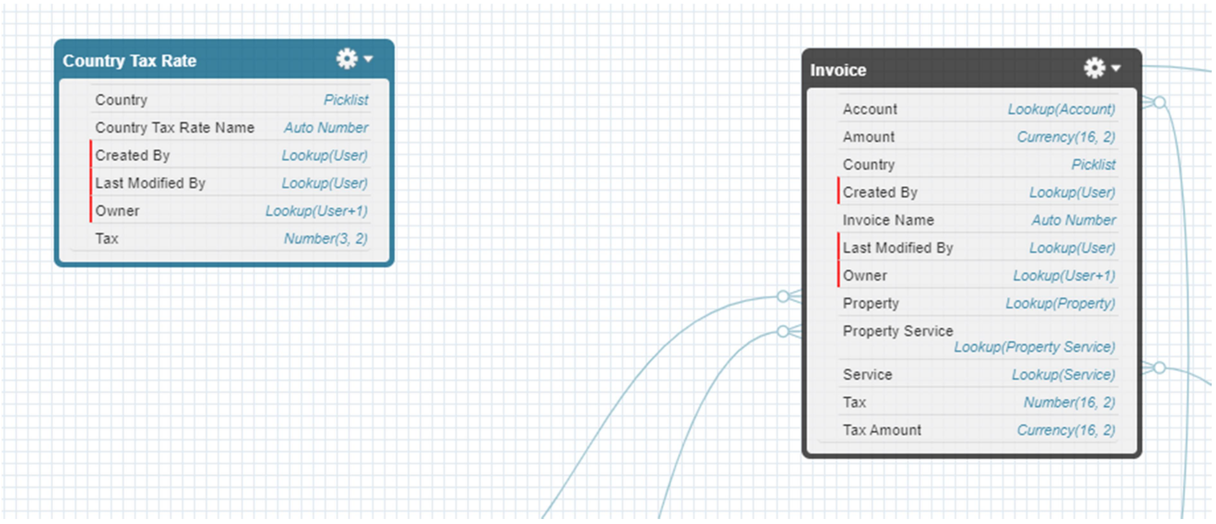
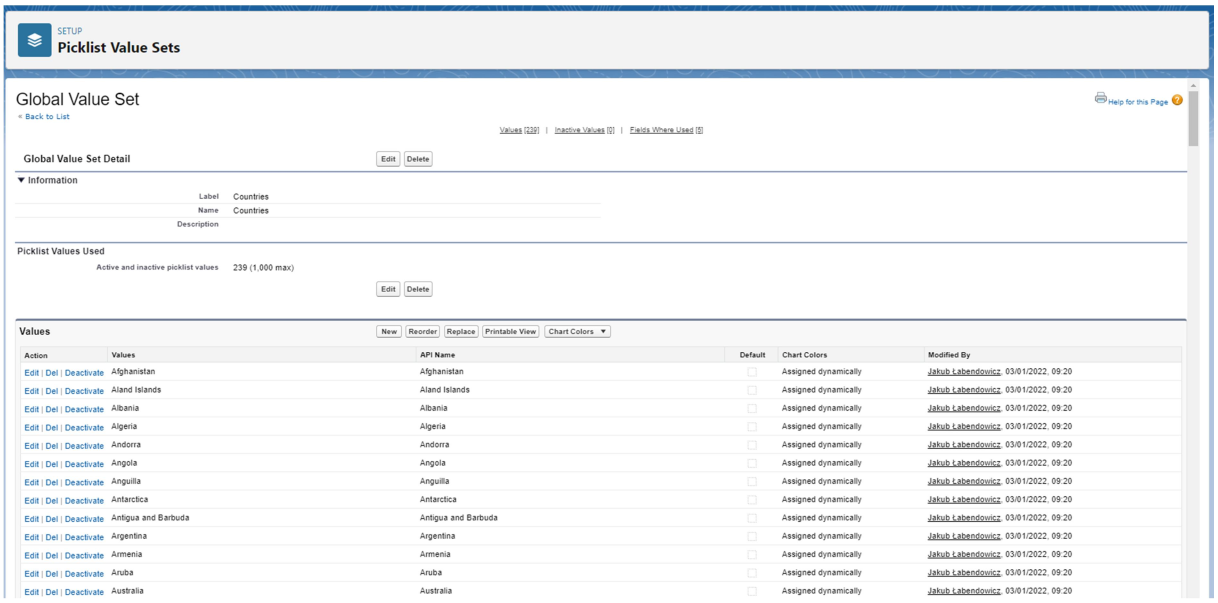
LABORATORIUM 2. ELEMENTY PROGRAMOWANIA W JĘZYKU APEX.

