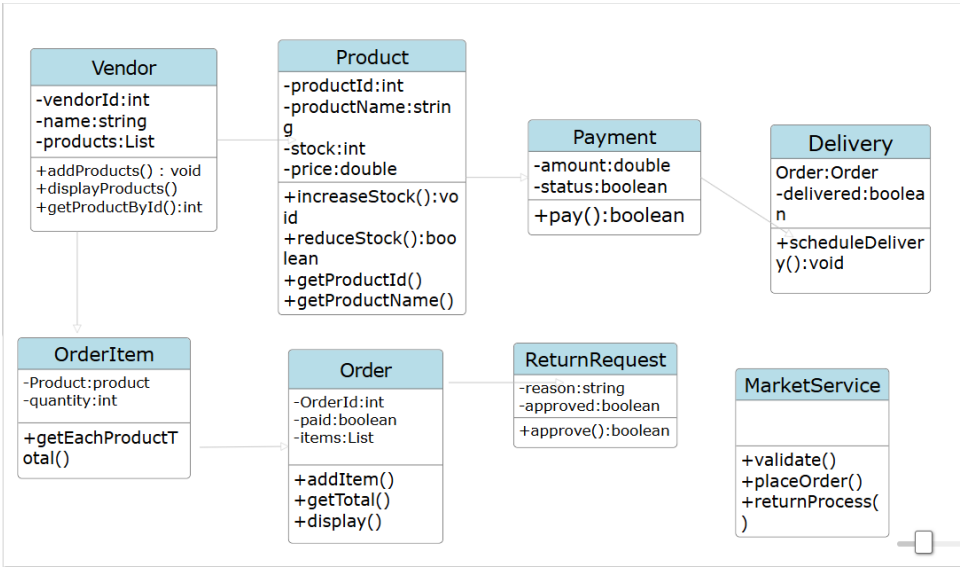
FARMER'S MARKET ORDERS AND PAYMENTS

1.UML DIAGRAM



2.JAVA FULL CODE

Com.kce.market.exception
MarketValidationException.java

```
package com.kce.market.exception;

public class MarketValidationException extends Exception{
	public String toString()
	{
		return "Invalid";
	}
}
```

com.kce.market.service
MarketService.java

```
package com.kce.market.service;
import com.kce.market.model.*;
import com.kce.market.exception.MarketValidationException;

public class MarketService {

public void validate(Product p , int qty) throws MarketValidationException{
	if(p==null || qty < 0)
	throw new MarketValidationException();
}

public void placeOrder(Order order, OrderItem item) {
   order.addItem(item);
   item.getProduct().reduceStock(item.getQuantity());
   System.out.println("Order Placed");
}
```

```
public void returnProcess(Product p,int qty)
{
	p.increaseStock(qty);
	System.out.println("Order returned");
}
```

com.kce.market.model;
Vendor.java

```
package com.kce.market.model;
import java.util.ArrayList;
import java.util.List;

public class Vendor {
	public int vendorId;
	public String name;
	public List<Product> products;

	public Vendor(int vendorId,String name)
	{
		this.vendorId = vendorId;
		this.name=name;
		this.products = new ArrayList<>();

	}

	public void addProducts(Product p)
	{
		products.add(p);
	}

	public void displayProducts()
	{
		for(Product p : products)
		{
		System.out.println(p);
		}
	}
	public Product getProductById(int id) {
    for (Product p : products) {
      if (p.getProductId() == id) {
        return p;
      }
    }
    return null;
  }

}
```

Product.java

```
package com.kce.market.model;
```

```java
public class Product {
	public int productId;
	public String productName;
	public int stock;
	public double price;

	public Product(int productId,String productName, int stock,double price)
	{
		this.productId=productId;
		this.productName = productName;
		this.stock = stock;
		this.price = price;
	}
	public int getProductId() {
		return productId;
		}
	 public String getProductName() {
		  return productName;
		}

	 public int getStock() {
		return stock;
		}

	public double getPrice() {
		return price;
		}

	public boolean reduceStock(int qty)
	{
		if(stock >= qty)
		{
			stock-=qty;
			return true;
		}
		return false;
	}

	public void increaseStock(int qty)
	{
		stock+=qty;
	}
}
```

Order.java

```
package com.kce.market.model;

import java.util.ArrayList;
import java.util.List;

public class Order {
   private int orderId;
   private boolean paid;
   private List<OrderItem> items;
```

```
    public Order(int orderId) {
        this.orderId = orderId;
        this.items = new ArrayList<>();
        this.paid = false;
    }

    public void addItem(OrderItem item) {
        items.add(item);
    }

    public double getTotal() {
        double total = 0;
        for (OrderItem i : items) {
            total += i.getEachProductTotal();
        }
        return total;
    }

    public void display() {
        System.out.println("OrderId" + orderId);
        for (OrderItem i : items) {
         System.out.println(i.getProduct().getProductName() + " x " + i.getQuantity() +   " = " +
i.getEachProductTotal());
        }
        System.out.println("Total cost: " + this.getTotal());
    }

    public boolean isPaid() {
        return paid;
    }

    public void setPaid(boolean paid) {
        this.paid = paid;
    }

    public int getOrderId() {
        return orderId;
    }
}
```

