


## APPLICATION DEVELOPMENT AND DEPLOYMENT ARCHITECTURE

### THEORY DIGITAL ASSESSMENT

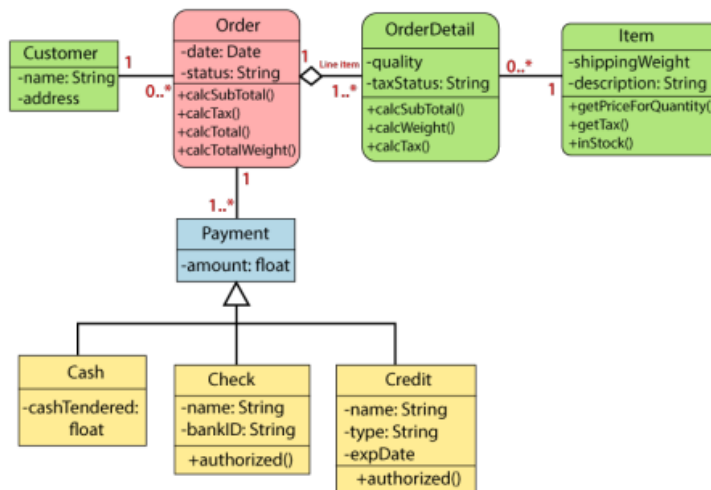
CHARATH H J (22MIC0088)

SREE KRISHNA A K (22MIC0093)

 <b>VIT</b> Vellore Institute of Technology	Digital Assignment	
Course code: CSI3025	Course Title :Advanced Development and Deployment Architecture	Batch : B1
	Course mode: Theory	Batch : B2
Class No:3194/3202	School: SCOPE	Venue: PRP G32
Emp. ID: 20774	Faculty Name: SUDHAKAR. P	Code : B

#### Instructions:

- Read the question carefully and answer to the question.
- Each questions carry marks.
- Submit the answers in the vtop (Before submission, double check document and submit the right document.)
- For any reason, resubmission or reconduct of DA was not allowed.
- Plagirism of any source is merely rejection of document and zero will be awarded. Hence do not ask your friends to submit / copy or do any action towards your assignment. No further consideration will be entertained.
- Your answer should be well explainable according to the question. If any code segment is required, include the same.
- Finally submit your answer as a pdf containing the answers and source code developed (after all questions are answered, add the source codes with appropriate explanation/ headline)



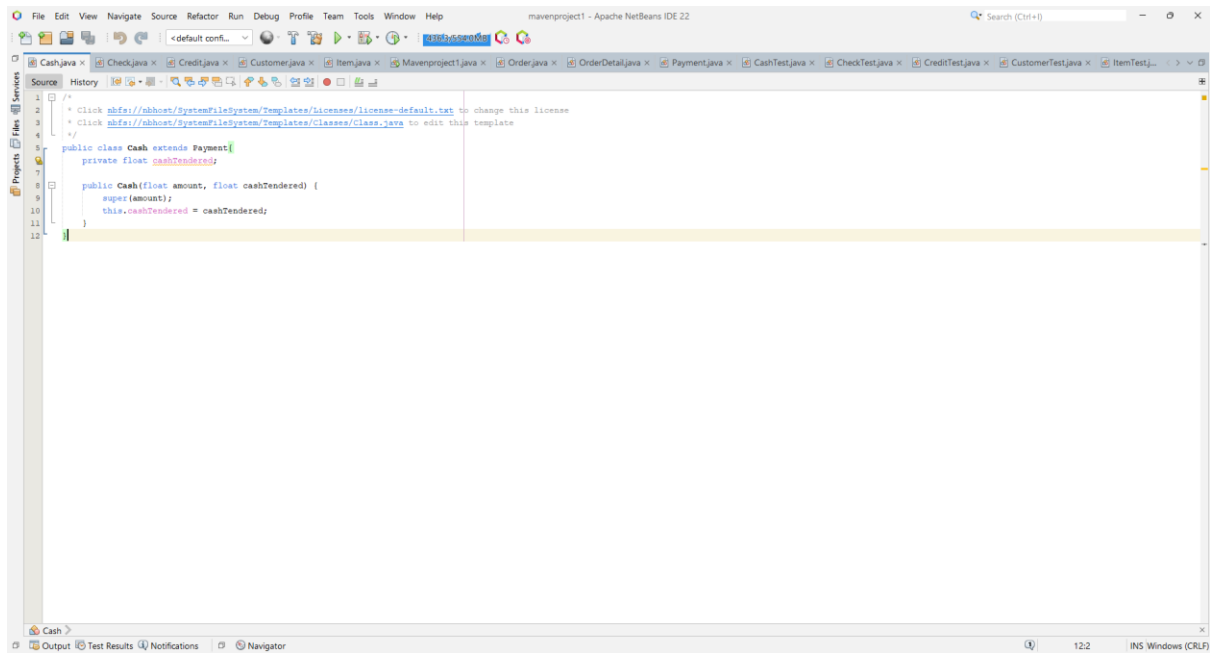
**Question 1:** Develop a prototype code for the Customer management software (refer Figure 1) using Java/C++/Python.[5 Marks]

**Question 2:** Develop a Unit test case for the given figure 1. Atleast 5 test cases must be developed [5 Marks]

**Question 3:** Study any automated testing software and test your code. Demonstrate your answers step-by-step screenshots alongwith necessary testing scripts. [10 Marks]

