Task: Equivalence Classes (Part – a , b and d):

Equivalence Classes :

a) As a first step, derive suitable equivalence classes from DHL's price model and take the different package characteristics into account. Please provide some information about the equivalence relation you've used for each characteristic.

Solution:

Step 1:

Extract input and output conditions:

1. Output is the price to be paid to DHL for delivery
2. Destination Codes are used such as zone 0 for inside Germany, zone 1 for EU and zone 2 to 8 for rest of the world
3. Weight of the package can be anywhere from 1Kg to 31,5Kg
4. Maximum Dimensions (L*B*H) has to be specified based on Package type such as (PäckchenXS, PaketXL) for zone 0 and based on weight in zone 1-8
5. Price depends on where the payment is made (Online/Branch/DHL Shop)
6. Quantity has to be specified to get separate offers for more sets of packages

Step 2:

Defining equivalence classes:

Conditions	**Valid EC**	**Invalid EC**
Destination Zone	**0(1)**	**Zone <0(1a)**
Max Weight	**0<=Weight<=31,5(2)**	**Weight<0(2a), Weight >31,5(2b)**
Max Dimension	**L*B*H: 30 x 30 x 15 cm (upto weight =1kg) (3.1)/ 60 x 30 x 15 cm(upto weight =2kg)(3.2)/ 120 x 60 x 60 cm (upto weight= 31,5)(3.3)(**	**Dimension greater than the maximum limit for specified weight(3a)**
Payment Method	**Online/Branch/ dhl-shop.de valid for Quantity ={10,50,100}(4)**	**Other than three methods of payment(4a)**
Quantity	**1/ 10/ 50/100(5)**	**Quantity other than {1,10,50,100}(5a)**

Conditions	**Valid EC**	**Invalid EC**
Destination Zone	**1-8**	**Zone other than {0,1,2,3,4,5,6,7,8}**
Max Weight	**1<=Weight<=31,5**	**Weight<1, Weight >31,5**
Max Dimension	**For Weight<=2kg, L+B+H=90cm && {L,B,H} <60 For Weight > 2kg, L*B*H:**	**For Weight<=2kg, L+B+H >90 cm For Weight > 2kg ,**

	{120 x 60 x60 cm}	**L*B*H dimension greater than {120 x 60 x 60}**
Payment Method	**Online/Branch/ dhl-shop.de valid for Quantity ={10,50,100}**	**Other than three methods**
Quantity	**1/3/10/50/100**	**Quantity other than {1,3,10,50,100}**

b) Select representatives and create a test suite which satisfies the weak equivalence class test criterion (This can be done manually). Enrich the test cases with expected results. You can derive them manually from the document or from the DHL online service. Store the resulting test suite as a CSV file.

Solution:

Representatives test suite for weak equivalence class criteria:

testcase	**Destination Zone**	**Max Weight**	**Max Dimension(L+B+H)**	**Payment Method**	**Quantity**	**EC**	**Expected result**	**Expected Value**
1	0	10	47	Online	10	1,2,3,4,5	Valid	182.83
2	0	10	47	Online	19	1,2,3,4,5a	Invalid	-1
3	0	10	47	Online1	10	1,2,3,4a,5	Invalid	-1
4	0	10	250	Online	10	1,2,3a,4,5	Invalid	-1
5	0	111	47	Online	10	1,2a,3,4,5	Invalid	-1
6	-1	10	47	Online	10	1a,2,3,4,5	Invalid	-1

Here we consider the invalid conditions as basis for weak equivalence class along with the valid conditions which tries to accommodate all the valid conditions possible by extending the range. However, the individual internal conditions can’t be shown because they are not independent.

c) Select representatives and create a test suite which satisfies the weak equivalence class test criterion (This can be done manually). Enrich the test cases with expected results. You can derive them manually from the document or from the DHL online service. Store the resulting test suite as a CSV file.