OrderItem.java

```
package com.kce.market.model;


public class OrderItem {
    private Product product;
    private int quantity;

    public OrderItem(Product product,int quantity) {
        this.product = product;
        this.quantity = quantity;
    }
```

```
    public Product getProduct() {
        return product;
    }

    public int getQuantity() {
        return quantity;
    }

    public double getEachProductTotal() {
        return product.getPrice() * quantity;
    }
}
```

Delivery.java

```
package com.kce.market.model;


public class Delivery {
    private Order order;
    private boolean delivered;

    public Delivery(Order order) {
        this.order = order;
        this.delivered = false;
    }
    public void scheduleDelivery() {
        if (order.isPaid()) {
            delivered = true;
            System.out.println( "Order has been delivered.");
        } else {
            System.out.println( "Payment not completed.");
        }
    }

    public boolean isDelivered() {
        return delivered;
    }

    public Order getOrder() {
        return order;
    }
}
```

Payment.java

```
package com.kce.market.model;

public class Payment {
       public double amount;
       public boolean status;

       public Payment(double amount)
       {
          this.amount = amount;
          this.status = false;
       }
```

```
        public boolean pay(double payAmount )
        {
           if(payAmount >= amount)
           {
                  status=true;
                  System.out.println("payment Successful");
                  return true;
           }
           System.out.println("payment is not done");
           return false;

        }
}
```

ReturnRequest.java

```
package com.kce.market.model;

public class ReturnRequest {
    private String reason;
    private boolean approved;

    public ReturnRequest(String reason) {
        this.reason = reason;
        this.approved = false;
    }

    public void approve(Product product, int qty) {
        product.increaseStock(qty);
        approved = true;
        System.out.println("Product Returned Reason: " + reason);
    }

    public boolean isApproved() {
        return approved;
    }

    public String getReason() {
        return reason;
    }
}
```