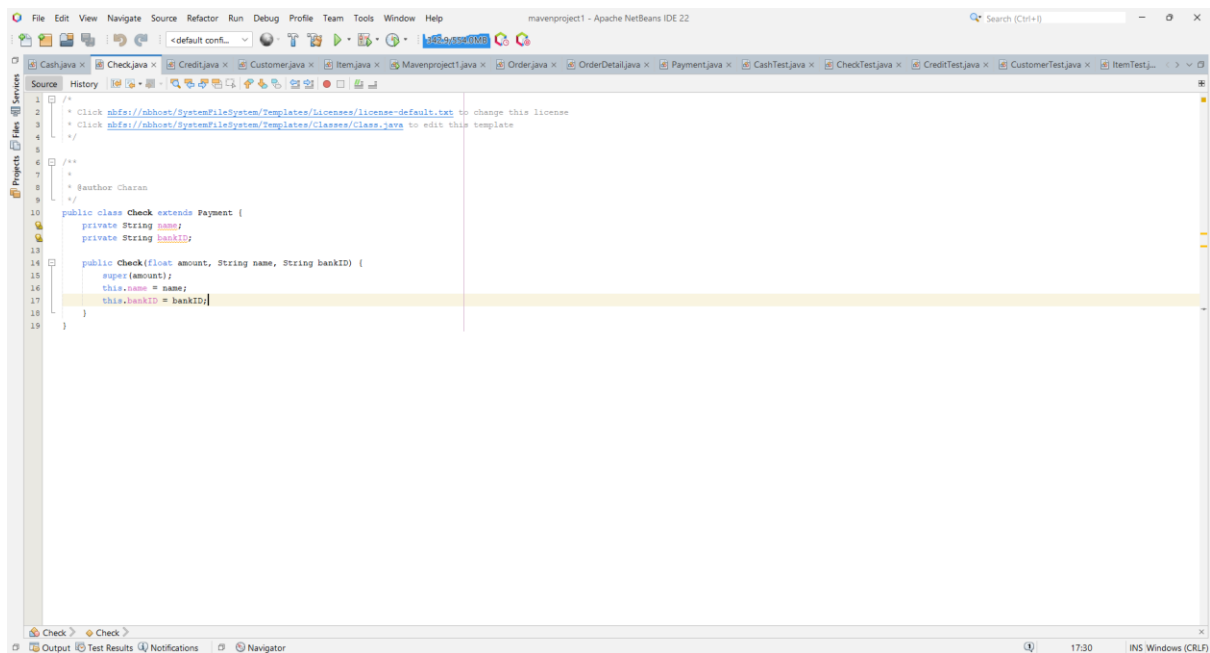
## ✓ SOURCE PACKAGES:

Cash.java:



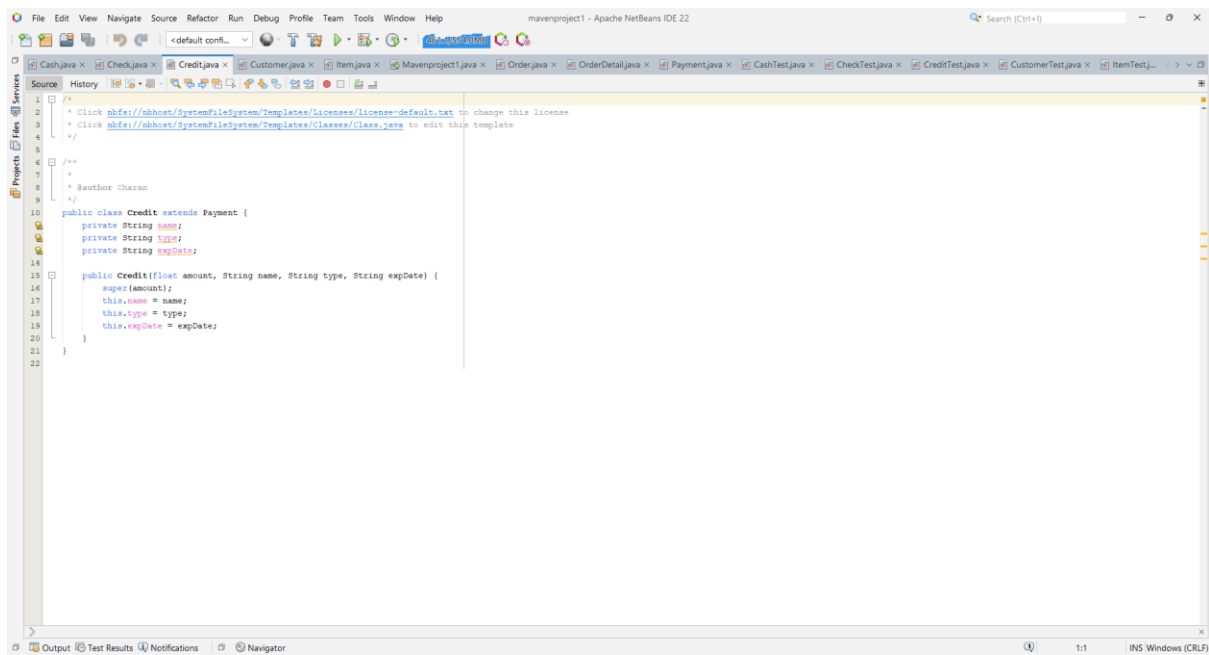
```
1  /*
2  * Click https://www.netbeans.org/javahelp/first-class.html to change this license
3  * Click https://www.netbeans.org/javahelp/first-class.html to edit this template
4  */
5
6  public class Cash extends Payment {
7      private float cashTendered;
8
9      public Cash(float amount, float cashTendered) {
10         super(amount);
11         this.cashTendered = cashTendered;
12     }
13 }
```

