | Quantity: 10
Product{id='P003', name='Tablet', purpose='Electronics', genre='Tech',
cost=40000.0} | Quantity: 10

--- User Registration and Verification ---
Account creation successful!
Your user ID is Alice_0
User successfully verified.

--- User Login ---
Login successful

--- Customer Operations ---
1. Product{id='P002', name='Phone', purpose='Electronics', genre='Tech',
cost=30000.0} - $30000.0
2. Product{id='P003', name='Tablet', purpose='Electronics', genre='Tech',
cost=40000.0} - $40000.0
Phone added to cart.
----------------------------
Product Name: Phone
Price: $30000.0
Description: Tech
Category: Phone
----------------------------
Successfully purchased: Phone
Inventory After Customer Purchase
Product{id='P003', name='Tablet', purpose='Electronics', genre='Tech',
cost=40000.0} | Quantity: 10
Product{id='P002', name='Phone', purpose='Electronics', genre='Tech',
cost=30000.0} | Quantity: 9
```