

CS5800 – Advanced Software Engineering

Professor Nima Davarpanah

Homework 4 – Structural Design Pattern

Devaansh Mann

GitHub Link: <https://github.com/DevaanshMann/CS5800-HW4>

1. Bridge (E-Commerce)

```
1 package Bridge;
2
3 import java.util.*;
4
5 abstract class PaymentNotification {
6     protected final NotificationChannel channel;
7     protected final String customer;
8     protected final double amount;
9
10    protected PaymentNotification(
11        NotificationChannel channel, String customer,
12        double amount) {
13        this.channel = channel;
14        this.customer = customer;
15        this.amount = amount;
16    }
17    public abstract void notifyCustomer();
18 }
19
20 interface NotificationChannel {
21     void send(String to, String message);
22 }
23
24 class EmailChannel implements NotificationChannel {
25     @Override
26     public void send(String to, String message) {
27         System.out.println("EMAIL to " + to +
28             " :: " + message);
29     }
30 }
31
32 class SmsChannel implements NotificationChannel {
33     @Override
34     public void send(String to, String message) {
35         System.out.println("SMS to " + to + " :: "
36             + message);
37     }
38 }
39
40 class PushChannel implements NotificationChannel {
41     @Override
```

```
38     public void send(String to, String message) {
39         System.out.println("PUSH to " + to + " :: "
        + message);
40     }
41 }
42
43 class OnlinePaymentNotification extends
    PaymentNotification {
44     private final String method;
45     public OnlinePaymentNotification(
        NotificationChannel c, String customer, double
        amount, String method) {
46         super(c, customer, amount);
47         this.method = method;
48     }
49     @Override
50     public void notifyCustomer() {
51         channel.send(customer, String.format("
        Online payment of $%.2f received via %s. Thank you
        !", amount, method));
52     }
53 }
54
55 class CashOnDeliveryPayment extends
    PaymentNotification {
56     public CashOnDeliveryPayment(
        NotificationChannel c, String customer, double
        amount) {
57         super(c, customer, amount);
58     }
59     @Override
60     public void notifyCustomer() {
61         channel.send(customer, String.format("Cash
        on Delivery: Please have $%.2f ready upon arrival."
        , amount));
62     }
63 }
64
65 class BitcoinPayment extends PaymentNotification {
66     private final String txId;
67     public BitcoinPayment(NotificationChannel c,
```

```
67 String customer, double amount, String txId) {
68     super(c, customer, amount);
69     this.txId = txId;
70 }
71 @Override
72 public void notifyCustomer() {
73     channel.send(customer, String.format("
    Bitcoin payment of $%.2f confirmed. TxId=%s",
    amount, txId));
74 }
75 }
76
77
78
```

```
1 package Bridge;
2
3 public class BridgeDriver {
4     public static void main(String[] args) {
5         NotificationChannel email = new
        EmailChannel();
6         NotificationChannel sms    = new SmsChannel
        ();
7         NotificationChannel push   = new PushChannel
        ();
8
9         new OnlinePaymentNotification(email, "alex@
        email.com", 129.99, "Credit Card").notifyCustomer
        ();
10        new OnlinePaymentNotification(sms, "+1-555-
        0100", 49.99, "Zelle").notifyCustomer();
11
12        new CashOnDeliveryPayment(email, "dave@
        email.com", 19.99).notifyCustomer();
13        new CashOnDeliveryPayment(sms, "+1-818-0101
        ", 89.00).notifyCustomer();
14
15        new BitcoinPayment(email, "dm@email.com",
        250.00, "tx_dm2312").notifyCustomer();
16        new BitcoinPayment(sms, "+1-818-0102", 75.
        25, "tx_mm2707").notifyCustomer();
17
18        new OnlinePaymentNotification(push, "
        dave_device_id", 15.49, "Credit Card").
        notifyCustomer();
19        new CashOnDeliveryPayment(push, "
        erin_device_id", 42.00).notifyCustomer();
20    }
21 }
```

```
Run BridgeDriver x
/Library/Java/JavaVirtualMachines/temurin-25.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=59873 -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -cl
EMAIL to alex@email.com :: Online payment of $129.99 received via Credit Card. Thank you!
SMS to +1-555-0100 :: Online payment of $49.99 received via Zelle. Thank you!
EMAIL to dave@email.com :: Cash on Delivery: Please have $19.99 ready upon arrival.
SMS to +1-818-0101 :: Cash on Delivery: Please have $89.00 ready upon arrival.
EMAIL to dm@email.com :: Bitcoin payment of $250.00 confirmed. TxId=tx_dm2312
SMS to +1-818-0102 :: Bitcoin payment of $75.25 confirmed. TxId=tx_mm2707
PUSH to dave_device_id :: Online payment of $15.49 received via Credit Card. Thank you!
PUSH to erin_device_id :: Cash on Delivery: Please have $42.00 ready upon arrival.

Process finished with exit code 0
```