Check.java



```
1  /**
2  *
3  * @author Charan
4  */
5
6  public class Check extends Payment {
7      private String name;
8      private String bankID;
9
10     public Check(float amount, String name, String bankID) {
11         super(amount);
12         this.name = name;
13         this.bankID = bankID;
14     }
15 }
```

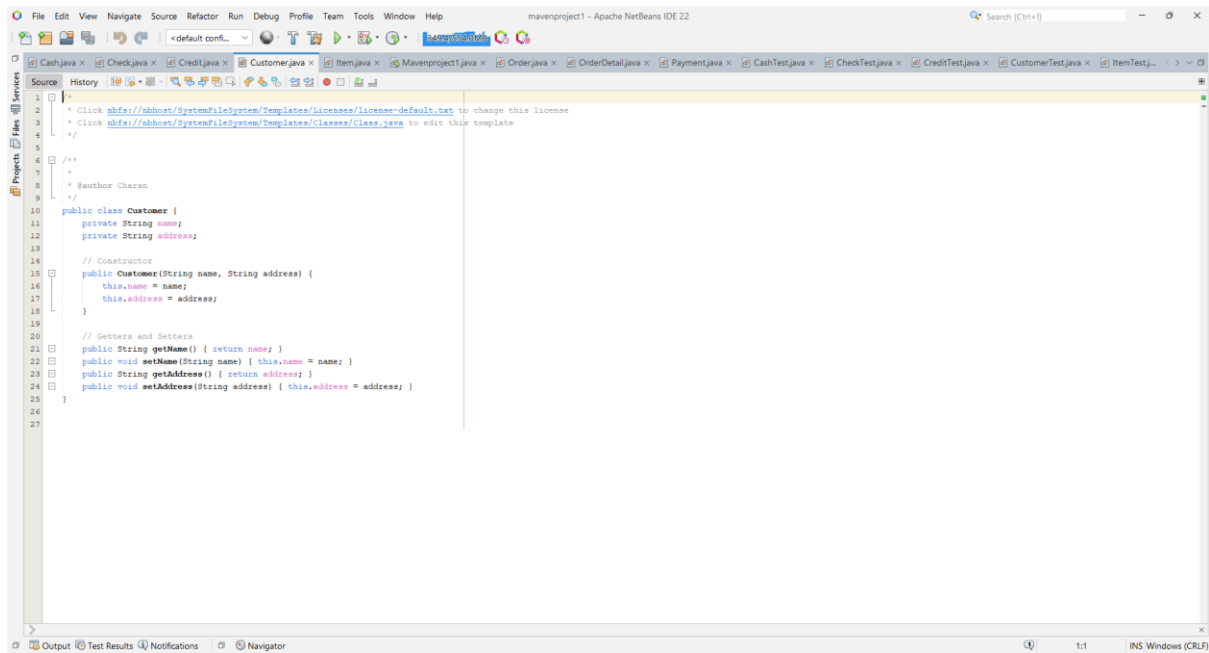
## Credit.java



The screenshot shows the NetBeans IDE with the 'Credit.java' file open. The code defines a 'Credit' class that extends the 'Payment' class. It includes private attributes for 'name', 'type', and 'expDate', and a constructor that initializes these attributes. The IDE interface includes a menu bar, a toolbar, and a project explorer on the left.

```
1  /*
2  * Click http://.netbeans.org/licenses/license-default.txt to change this license
3  * Click http://.netbeans.org/licenses/license-default.txt to edit this template
4  */
5
6  /**
7   *
8   * @author Charan
9   */
10 public class Credit extends Payment {
11     private String name;
12     private String type;
13     private String expDate;
14
15     public Credit(float amount, String name, String type, String expDate) {
16         super(amount);
17         this.name = name;
18         this.type = type;
19         this.expDate = expDate;
20     }
21 }
22
```