CalculatePrice Function : The CalculatePrice function takes the parameters type, weight, zone, dimension, paymentMethod and quantity. This function

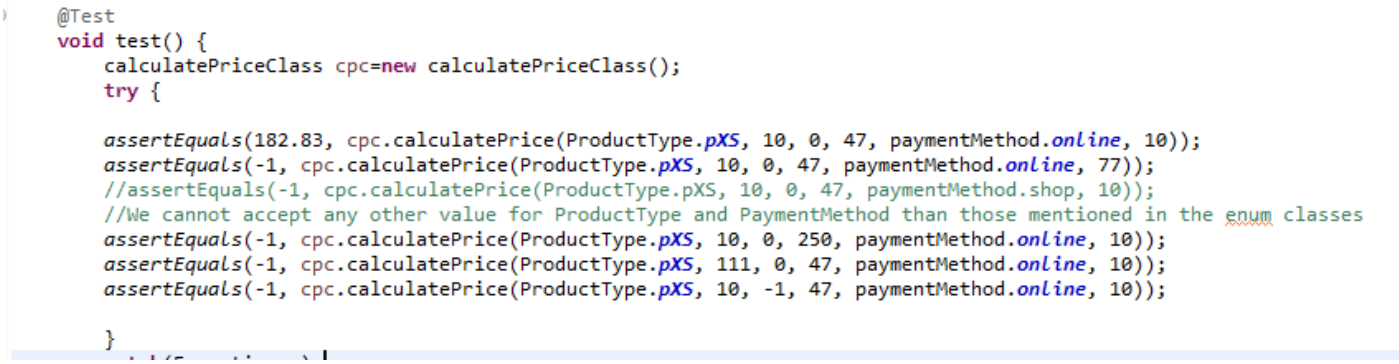
calculates the price similar to the one shown in the DHL site. The calculation differs based on productType and paymentMethod. The invalid inputs are also handled by returning -1 if they are not in specified range/invalid.

```
price=(weight*(zone+1)*dimension*quantity)/100
```

```
public class calculatePriceClass {

	public double calculatePrice(ProductType type, double weight, int
zone, double dimension, paymentMethod pm, int quantity)
	{
		double price=1.0;
		double pricePerUnit=0.0;
		double total=0.0;
		if( quantity <0 || quantity%10!=0 || quantity>50)
			return -1;
		if( dimension <0 || dimension>240)
			return -1;
		if( zone <0 || zone>8)
			return -1;
		if( weight <0 || weight>31.5)
			return -1;

		switch(type) {
		case pXS:
			if(pm == paymentMethod.online)
				pricePerUnit=3.89;
			else
				pricePerUnit=4.0;
			break;
		case pS:
			if(pm == paymentMethod.online)
				pricePerUnit=4.39;
			else
				pricePerUnit=4.5;
			break;
		case s:
			if(pm == paymentMethod.online)
				pricePerUnit=4.99;
			else
				pricePerUnit=0;
			break;
		case m:
			if(pm == paymentMethod.online)
				pricePerUnit=5.99;
			else
				pricePerUnit=6.99;
			break;
		case l:
			if(pm == paymentMethod.online)
				pricePerUnit=8.49;
			else
				pricePerUnit=9.49;
			break;
```