ZADANIE 2.2. WZBOGAĆ MODEL DANYCH




SETUP
Picklist Value Sets

Global Value Set
« Back to List

Values [239] | Inactive Values [0] | Fields Where Used [5]

Global Value Set Detail

▼ Information

Label	Countries
Name	Countries
Description	

Picklist Values Used

Active and inactive picklist values	239 (1,000 max)

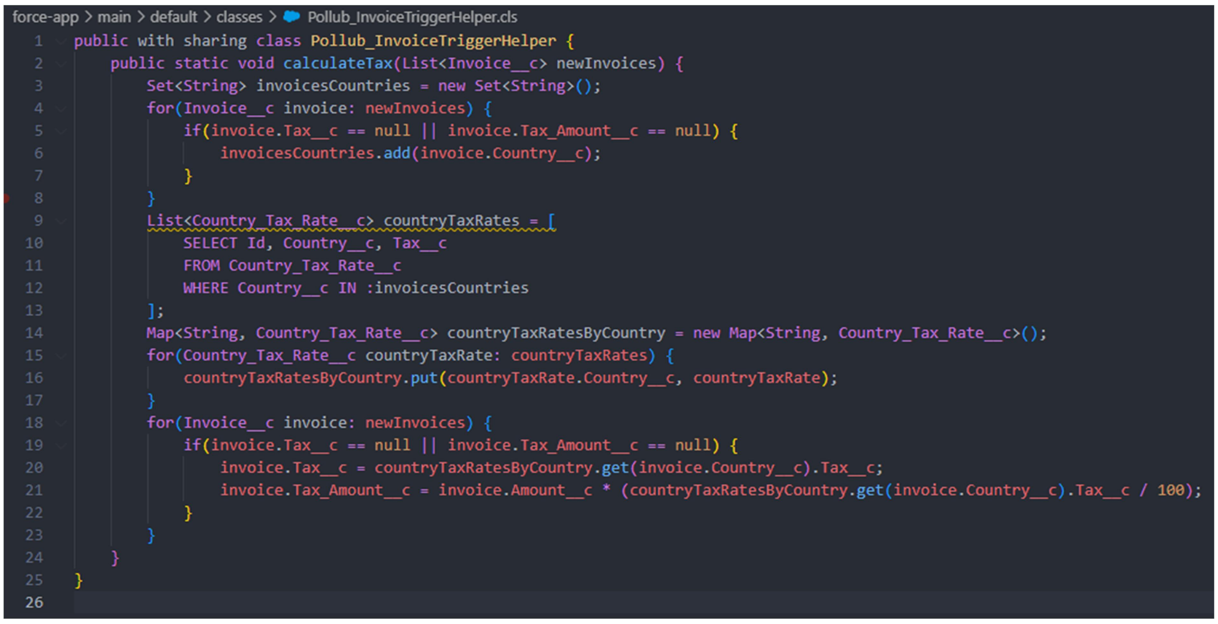
Values

Action	Values	API Name	Default	Chart Colors	Modified By
Edit | Del | Deactivate	Afghanistan	Afghanistan	☐	Assigned dynamically	Jakub Łabendowicz, 03/01/2022, 09:20
Edit | Del | Deactivate	Aland Islands	Aland Islands	☐	Assigned dynamically	Jakub Łabendowicz, 03/01/2022, 09:20
Edit | Del | Deactivate	Albania	Albania	☐	Assigned dynamically	Jakub Łabendowicz, 03/01/2022, 09:20
Edit | Del | Deactivate	Algeria	Algeria	☐	Assigned dynamically	Jakub Łabendowicz, 03/01/2022, 09:20
Edit | Del | Deactivate	Andorra	Andorra	☐	Assigned dynamically	Jakub Łabendowicz, 03/01/2022, 09:20
Edit | Del | Deactivate	Angola	Angola	☐	Assigned dynamically	Jakub Łabendowicz, 03/01/2022, 09:20
Edit | Del | Deactivate	Anguilla	Anguilla	☐	Assigned dynamically	Jakub Łabendowicz, 03/01/2022, 09:20
Edit | Del | Deactivate	Antarctica	Antarctica	☐	Assigned dynamically	Jakub Łabendowicz, 03/01/2022, 09:20
Edit | Del | Deactivate	Antigua and Barbuda	Antigua and Barbuda	☐	Assigned dynamically	Jakub Łabendowicz, 03/01/2022, 09:20
Edit | Del | Deactivate	Argentina	Argentina	☐	Assigned dynamically	Jakub Łabendowicz, 03/01/2022, 09:20
Edit | Del | Deactivate	Armenia	Armenia	☐	Assigned dynamically	Jakub Łabendowicz, 03/01/2022, 09:20
Edit | Del | Deactivate	Aruba	Aruba	☐	Assigned dynamically	Jakub Łabendowicz, 03/01/2022, 09:20
Edit | Del | Deactivate	Australia	Australia	☐	Assigned dynamically	Jakub Łabendowicz, 03/01/2022, 09:20