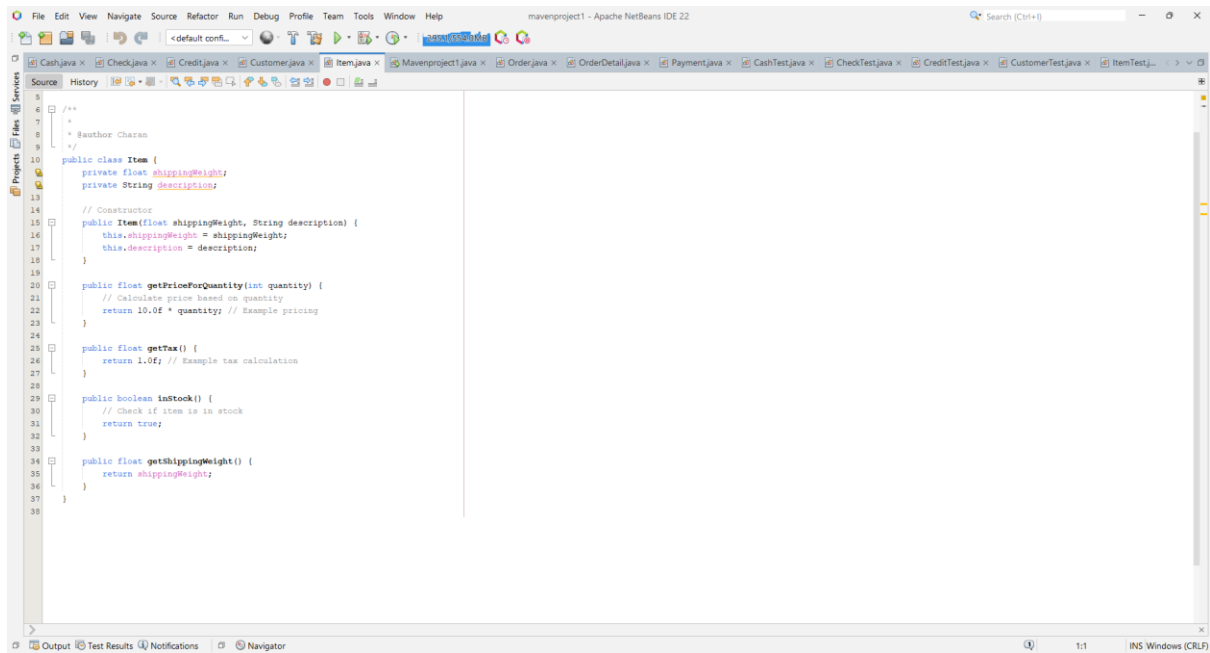
## Customer.java



The screenshot shows the NetBeans IDE with the 'Customer.java' file open. The code defines a 'Customer' class with private attributes for 'name' and 'address'. It includes a constructor and getter/setter methods for these attributes. The IDE interface is consistent with the previous screenshot.

```
1  /*
2  * Click http://.netbeans.org/licenses/license-default.txt to change this license
3  * Click http://.netbeans.org/licenses/license-default.txt to edit this template
4  */
5
6  /**
7   *
8   * @author Charan
9   */
10 public class Customer {
11     private String name;
12     private String address;
13
14     // Constructor
15     public Customer(String name, String address) {
16         this.name = name;
17         this.address = address;
18     }
19
20     // Getters and Setters
21     public String getName() { return name; }
22     public void setName(String name) { this.name = name; }
23     public String getAddress() { return address; }
24     public void setAddress(String address) { this.address = address; }
25 }
26
27
```

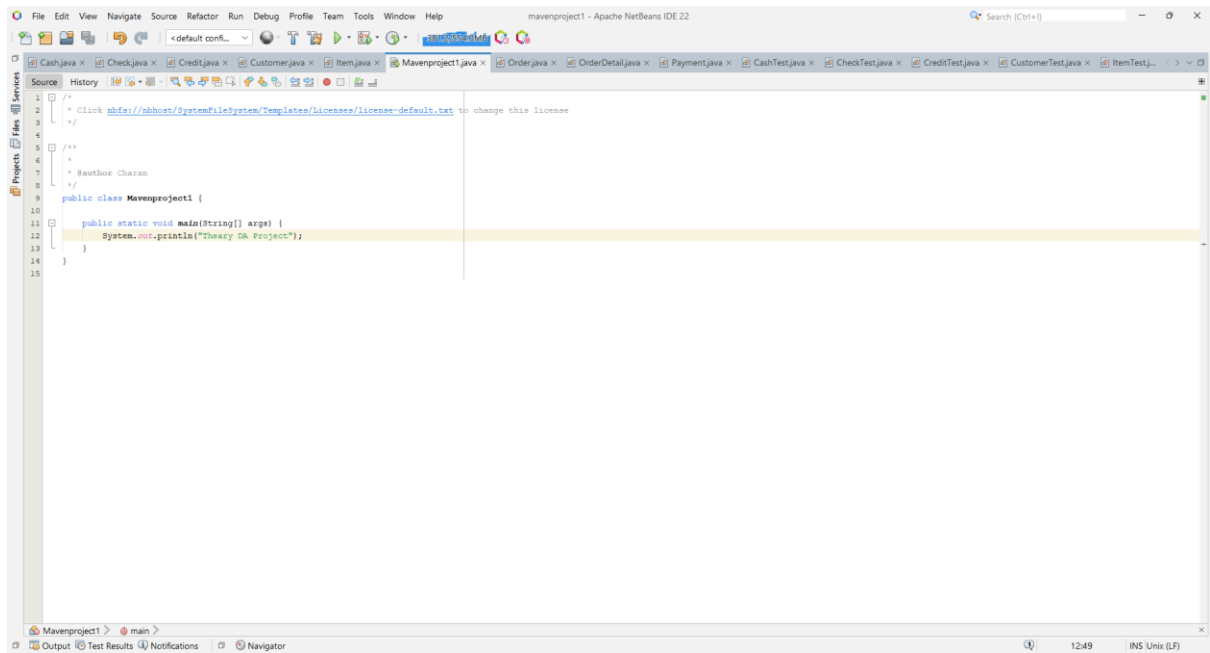
## Item.java



The screenshot shows the NetBeans IDE with the 'Item.java' file open. The code defines a class 'Item' with private attributes 'shippingWeight' (float) and 'description' (String). It includes a constructor, a 'getPriceForQuantity' method, a 'getTax' method, an 'isInStock' method, and a 'getShippingWeight' method. The IDE interface includes a menu bar, a toolbar, a project explorer on the left, and a status bar at the bottom.

```
1  /**  
2   *  
3   * @author Charan  
4   */  
5  
6  public class Item {  
7      private float shippingWeight;  
8      private String description;  
9  }  
10  
11  // Constructor  
12  public Item(float shippingWeight, String description) {  
13      this.shippingWeight = shippingWeight;  
14      this.description = description;  
15  }  
16  
17  public float getPriceForQuantity(int quantity) {  
18      // Calculate price based on quantity  
19      return 10.0f * quantity; // Example pricing  
20  }  
21  
22  public float getTax() {  
23      return 1.0f; // Example tax calculation  
24  }  
25  
26  public boolean isInStock() {  
27      // Check if item is in stock  
28      return true;  
29  }  
30  
31  public float getShippingWeight() {  
32      return shippingWeight;  
33  }  
34  }  
35  
36  
37  
38
```