CS5800-HW4 > src > main > java > Bridge > BridgeDriver 21:2 LF UTF-8 4 spaces

2. Decorator (Restaurant)


```
1 package Decorator;
2
3 import java.util.*;
4
5 interface FoodItem {
6     String description();
7     double cost();
8 }
9
10 class Burger implements FoodItem {
11     @Override
12     public String description() {
13         return "Burger";
14     }
15     @Override
16     public double cost() {
17         return 5.00;
18     }
19 }
20
21 class Fries implements FoodItem {
22     @Override
23     public String description() {
24         return "Fries";
25     }
26     @Override
27     public double cost() {
28         return 2.50;
29     }
30 }
31
32 class HotDog implements FoodItem {
33     @Override
34     public String description() {
35         return "Hot Dog";
36     }
37
38     @Override
39     public double cost() {
40         return 3.50;
41     }
42 }
```

```
42 }
43
44 abstract class ToppingDecorator implements FoodItem
    {
45     protected final FoodItem base;
46     protected ToppingDecorator(FoodItem base) {
47         this.base = base;
48     }
49 }
50
51 class Ketchup extends ToppingDecorator {
52     public Ketchup(FoodItem base) {
53         super(base);
54     }
55
56     @Override
57     public String description() {
58         return base.description() + " + Ketchup";
59     }
60
61     @Override
62     public double cost() {
63         return base.cost() + 0.20;
64     }
65 }
66
67 class Cheese extends ToppingDecorator {
68     public Cheese(FoodItem base) {
69         super(base);
70     }
71
72     @Override
73     public String description() {
74         return base.description() + " + Cheese";
75     }
76
77     @Override
78     public double cost() {
79         return base.cost() + 0.50;
80     }
81 }
```

```
82
83 class Onions extends ToppingDecorator {
84     public Onions(FoodItem base) {
85         super(base);
86     }
87
88     @Override
89     public String description() {
90         return base.description() + " + Onions";
91     }
92
93     @Override
94     public double cost() {
95         return base.cost() + 0.30;
96     }
97 }
98
99 class Order {
100     private final List<FoodItem> items = new
    ArrayList<>();
101
102     public void add(FoodItem item) {
103         items.add(item);
104     }
105
106     public double subtotal() {
107         return items.stream().mapToDouble(FoodItem
    ::cost).sum();
108     }
109
110     public void printItems() {
111         for (FoodItem i : items) {
112             System.out.printf(" - %-40s $%.2f%n",
    i.description(), i.cost());
113         }
114     }
115 }
116
117 enum LoyaltyTier {
118     NONE(0.00), SILVER(0.05), GOLD(0.10), PLATINUM
    (0.20);