ZADANIE 2.1. STWORZENIE KLASY WYLICZAJĄCEJ PODATEK,

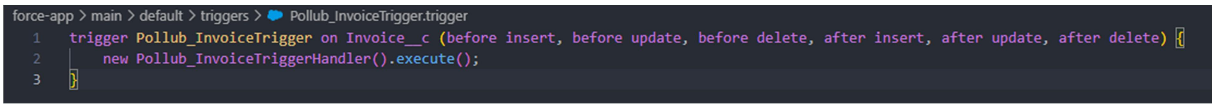
ZADANIE 2.3. ZMODYFIKUJ KLASĘ WYLICZAJĄCĄ PODATEK,

ZADANIE 2.4. PRZETWARZANIE KOLEKCJI DANYCH

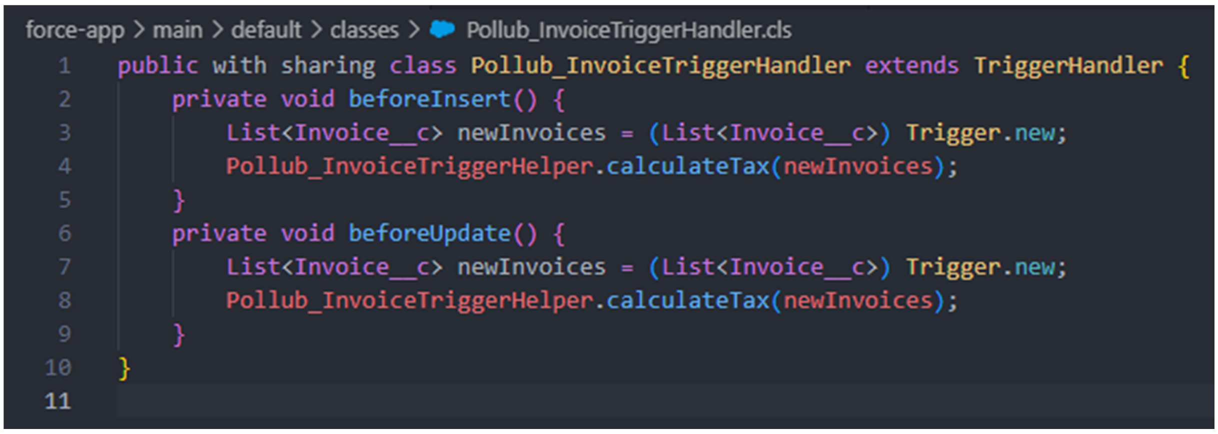

force-app > main > default > classes > Pollub_InvoiceTriggerHelper.cls

```
public with sharing class Pollub_InvoiceTriggerHelper {
    public static void calculateTax(List<Invoice__c> newInvoices) {
        Set<String> invoicesCountries = new Set<String>();
        for(Invoice__c invoice: newInvoices) {
            if(invoice.Tax__c == null || invoice.Tax_Amount__c == null) {
                invoicesCountries.add(invoice.Country__c);
            }
        }
        List<Country_Tax_Rate__c> countryTaxRates = [
            SELECT Id, Country__c, Tax__c
            FROM Country_Tax_Rate__c
            WHERE Country__c IN :invoicesCountries
        ];
        Map<String, Country_Tax_Rate__c> countryTaxRatesByCountry = new Map<String, Country_Tax_Rate__c>();
        for(Country_Tax_Rate__c countryTaxRate: countryTaxRates) {
            countryTaxRatesByCountry.put(countryTaxRate.Country__c, countryTaxRate);
        }
        for(Invoice__c invoice: newInvoices) {
            if(invoice.Tax__c == null || invoice.Tax_Amount__c == null) {
                invoice.Tax__c = countryTaxRatesByCountry.get(invoice.Country__c).Tax__c;
                invoice.Tax_Amount__c = invoice.Amount__c * (countryTaxRatesByCountry.get(invoice.Country__c).Tax__c / 100);
            }
        }
    }
}
```