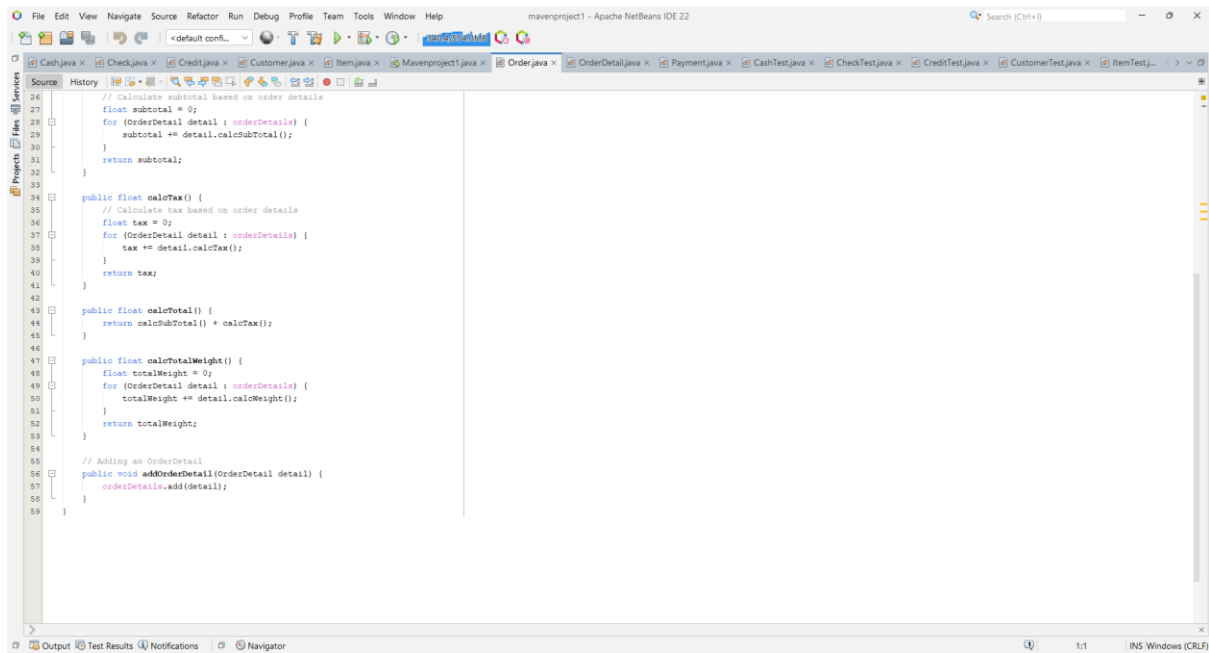
## Mavenproject1.java



The screenshot shows the NetBeans IDE with the 'Mavenproject1.java' file open. The code defines a class 'Mavenproject1' with a 'main' method that prints 'Theory DA Project'. The IDE interface includes a menu bar, a toolbar, a project explorer on the left, and a status bar at the bottom.

```
1  /**  
2   * Click https://github.com/abhishek099/TemplateSystem/TemplateSystem/licenses/license-default.txt to change this license  
3   */  
4  
5  /**  
6   *  
7   * @author Charan  
8   */  
9  
10 public class Mavenproject1 {  
11  
12     public static void main(String[] args) {  
13         System.out.println("Theory DA Project");  
14     }  
15 }  
16  
17  
18  
19
```