```

```
119     final double discountRate;
120     LoyaltyTier(double r) {
121         this.discountRate = r;
122     }
123 }
124
125 class LoyaltyStatus {
126     private final LoyaltyTier tier;
127     public LoyaltyStatus(LoyaltyTier tier) {
128         this.tier = tier;
129     }
130     public double applyDiscount(double amount) {
131         return amount * (1.0 - tier.discountRate);
132     }
133     public LoyaltyTier tier() {
134         return tier;
135     }
136 }
137
138
139
```

```
1 package Decorator;
2
3 public class DecoratorDriver {
4     public static void main(String[] args) {
5
6         FoodItem item1 = new Cheese(new Ketchup(new
          Onions(new Burger())));
7         FoodItem item2 = new Onions(new Cheese(new
          HotDog()));
8         FoodItem item3 = new Ketchup(new Fries());
9
10        Order order = new Order();
11        order.add(item1);
12        order.add(item2);
13        order.add(item3);
14
15        System.out.println("          DECORATOR
          DEMO");
16        order.printItems();
17        double subtotal = order.subtotal();
18        System.out.printf("Subtotal: $%.2f\n",
          subtotal);
19
20        LoyaltyStatus status = new LoyaltyStatus(
          LoyaltyTier.PLATINUM);
21        double totalAfterDiscount = status.
          applyDiscount(subtotal);
22        System.out.printf("Loyalty Tier: %s (%.0f
          %% off)\n", status.tier(), status.tier().
          discountRate * 100);
23        System.out.printf("Total after discount: $
          %.2f\n", totalAfterDiscount);
24    }
25 }
```

```
Run DecoratorDriver x
/Library/Java/JavaVirtualMachines/temurin-25.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=59877 -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -cla
DECORATOR DEMO
- Burger + Onions + Ketchup + Cheese      $6.00
- Hot Dog + Cheese + Onions               $4.30
- Fries + Ketchup                         $2.70
Subtotal: $13.00
Loyalty Tier: PLATINUM (20% off)
Total after discount: $10.40

Process finished with exit code 0
```

CS5800-HW4 > src > main > java > Decorator > DecoratorDriver 25:2 LF UTF-8 4 spaces

Testing

BridgeTesting


```
1 package Bridge;
2
3 import org.junit.jupiter.api.*;
4 import java.io.ByteArrayOutputStream;
5 import java.io.PrintStream;
6 import static org.junit.jupiter.api.Assertions.*;
7
8 public class BridgeTest {
9
10     private final PrintStream originalOut = System.
out;
11     private ByteArrayOutputStream capture;
12
13     @BeforeEach
14     void setUp() {
15         capture = new ByteArrayOutputStream();
16         System.setOut(new PrintStream(capture));
17     }
18
19     @AfterEach
20     void tearDown() {
21         System.setOut(originalOut);
22     }
23
24     @Test
25     void emailSend() {
26         NotificationChannel ch = new EmailChannel
();
27         ch.send("user@email.com", "pass");
28         String out = capture.toString();
29         assertTrue(out.contains("EMAIL to user@
email.com :: pass"));
30     }
31
32     @Test
33     void smsSend() {
34         NotificationChannel ch = new SmsChannel();
35         ch.send("+1-555-0101", "otp:1234");
36         String out = capture.toString();
37         assertTrue(out.contains("SMS to +1-555-0101
:: otp:1234"));
38     }
39 }
```

```
38     }
39
40     @Test
41     void pushSend() {
42         NotificationChannel ch = new PushChannel();
43         ch.send("device_abc", "received");
44         String out = capture.toString();
45         assertTrue(out.contains("PUSH to device_abc
:: received"));
46     }
47
48     @Test
49     void onlineNotification() {
50         NotificationChannel email = new
EmailChannel();
51         PaymentNotification n = new
OnlinePaymentNotification(email, "alex@email.com",
129.99, "Credit Card");
52         n.notifyCustomer();
53         String out = capture.toString();
54         assertTrue(out.contains("EMAIL to alex@
email.com"));
55         assertTrue(out.contains("Online payment of
$129.99 received via Credit Card. Thank you!"));
56     }
57
58     @Test
59     void codNotification() {
60         NotificationChannel sms = new SmsChannel();
61         PaymentNotification n = new
CashOnDeliveryPayment(sms, "+1-555-0101", 89.00);
62         n.notifyCustomer();
63         String out = capture.toString();
64         assertTrue(out.contains("SMS to +1-555-0101
"));
65         assertTrue(out.contains("Cash on Delivery:
Please have $89.00 ready upon arrival."));
66     }
67
68     @Test
69     void bitcoinNotification() {
```

```
70         NotificationChannel push = new PushChannel
      ();
71         PaymentNotification n = new BitcoinPayment
      (push, "dev_device", 15.49, "tx_abc");
72         n.notifyCustomer();
73         String out = capture.toString();
74         assertTrue(out.contains("PUSH to
      dev_device"));
75         assertTrue(out.contains("Bitcoin payment
      of $15.49 confirmed. TxId=tx_abc"));
76     }
77 }
78
```