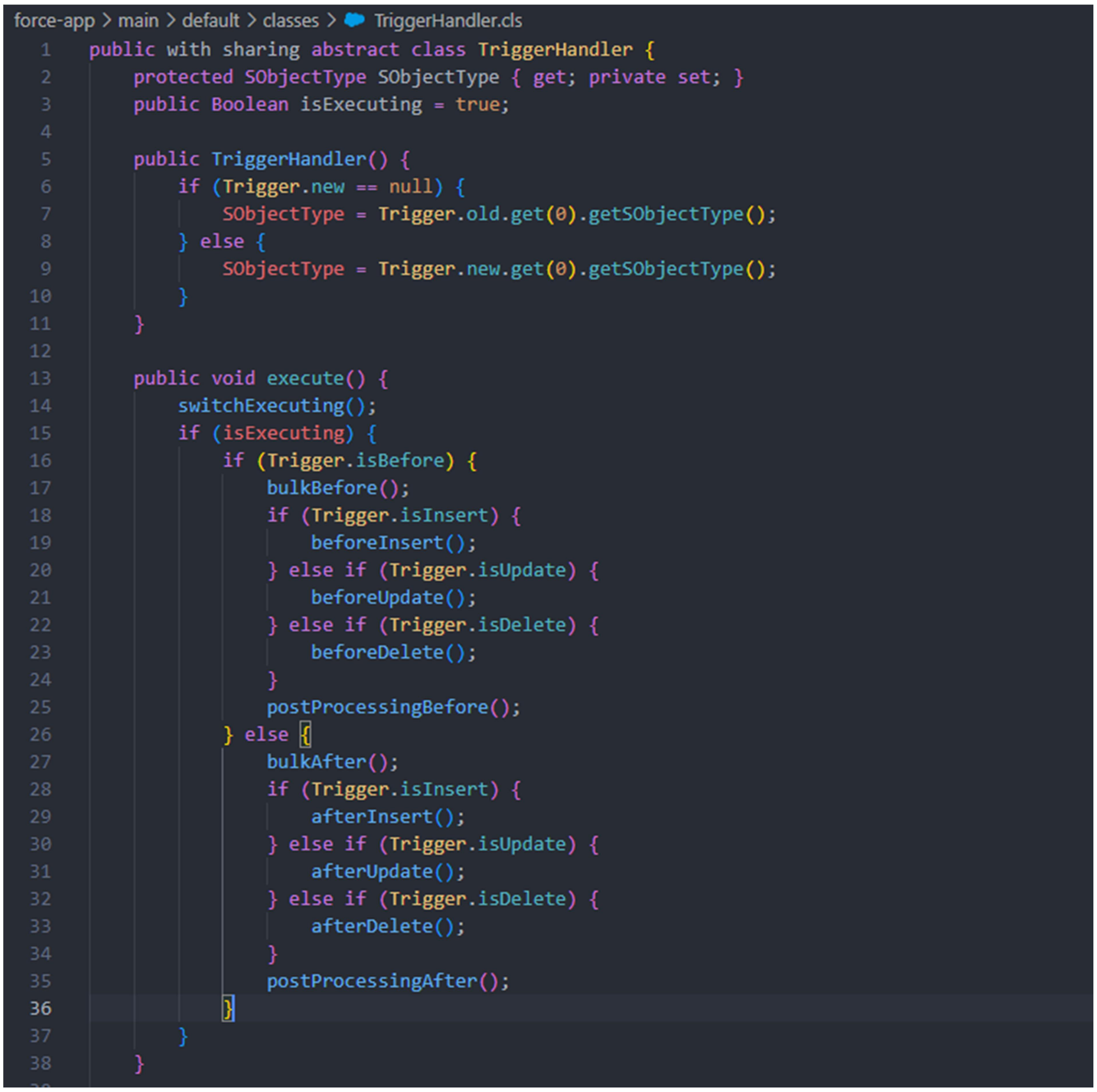
ZADANIE 2.5. WYKONAJ LOGIKĘ KLASY PODCZAS TWORZENIA FAKTURY


force-app > main > default > triggers > Pollub_InvoiceTrigger.trigger

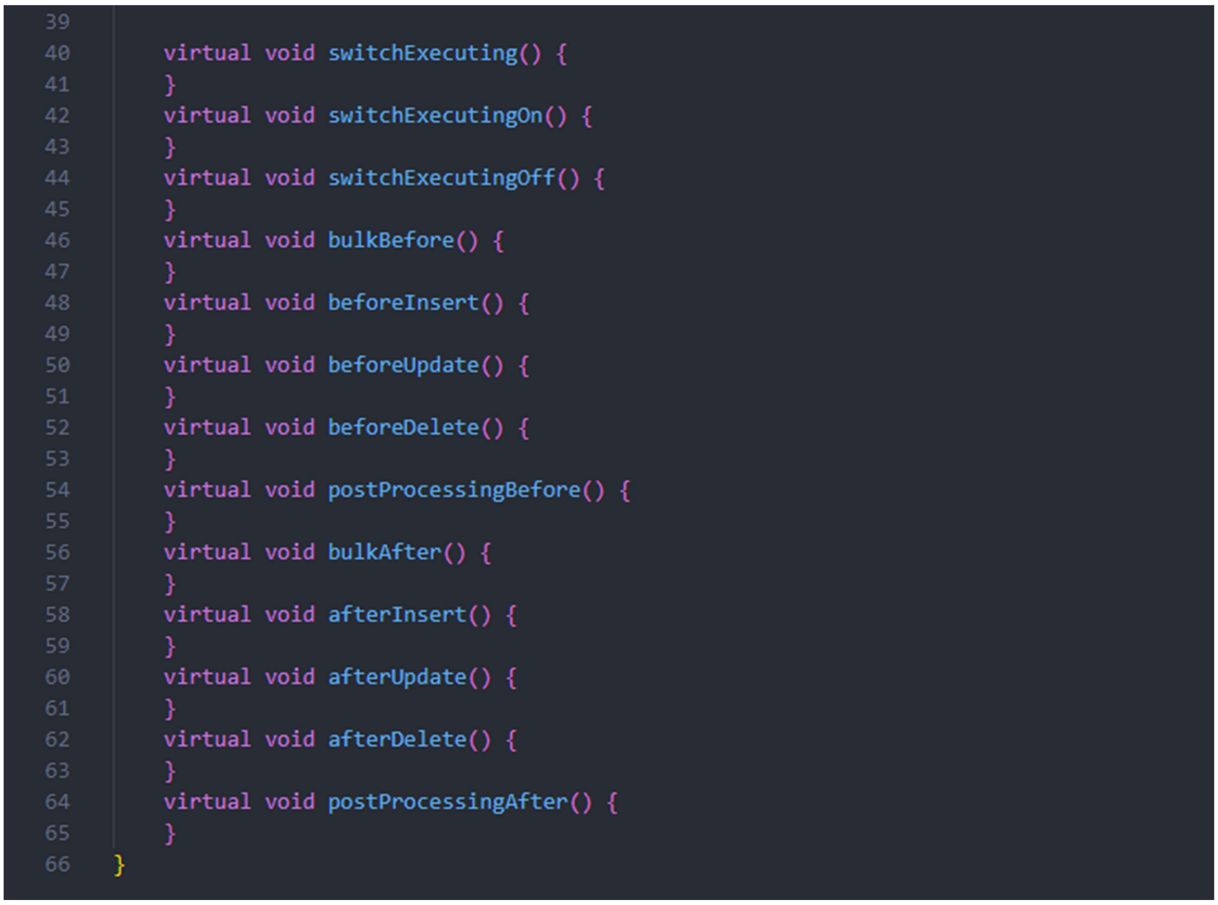
```
trigger Pollub_InvoiceTrigger on Invoice__c (before insert, before update, before delete, after insert, after update, after delete) {
    new Pollub_InvoiceTriggerHandler().execute();
}
```


force-app > main > default > classes > Pollub_InvoiceTriggerHandler.cls

```
public with sharing class Pollub_InvoiceTriggerHandler extends TriggerHandler {
    private void beforeInsert() {
        List<Invoice__c> newInvoices = (List<Invoice__c>) Trigger.new;
        Pollub_InvoiceTriggerHelper.calculateTax(newInvoices);
    }
    private void beforeUpdate() {
        List<Invoice__c> newInvoices = (List<Invoice__c>) Trigger.new;
        Pollub_InvoiceTriggerHelper.calculateTax(newInvoices);
    }
}
```

