

Practical About jUnit 4: Equivalence Partitioning / Boundary Value Analysis / Branch Coverage

The following Java method calculates the **final order price** for a customer based on item prices, customer type, VIP status, discount codes, and an additional **service fee**. The method applies different discount rates depending on the customer's status and discount codes. If the final price is **below 50**, a fixed **service fee of \$5** will be added.

```
public class PriceCalculator {

    public double calculateTotalPrice(double[] itemPrices, String
customerType, boolean isVIP, String discountCode) {
        if (itemPrices == null || itemPrices.length == 0) {
            throw new IllegalArgumentException("No items in the order.");
        }
        double totalPrice = 0.0;
        for (double price : itemPrices) {
            if (price <= 0) {
                throw new IllegalArgumentException("Item price must be
greater than zero.");
            }
            totalPrice += price;
        }

        double discount = 0.0;
        if (isVIP) {
            discount = 0.15; // 15% discount for VIP customers
        } else if (customerType.equalsIgnoreCase("Regular")) {
            discount = 0.05; // 5% discount for Regular customers
        }

        if (discountCode != null && !discountCode.isEmpty()) {
            if (discountCode.equals("SALE15")) {
                discount += 0.15; // Additional 15% discount
            } else if (discountCode.equals("WELCOME5")) {
                discount += 0.05; // Additional 5% discount
            }
        }

        double finalPrice = totalPrice * (1 - discount);

        if (finalPrice < 50) {
            finalPrice += 5; // Service fee if the price is below 50
        }

        return finalPrice < 0 ? 0 : finalPrice; // Ensure price is not
negative
    }
}
```

Requirements:

- Write **JUnit test cases** that cover:
 - Equivalence Partitioning
 - Boundary Value Analysis
 - Branch Coverage
- Provide a **one-line summary** for each test case. (Comment in code, above the test unit function).

Example:

//ID: TC1; Test for _____; Input parameter: _____; Expected result: _____

- Ensure **100% code coverage**.
- Copy the **test code** and **summary table** into a Word document for submission.

Example Code:

```
public class PriceCalculatorTest {

    // TC1: Test for null items array; Input: itemPrices=null,
    // customerType="VIP", isVIP=true, discountCode="SALE15"; Expected:
    // IllegalArgumentException ("No items in the order.")
    @Test(expected = IllegalArgumentException.class)
    public void testNullItems() {
        PriceCalculator calculator = new PriceCalculator();
        calculator.calculateTotalPrice(null, "VIP", true, "SALE15");
    }

    // TC2: Test for empty items array; Input: itemPrices={},
    // customerType="Regular", isVIP=false, discountCode=null; Expected:
    // IllegalArgumentException ("No items in the order.")
    @Test(expected = IllegalArgumentException.class)
    public void testEmptyItems() {
        PriceCalculator calculator = new PriceCalculator();
        calculator.calculateTotalPrice(new double[] {}, "Regular", false,
null);
    }

    // Add more test cases
}
```

Example summary table

ID	Test for	Input Parameter	Expected Result	Test Category
TC1	Null items array	itemPrices=null, customerType="VIP", isVIP=true, discountCode="SALE15"	Throws IllegalArgumentException ("No items in the order.")	EP, BC
TC2	Empty items array	itemPrices={}, customerType="Regular", isVIP=false, discountCode=null	Throws IllegalArgumentException ("No items in the order.")	EP, BC
...	...			
...		

----- THE END -----