Run BridgeTest x

✓ BridgeTest (Bridge) 43 ms

- ✓ smsSend() 25 ms
- ✓ codNotification() 15 ms
- ✓ onlineNotification() 1 ms
- ✓ bitcoinNotification() 1 ms
- ✓ pushSend() 1 ms
- ✓ emailSend()

✓ 6 tests passed 6 tests total, 43ms

/Library/Java/JavaVirtualMachines/temurin-25.jdk/Contents/Home/bin/java ...

Process finished with exit code 0

CS5800-HW4 > src > test > java > Bridge > BridgeTest 8:14 LF UTF-8 4 spaces

DecoratorTesting

```
1 package Decorator;
2
3 import org.junit.jupiter.api.*;
4 import java.io.ByteArrayOutputStream;
5 import java.io.PrintStream;
6 import static org.junit.jupiter.api.Assertions.*;
7
8 public class DecoratorTest {
9
10     private final PrintStream originalOut = System.
out;
11     private ByteArrayOutputStream capture;
12
13     @BeforeEach
14     void setUp() {
15         capture = new ByteArrayOutputStream();
16         System.setOut(new PrintStream(capture));
17     }
18
19     @AfterEach
20     void tearDown() {
21         System.setOut(originalOut);
22     }
23
24     @Test void burgerDescription() {
25         FoodItem b = new Burger();
26         assertEquals("Burger", b.description());
27     }
28
29     @Test void burgerCost() {
30         FoodItem b = new Burger();
31         assertEquals(5.00, b.cost(), 1e-2);
32     }
33
34     @Test void friesDescription() {
35         FoodItem f = new Fries();
36         assertEquals("Fries", f.description());
37     }
38
39     @Test void friesCost() {
40         FoodItem f = new Fries();
```

```
41         assertEquals(2.50, f.cost(), 1e-2);
42     }
43
44     @Test void hotDogDescription() {
45         FoodItem h = new HotDog();
46         assertEquals("Hot Dog", h.description());
47     }
48
49     @Test void hotDogCost() {
50         FoodItem h = new HotDog();
51         assertEquals(3.50, h.cost(), 1e-2);
52     }
53
54     @Test void ketchupDescription() {
55         FoodItem item = new Ketchup(new Burger());
56         assertEquals("Burger + Ketchup", item.
description());
57     }
58
59     @Test void ketchupCost() {
60         FoodItem item = new Ketchup(new Burger());
61         assertEquals(5.00 + 0.20, item.cost(), 1e-2
);
62     }
63
64     @Test void cheeseDescription() {
65         FoodItem item = new Cheese(new Fries());
66         assertEquals("Fries + Cheese", item.
description());
67     }
68
69     @Test void cheeseCost() {
70         FoodItem item = new Cheese(new Fries());
71         assertEquals(2.50 + 0.50, item.cost(), 1e-2
);
72     }
73
74     @Test void onionsDescription() {
75         FoodItem item = new Onions(new HotDog());
76         assertEquals("Hot Dog + Onions", item.
description());
```

```
77     }
78
79     @Test void onionsCost() {
80         FoodItem item = new Onions(new HotDog());
81         assertEquals(3.50 + 0.30, item.cost(), 1e-
82     2);
83     }
84
85     @Test void orderSubtotal() {
86         Order order = new Order();
87         order.add(new Cheese(new Ketchup(new
88     Burger()))); // 5.70
89         order.add(new Onions(new Cheese(new HotDog
90     ())))); // 4.30
91         order.add(new Ketchup(new Fries
92     ())))); // 2.70
93         assertEquals(12.70, order.subtotal(), 1e-2
94     );
95     }
96
97     @Test void orderPrintItems() {
98         Order order = new Order();
99         FoodItem item = new Cheese(new Ketchup(new
100     Burger()))); // "Burger + Ketchup + Cheese"
101         order.add(item);
102         order.printItems();
103         String out = capture.toString();
104         assertTrue(out.contains("Burger + Ketchup
105     + Cheese"));
106         assertTrue(out.contains("$5.70"));
107     }
108
109     @Test void loyaltyApplyDiscount() {
110         LoyaltyStatus status = new LoyaltyStatus(
111     LoyaltyTier.GOLD);
112         assertEquals(100.0 * 0.90, status.
113     applyDiscount(100.0), 1e-2);
114     }
115
116     @Test void loyaltyTierGetter() {
117         LoyaltyStatus status = new LoyaltyStatus(
```



```
108 LoyaltyTier.SILVER);
109     assertEquals(LoyaltyTier.SILVER, status.
    tier());
110 }
111
112 @Test void stackedDecorators() {
113     FoodItem item = new Onions(new Cheese(new
    Ketchup(new Burger())));
114     assertEquals(6.00, item.cost(), 1e-2);
115     assertEquals("Burger + Ketchup + Cheese +
    Onions", item.description());
116 }
117 }
118
119
```