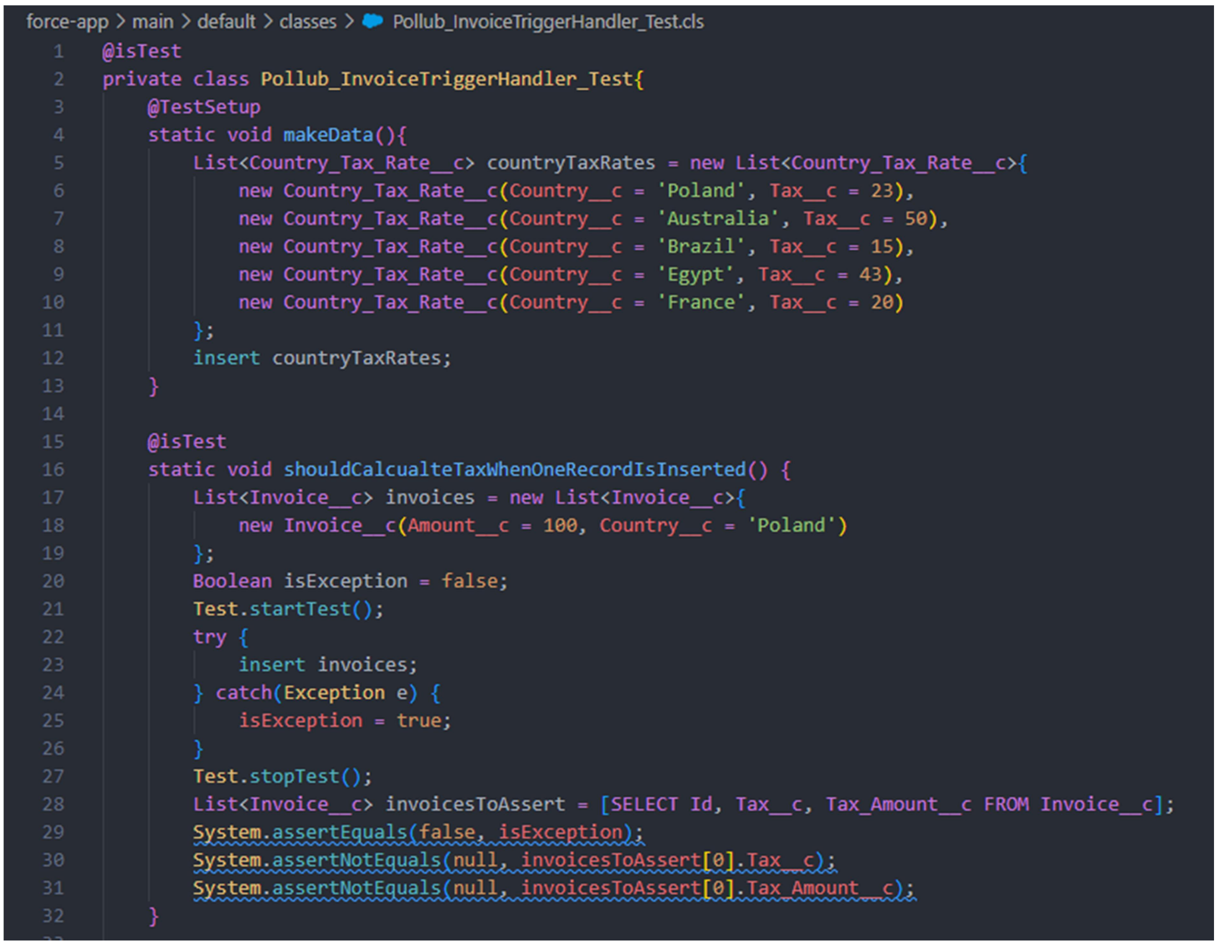

force-app > main > default > classes > TriggerHandler.cls

```
public with sharing abstract class TriggerHandler {
    protected SObjectType SObjectType { get; private set; }
    public Boolean isExecuting = true;

    public TriggerHandler() {
        if (Trigger.new == null) {
            SObjectType = Trigger.old.get(0).getSObjectType();
        } else {
            SObjectType = Trigger.new.get(0).getSObjectType();
        }
    }

    public void execute() {
        switchExecuting();
        if (isExecuting) {
            if (Trigger.isBefore) {
                bulkBefore();
                if (Trigger.isInsert) {
                    beforeInsert();
                } else if (Trigger.isUpdate) {
                    beforeUpdate();
                } else if (Trigger.isDelete) {
                    beforeDelete();
                }
                postProcessingBefore();
            } else {