Com.kce.market.main

Main.java

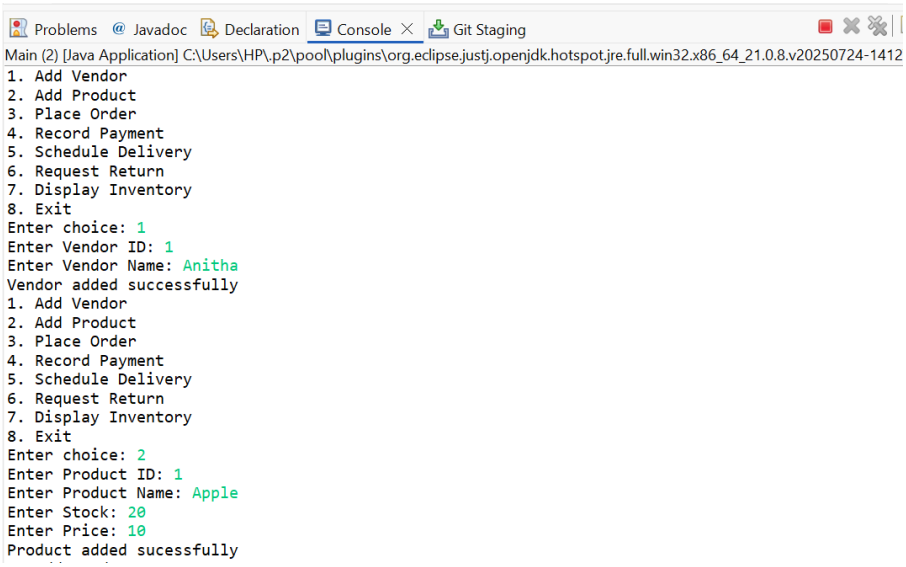
```
package com.kce.market.main;


import com.kce.market.model.*;
import com.kce.market.service.MarketService;
import com.kce.market.exception.MarketValidationException;
import java.util.*;

public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      MarketService service = new MarketService();
```

```
Vendor vendor = null;
Order order = null;
Delivery delivery = null;

boolean exit = false;
while (!exit) {
   System.out.println("1. Add Vendor");
   System.out.println("2. Add Product");
   System.out.println("3. Place Order");
   System.out.println("4. Record Payment");
   System.out.println("5. Schedule Delivery");
   System.out.println("6. Request Return");
   System.out.println("7. Display Inventory");
   System.out.println("8. Exit");
   System.out.print("Enter choice: ");
   int choice = sc.nextInt();
   sc.nextLine();

   switch (choice) {
      case 1:
         System.out.print("Enter Vendor ID: ");
         int vid = sc.nextInt(); sc.nextLine();
         System.out.print("Enter Vendor Name: ");
         String vname = sc.nextLine();
         vendor = new Vendor(vid, vname);
         System.out.println("Vendor added successfully");
         break;

      case 2:
         if (vendor == null) {
               System.out.println("Add vendor");
               break;
               }
         System.out.print("Enter Product ID: ");
         int pid = sc.nextInt(); sc.nextLine();
         System.out.print("Enter Product Name: ");
         String pname = sc.nextLine();
         System.out.print("Enter Stock: ");
         int stock = sc.nextInt();
         System.out.print("Enter Price: ");
         double price = sc.nextDouble();
         Product p = new Product(pid, pname, stock, price);
         vendor.addProducts(p);
         System.out.println("Product added sucessfully");
         break;

      case 3:
         if (vendor == null) {
               System.out.println("Add vendor and product first!");
               break;
               }
```

```
    vendor.displayProducts();
    System.out.print("Enter Product ID to order: ");
    int opid = sc.nextInt();
    Product product = vendor.getProductById(opid);

    System.out.print("Enter quantity: ");
    int qty = sc.nextInt();
    order = new Order(1);
    try {
        service.validate(product, qty);
        OrderItem item = new OrderItem(product, qty);
        service.placeOrder(order, item);
        order.display();
    } catch (MarketValidationException e) {
        System.out.println(e);
    }
    break;

case 4:
    if (order == null) {
            System.out.println("No order placed!");
            break;
            }
    System.out.print("Enter payment amount: ");
    double amt = sc.nextDouble();
    Payment pay = new Payment(order.getTotal());
    pay.pay(amt);
    if (pay.status) order.setPaid(true);
    break;

case 5:
    delivery = new Delivery(order);
    delivery.scheduleDelivery();
    break;

case 6:
    System.out.print("Enter Product ID to return: ");
    int rid = sc.nextInt();
    Product rp = vendor.getProductById(rid);
    if (rp == null) {
    System.out.println("Product not found!");
    break;
    }
    System.out.print("Enter return quantity: ");
    int rqty = sc.nextInt(); sc.nextLine();
    System.out.print("Enter reason: ");
    String reason = sc.nextLine();
    ReturnRequest rr = new ReturnRequest(reason);
    rr.approve(rp, rqty);
    break;
```

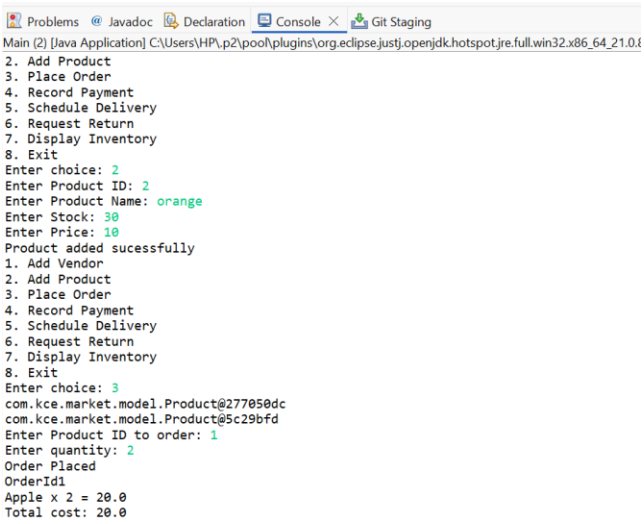
```
            case 7:
                vendor.displayProducts();
                break;

            case 8:
                exit = true;
                System.out.println("Exiting...");
                break;

            default:
                System.out.println("Invalid choice!");
        }
    }
    sc.close();
  }
}
```

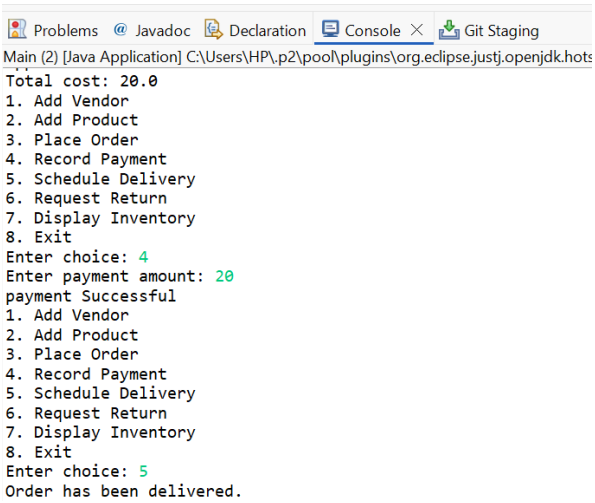
3.OUTPUTS:

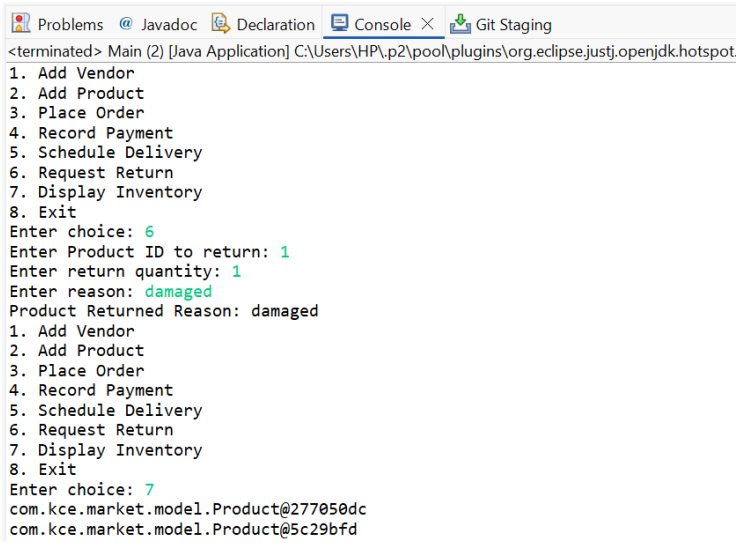


4.Git Repository Link:

https://github.com/Anitharani205/Farmer-s-Market-Orders-and-Payment-console-based-application-


Problems Javadoc Declaration Console Git Staging
Main (2) [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.8
2. Add Product
3. Place Order
4. Record Payment
5. Schedule Delivery
6. Request Return
7. Display Inventory
8. Exit
Enter choice: 2
Enter Product ID: 2
Enter Product Name: orange
Enter Stock: 30
Enter Price: 10
Product added sucessfully
1. Add Vendor
2. Add Product
3. Place Order
4. Record Payment
5. Schedule Delivery
6. Request Return
7. Display Inventory
8. Exit
Enter choice: 3
com.kce.market.model.Product@277050dc
com.kce.market.model.Product@5c29bfd
Enter Product ID to order: 1
Enter quantity: 2
Order Placed
OrderId1
Apple x 2 = 20.0
Total cost: 20.0


Problems Javadoc Declaration Console Git Staging
Main (2) [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hots
Total cost: 20.0
1. Add Vendor
2. Add Product
3. Place Order
4. Record Payment
5. Schedule Delivery
6. Request Return
7. Display Inventory
8. Exit
Enter choice: 4
Enter payment amount: 20
payment Successful
1. Add Vendor
2. Add Product
3. Place Order
4. Record Payment
5. Schedule Delivery
6. Request Return
7. Display Inventory
8. Exit
Enter choice: 5
Order has been delivered.


Problems Javadoc Declaration Console Git Staging
<terminated> Main (2) [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.
1. Add Vendor
2. Add Product
3. Place Order
4. Record Payment
5. Schedule Delivery
6. Request Return
7. Display Inventory
8. Exit
Enter choice: 6
Enter Product ID to return: 1
Enter return quantity: 1
Enter reason: damaged
Product Returned Reason: damaged
1. Add Vendor
2. Add Product
3. Place Order
4. Record Payment
5. Schedule Delivery
6. Request Return
7. Display Inventory
8. Exit
Enter choice: 7
com.kce.market.model.Product@277050dc
com.kce.market.model.Product@5c29bfd

```
Enter quantity: 30
Order Placed
OrderId1
apple x 30 = 360.0
Total cost: 360.0
1. Add Vendor
2. Add Product
3. Place Order
4. Record Payment
5. Schedule Delivery
6. Request Return
7. Display Inventory
8. Exit
Enter choice: 4
Enter payment amount: 200
payment is not done
1. Add Vendor
2. Add Product
3. Place Order
4. Record Payment
5. Schedule Delivery
6. Request Return
7. Display Inventory
8. Exit
```

```
1. Add Vendor
2. Add Product
3. Place Order
4. Record Payment
5. Schedule Delivery
6. Request Return
7. Display Inventory
8. Exit
Enter choice: 1
Enter Vendor ID: anitha
Exception in thread "main" java.util.InputMismatchException
        at java.base/java.util.Scanner.throwFor(Scanner.java:947)
        at java.base/java.util.Scanner.next(Scanner.java:1602)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2267)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2221)
        at FarmerMarketOrders/com.kce.market.main.Main.main(Main.java:33)
```