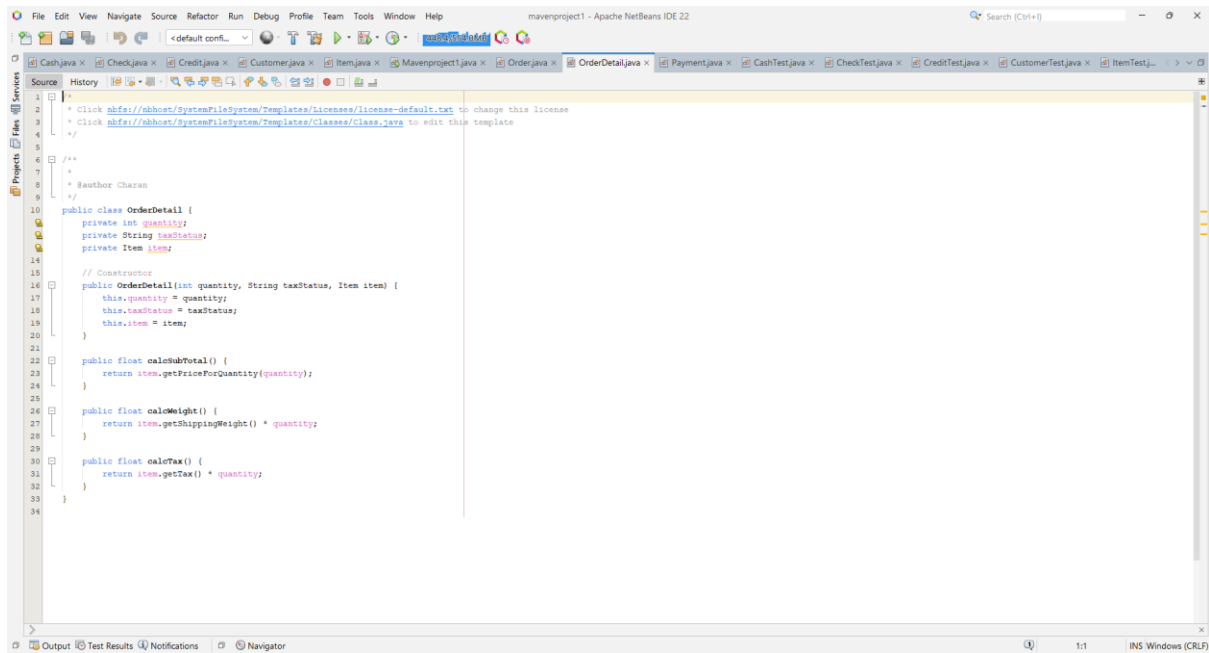
## Order.java



The screenshot shows the NetBeans IDE with the `Order.java` file open. The code implements methods for calculating subtotal, tax, total, and weight for an order, and a method to add order details.

```
26 // Calculate subtotal based on order details
27 float subtotal = 0;
28 for (OrderDetail detail : orderDetails) {
29     subtotal += detail.calcSubTotal();
30 }
31 return subtotal;
32 }
33
34 public float calcTax() {
35     // Calculate tax based on order details
36     float tax = 0;
37     for (OrderDetail detail : orderDetails) {
38         tax += detail.calcTax();
39     }
40     return tax;
41 }
42
43 public float calcTotal() {
44     return calcSubTotal() + calcTax();
45 }
46
47 public float calcTotalWeight() {
48     float totalWeight = 0;
49     for (OrderDetail detail : orderDetails) {
50         totalWeight += detail.calcWeight();
51     }
52     return totalWeight;
53 }
54
55 // Adding an OrderDetail
56 public void addOrderDetail(OrderDetail detail) {
57     orderDetails.add(detail);
58 }
59 }
```

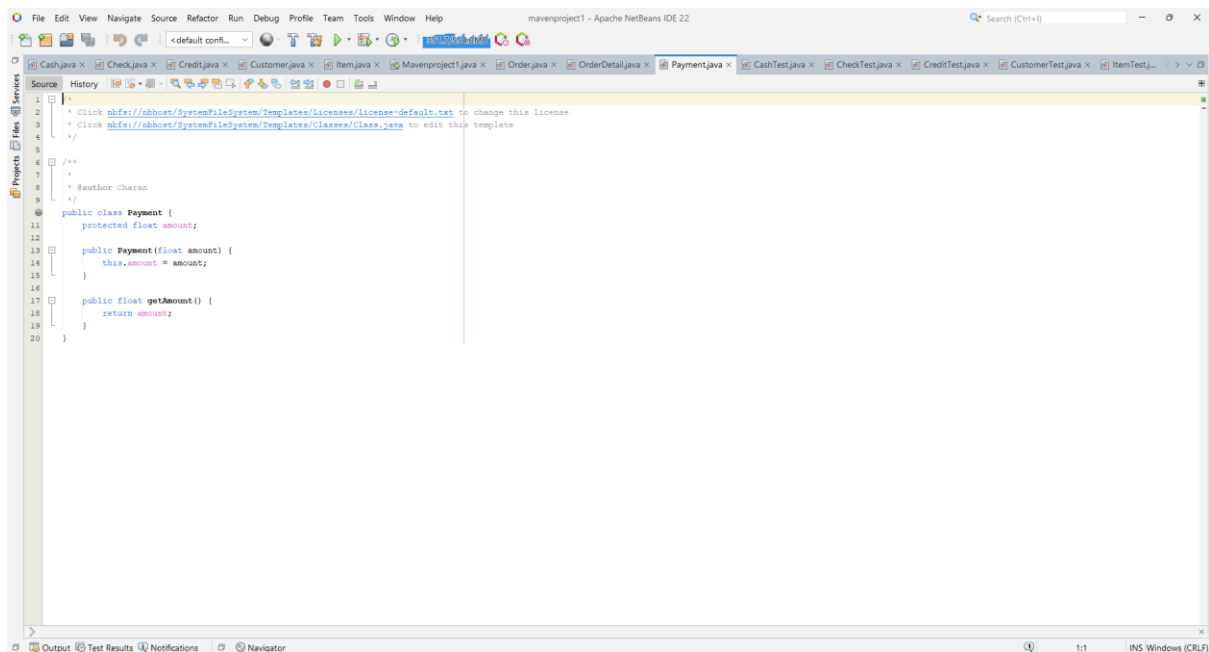
## OrderDetail.java



The screenshot shows the NetBeans IDE with the `OrderDetail.java` file open. The code defines the `OrderDetail` class with private attributes `quantity`, `taxStatus`, and `item`, and methods for calculating subtotal, weight, and tax.

```
1 /**
2  *
3  * Click dbfs://localhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
4  * Click dbfs://localhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
5  */
6
7 // Author Charan
8
9
10 public class OrderDetail {
11     private int quantity;
12     private String taxStatus;
13     private Item item;
14
15     // Constructor
16     public OrderDetail(int quantity, String taxStatus, Item item) {
17         this.quantity = quantity;
18         this.taxStatus = taxStatus;
19         this.item = item;
20     }
21
22     public float calcSubTotal() {
23         return item.getPriceForQuantity(quantity);
24     }
25
26     public float calcWeight() {
27         return item.getShippingWeight() * quantity;
28     }
29
30     public float calcTax() {
31         return item.getTax() * quantity;
32     }
33 }
34 }
```