Run DecoratorTest x

✓ DecoratorTest (Decorator) 41 ms

- ✓ burgerCost() 18 ms
- ✓ orderPrintItems() 12 ms
- ✓ cheeseDescription() 1 ms
- ✓ onionsDescription() 1 ms
- ✓ friesCost() 1 ms
- ✓ hotDogDescription() 1 ms
- ✓ onionsCost() 1 ms
- ✓ orderSubtotal() 4 ms
- ✓ stackedDecorators() 1 ms
- ✓ ketchupDescription() 2 ms
- ✓ friesDescription() 1 ms
- ✓ ketchupCost() 1 ms
- ✓ loyaltyApplyDiscount() 1 ms
- ✓ cheeseCost() 1 ms
- ✓ burgerDescription() 1 ms
- ✓ loyaltyTierGetter() 1 ms
- ✓ hotDogCost() 1 ms

✓ 17 tests passed 17 tests total, 41 ms

/Library/Java/JavaVirtualMachines/temurin-25.jdk/Contents/Home/bin/java ...

Process finished with exit code 0

CS5800-HW4 > src > test > java > Decorator > DecoratorTest > hotDogDescription 46:27 LF UTF-8 4 spaces

CombinedTesting

```
1 package CombinedTest;
2
3 import org.junit.platform.suite.api.SelectPackages;
4 import org.junit.platform.suite.api.Suite;
5
6 @Suite
7 @SelectPackages({"Bridge", "Decorator"})
8 public class combinedTest {
9 }
10
```

Run combinedTest x

✓ combinedTest (CombinedTest) 47 ms

- ✓ BridgeTest 36 ms
 - ✓ smsSend() 23 ms
 - ✓ codNotification() 10 ms
 - ✓ onlineNotification() 1 ms
 - ✓ bitcoinNotification() 1 ms
 - ✓ pushSend() 1 ms
 - ✓ emailSend() 1 ms
- ✓ DecoratorTest 11 ms
 - ✓ burgerCost() 1 ms
 - ✓ orderPrintItems() 3 ms
 - ✓ cheeseDescription() 1 ms
 - ✓ onionsDescription() 1 ms
 - ✓ friesCost() 1 ms
 - ✓ hotDogDescription() 1 ms
 - ✓ onionsCost() 1 ms
 - ✓ orderSubtotal() 2 ms
 - ✓ stackedDecorators() 1 ms
 - ✓ ketchupDescription() 1 ms
 - ✓ friesDescription() 1 ms
 - ✓ ketchupCost() 1 ms
 - ✓ loyaltyApplyDiscount() 1 ms
 - ✓ cheeseCost() 1 ms
 - ✓ burgerDescription() 1 ms
 - ✓ loyaltyTierGetter() 1 ms
 - ✓ hotDogCost() 1 ms

✓ 23 tests passed 23 tests total, 47 ms

/Library/Java/JavaVirtualMachines/temurin-25.jdk/Contents/Home/bin/java ...

Process finished with exit code 0

CS5800-HW4 > src > test > java > CombinedTest > combinedTest 8:14 LF UTF-8 4 spaces