                bulkAfter();
                if (Trigger.isInsert) {
                    afterInsert();
                } else if (Trigger.isUpdate) {
                    afterUpdate();
                } else if (Trigger.isDelete) {
                    afterDelete();
                }
                postProcessingAfter();
            }
        }
    }
```



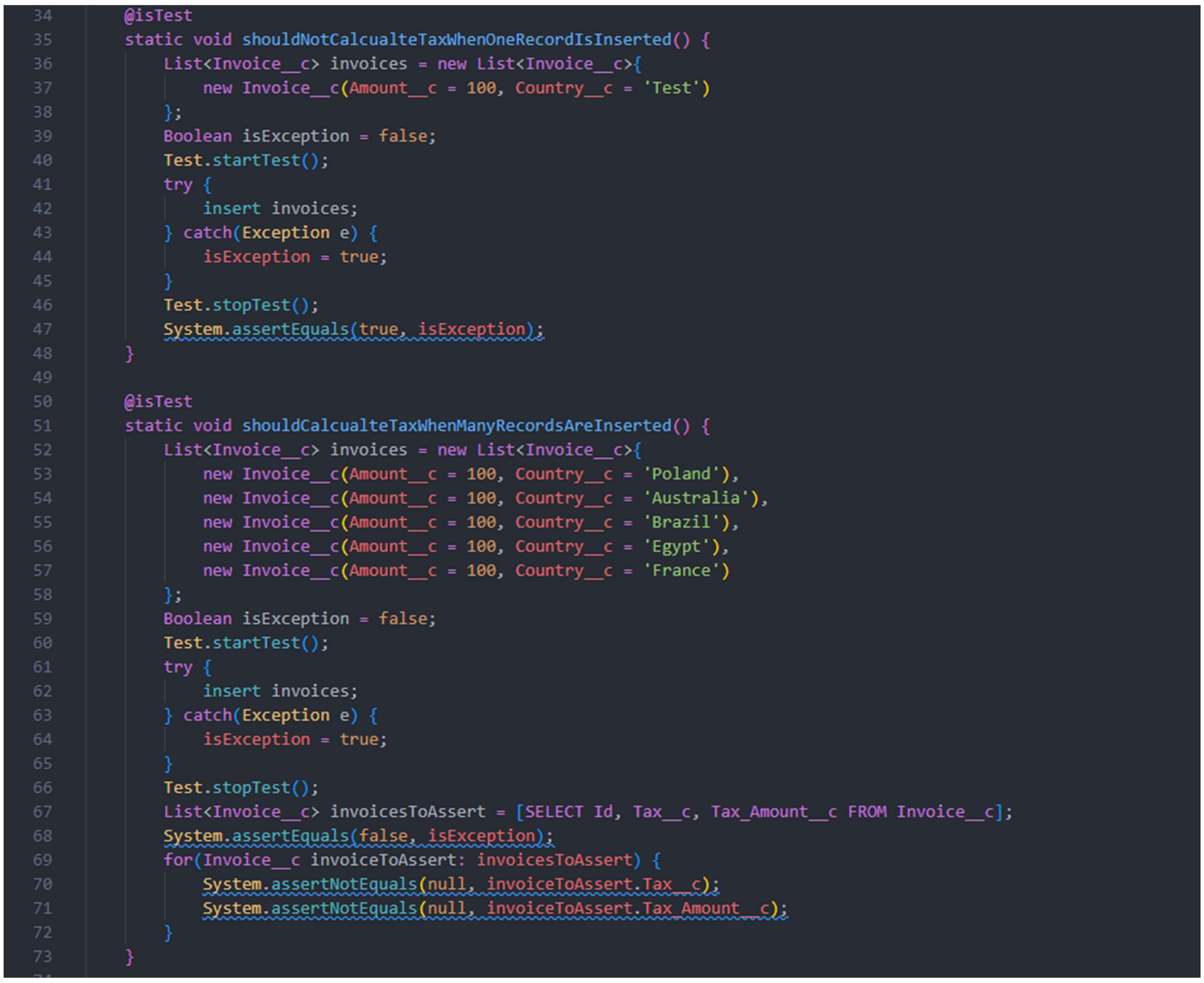
```

    virtual void switchExecuting() {
    }
    virtual void switchExecutingOn() {
    }
    virtual void switchExecutingOff() {
    }
    virtual void bulkBefore() {
    }
    virtual void beforeInsert() {
    }
    virtual void beforeUpdate() {
    }
    virtual void beforeDelete() {
    }
    virtual void postProcessingBefore() {
    }
    virtual void bulkAfter() {
    }
    virtual void afterInsert() {
    }
    virtual void afterUpdate() {
    }
    virtual void afterDelete() {
    }
    virtual void postProcessingAfter() {
    }
}
```