## Payment.java



### ✓ Tool Used:

We have used JUnit for automated testing, which is one of the most popular testing frameworks in Java for automated testing. It allows a developer to write and run repeatable tests so that functionality in code can be verified. Consequently, you will be able to develop unit tests that will determine how separate components, like classes and methods, behave in isolation; hence, helping you ensure your code works as you expect before it is shipped.

These are the test-codes for automated testing of each class of the application, each test can be run multiple times with different inputs to check different conditions. It very easily integrates with Maven and Gradle, making it a powerful test automation tool for continuous integration in Java development.

## ✓ Test Packages:

### CashTest.java

The screenshot displays the Apache NetBeans IDE interface. The top pane shows the source code of `CashTest.java`. The bottom pane shows the test results for the `CashTest` class.

**Source Code:**

```
1 import org.junit.jupiter.api.BeforeEach;
2 import org.junit.jupiter.api.Test;
3 import static org.junit.jupiter.api.Assertions.*;
4
5 public class CashTest {
6     private Cash cashPayment;
7
8     @BeforeEach
9     public void setUp() {
10         cashPayment = new Cash(50.0f, 60.0f);
11     }
12
13     @Test
14     public void testAmount() {
15         assertEquals(50.0f, cashPayment.getAmount(), 0.01);
16         System.out.println("CashTest success");
17     }
18 }
```

**Test Results:**

The test results pane shows the following information:

- Test Results: 100.00 %
- The test passed. (0.054 s)
- Output - Test (CashTest): CashTest success

## CheckTest.java

The screenshot displays the Apache NetBeans IDE interface. The top pane shows the source code of `CheckTest.java`, which includes imports for JUnit, a `@BeforeEach` setup method, and a `@Test` method `testAmount()` that verifies a check's amount.

```
1 import org.junit.jupiter.api.BeforeEach;
2 import org.junit.jupiter.api.Test;
3 import static org.junit.jupiter.api.Assertions.*;
4
5 public class CheckTest {
6     private Check checkPayment;
7
8     @BeforeEach
9     public void setUp() {
10         checkPayment = new Check(100.0f, "Charath", "Bank123");
11     }
12
13     @Test
14     public void testAmount() {
15         assertEquals(100.0f, checkPayment.getAmount(), 0.01);
16         System.out.println("CheckTest success");
17     }
18 }
```

The bottom pane shows the 'Test Results' tab for the `CheckTest` class. It indicates that all tests passed with 100.00% success. The output window shows the message 'CheckTest success'.

Test Results: Tests passed: 100.00 %

Output - Test (CheckTest):

vit.adda:TheoryDAJar:1.0-SNAPSHOT (Unit) × Theory:mavenproject1:jar:1.0-SNAPSHOT (Unit) ×

The test passed. (0.056 s)

CheckTest success



## CreditTest.java

The screenshot displays an IDE window for a project named 'mavenproject1'. The main editor shows the source code of 'CreditTest.java'. The code includes imports for JUnit and AssertJ, a private 'Credit' object 'creditPayment', a '@BeforeEach' annotated 'setUp()' method that initializes 'creditPayment' with a 'Credit' object (200.0f, 'Krishna', 'Visa', '12/25'), and a '@Test' annotated 'testAmount()' method that asserts 'creditPayment.getAmount()' is equal to 0.01 and prints 'CreditTest success'.

Below the editor, the 'Test Results' tab is active, showing the test 'CreditTest' passed with 100.00% success. The output pane shows 'CreditTest success'.

```
1 import org.junit.jupiter.api.BeforeEach;
2 import org.junit.jupiter.api.Test;
3 import static org.junit.jupiter.api.Assertions.*;
4
5 public class CreditTest {
6     private Credit creditPayment;
7
8     @BeforeEach
9     public void setUp() {
10         creditPayment = new Credit(200.0f, "Krishna", "Visa", "12/25");
11     }
12
13     @Test
14     public void testAmount() {
15         assertEquals(200.0f, creditPayment.getAmount(), 0.01);
16         System.out.println("CreditTest success");
17     }
18 }
```

Test Results: CreditTest

Tests passed: 100.00 %

CreditTest success

## CustomerTest.java

The screenshot displays an IDE window with the `CustomerTest.java` file open. The code defines a `CustomerTest` class with a `private Customer customer;` field and several test methods. The IDE interface includes a menu bar, a toolbar, and a project explorer on the left. The bottom panel shows the test results for the `testSetCustomerName` method, indicating that all tests passed.

```
public class CustomerTest {  
    private Customer customer;  
  
    @BeforeEach  
    public void setUp() {  
        customer = new Customer("Charath", "123 Elm Street");  
    }  
  
    @Test  
    public void testCustomerName() {  
        assertEquals("Charath", customer.getName());  
    }  
  
    @Test  
    public void testCustomerAddress() {  
        assertEquals("123 Elm Street", customer.getAddress());  
    }  
  
    @Test  
    public void testSetCustomerName() {  
        customer.setName("Krishna");  
        assertEquals("Krishna", customer.getName());  
    }  
  
    @Test  
    public void testSetCustomerAddress() {  
        customer.setAddress("456 Oak Street");  
        assertEquals("456 Oak Street", customer.getAddress());  
        System.out.println("CustomerTest success");  
    }  
}
```