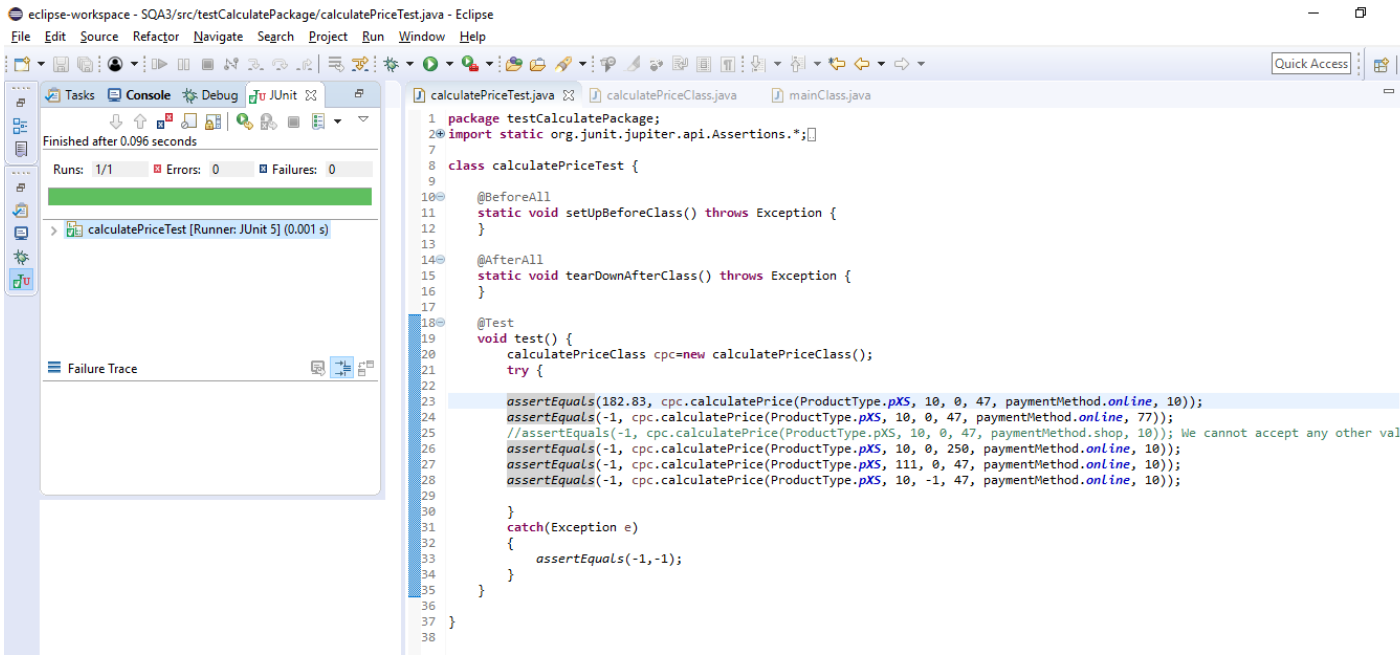
```
            case xl:
                  pricePerUnit=16.49;
                  break;
            case rgp:
                  pricePerUnit=16.49;
                  break;
            default: pricePerUnit=-1.0;
            }


            price=weight*(zone+1)*dimension*quantity;
            total=(price*pricePerUnit)/100;

            if(total>10 && pm == paymentMethod.shop)
                  return total+9.99;
            return total;
      }
}
```

Considering the equivalence class in 2b (the csv file), we are writing test cases as shown below:



```
@Test
void test() {
    calculatePriceClass cpc=new calculatePriceClass();
    try {

    assertEquals(182.83, cpc.calculatePrice(ProductType.pXS, 10, 0, 47, paymentMethod.online, 10));
    assertEquals(-1, cpc.calculatePrice(ProductType.pXS, 10, 0, 47, paymentMethod.online, 77));
    //assertEquals(-1, cpc.calculatePrice(ProductType.pXS, 10, 0, 47, paymentMethod.shop, 10));
    //We cannot accept any other value for ProductType and PaymentMethod than those mentioned in the enum classes
    assertEquals(-1, cpc.calculatePrice(ProductType.pXS, 10, 0, 250, paymentMethod.online, 10));
    assertEquals(-1, cpc.calculatePrice(ProductType.pXS, 111, 0, 47, paymentMethod.online, 10));
    assertEquals(-1, cpc.calculatePrice(ProductType.pXS, 10, -1, 47, paymentMethod.online, 10));

    }
```

Instead of importing the CSV file, we are testing using the same values as shown above. And we can see that all the above test cases have passed as seen below.



The project is there inside the folder SQA3.

SQA3.zip

d) How can the equivalence classes be extended by boundary values? Please note: You don't have to implement this.

Solution:

The equivalence classes can be extended to boundary values by ordering the set of test cases according to each category/condition and then defining the boundaries as follows. This boundary has the maximum range of all the subcategories or conditions.

The equivalence class for all zones can be extended by boundary values as follows:

Condition	min-	min	min+	nom	max-	max	max+
Destination Zone	-1	0	1	4	7	8	9
Max Weight	-1	0	1	16	31	31.5	32
Max Dimension(upto 31.5kg)	-1x-1x-1	0x0x0	1x1x1	60x30x30	119x59x59	120x60x60	121x61x61
Payment Method	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Quantity	0	1	3	10	50	100	101

Payment method and quantity does not yield proper results as the in between range input don’t have valid output.

Max dimensions tries to accommodate all the categories and hence has the highest range of all categories.