CustomerTest > testSetCustomerName

Notifications Test Results Output - Test (CustomerTest)

vit.adda:TheoryD\jar:1.0-SNAPSHOT (Unit) × Theory:mavenproject1\jar:1.0-SNAPSHOT (Unit) ×

Tests passed: 100.00 % CustomerTest success

All 4 tests passed. (0.06 s)

26:37 INS Windows (CRLF)

## ItemTest.java

The screenshot displays the Apache NetBeans IDE interface. The top pane shows the source code of `ItemTest.java`, which includes imports for JUnit and Assert, a private `Item` instance, and several test methods: `setUp()`, `testGetPriceForQuantity()`, `testGetTax()`, `testInStock()`, and `testGetShippingWeight()`. The bottom pane shows the test results for `ItemTest`, indicating that all 4 tests passed successfully (100.00 %).

```
1 import org.junit.jupiter.api.BeforeEach;
2 import org.junit.jupiter.api.Test;
3 import static org.junit.jupiter.api.Assertions.*;
4
5 public class ItemTest {
6
7     private Item item;
8
9     @BeforeEach
10    public void setUp() {
11        item = new Item(2.0f, "Test Item");
12    }
13
14    @Test
15    public void testGetPriceForQuantity() {
16        assertEquals(30.0f, item.getPriceForQuantity(3), 0.01);
17    }
18
19    @Test
20    public void testGetTax() {
21        assertEquals(1.0f, item.getTax(), 0.01);
22    }
23
24    @Test
25    public void testInStock() {
26        assertTrue(item.inStock());
27    }
28
29    @Test
30    public void testGetShippingWeight() {
31        assertEquals(2.0f, item.getShippingWeight(), 0.01);
32        System.out.println("ItemTest success");
33    }
34 }
```

Test Results: All 4 tests passed. (0.063 s)

ItemTest success

## OrderTest.java

The screenshot displays the Apache NetBeans IDE interface. The top pane shows the source code of `OrderTest.java`, which includes a `@BeforeEach` setup method and four `@Test` methods: `testCalcSubTotal`, `testCalcTax`, `testCalcTotal`, and `testCalcTotalWeight`. The bottom pane shows the test results for the `OrderTest` class, indicating that all 4 tests passed with a 100.00 % success rate. The test results pane also shows the output of the tests, including the message "OrderTest success".

```
8 private Order order;
9 private Item item;
10 private OrderDetail orderDetail;
11
12
13 @BeforeEach
14 public void setUp() {
15     item = new Item(2.0f, "Test Item");
16     order = new Order(new Date(), "Pending");
17     orderDetail = new OrderDetail(3, "Taxable", item);
18     order.addOrderDetail(orderDetail);
19 }
20
21 @Test
22 public void testCalcSubTotal() {
23     assertEquals(30.0f, order.calcSubTotal(), 0.01);
24 }
25
26 @Test
27 public void testCalcTax() {
28     assertEquals(3.0f, order.calcTax(), 0.01);
29 }
30
31 @Test
32 public void testCalcTotal() {
33     assertEquals(33.0f, order.calcTotal(), 0.01);
34 }
35
36 @Test
37 public void testCalcTotalWeight() {
38     assertEquals(6.0f, order.calcTotalWeight(), 0.01);
39     System.out.println("OrderTest success");
40 }
41 }
```

Test Results: OrderTest

Tests passed: 100.00 %

All 4 tests passed. (0.063 s)

OrderTest success

## OrderDetailTest.java

The screenshot displays the Apache NetBeans IDE interface. The top pane shows the source code of `OrderDetailTest.java`. The bottom pane shows the test results for the `testCalcTax` method, indicating that all tests passed.

```
1 import org.junit.jupiter.api.BeforeEach;
2 import org.junit.jupiter.api.Test;
3 import static org.junit.jupiter.api.Assertions.*;
4
5 public class OrderDetailTest {
6
7     private Item item;
8     private OrderDetail orderDetail;
9
10    @BeforeEach
11    public void setUp() {
12        item = new Item(2.0f, "Test Item");
13        orderDetail = new OrderDetail(3, "Taxable", item);
14    }
15
16    @Test
17    public void testCalcSubTotal() {
18        assertEquals(30.0f, orderDetail.calcSubTotal(), 0.01);
19    }
20
21    @Test
22    public void testCalcTax() {
23        assertEquals(3.0f, orderDetail.calcTax(), 0.01);
24    }
25
26    @Test
27    public void testCalcWeight() {
28        assertEquals(6.0f, orderDetail.calcWeight(), 0.01);
29        System.out.println("OrderDetailTest success");
30    }
31 }
```

Test Results:

- Tests passed: 100.00 %
- All 3 tests passed. (0.064 s)
- OrderDetailTest success

✓ **Git Hub Link (for all source and junit test codes of maven project):**

[https://github.com/Sree-Krishna04/ADDA\\_TH\\_DA.git](https://github.com/Sree-Krishna04/ADDA_TH_DA.git)