ZADANIE 2.6. STWÓRZ KLASĘ TESTOWĄ

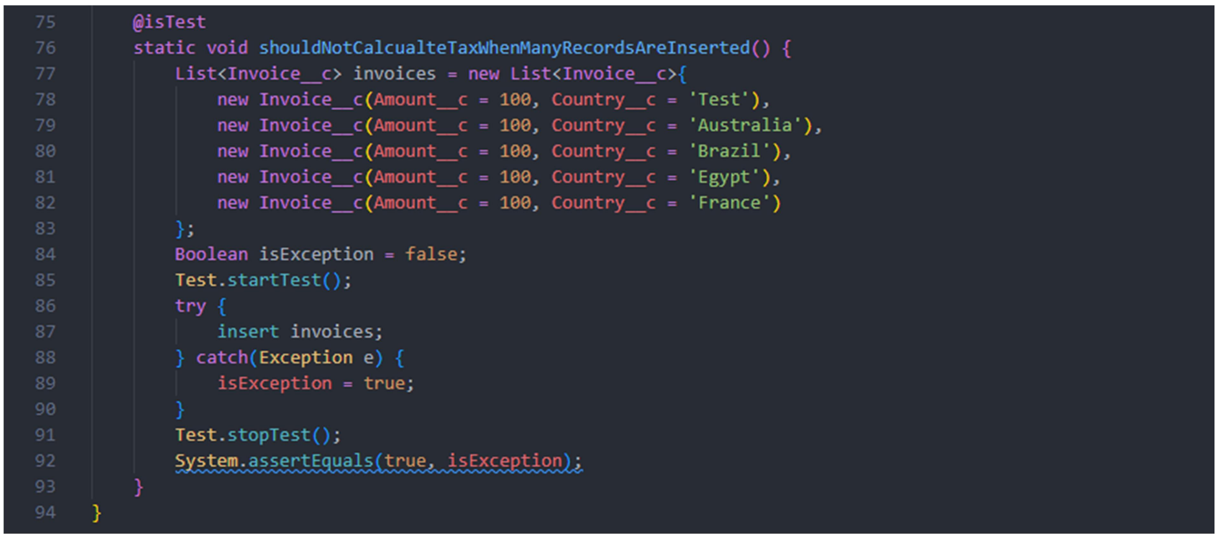


force-app > main > default > classes > Pollub_InvoiceTriggerHandler_Test.cls

```
@isTest
private class Pollub_InvoiceTriggerHandler_Test{
    @TestSetup
    static void makeData(){
        List<Country_Tax_Rate__c> countryTaxRates = new List<Country_Tax_Rate__c>{
            new Country_Tax_Rate__c(Country__c = 'Poland', Tax__c = 23),
            new Country_Tax_Rate__c(Country__c = 'Australia', Tax__c = 50),
            new Country_Tax_Rate__c(Country__c = 'Brazil', Tax__c = 15),
            new Country_Tax_Rate__c(Country__c = 'Egypt', Tax__c = 43),
            new Country_Tax_Rate__c(Country__c = 'France', Tax__c = 20)
        };
        insert countryTaxRates;
    }

    @isTest
    static void shouldCalcualteTaxWhenOneRecordIsInserted() {
        List<Invoice__c> invoices = new List<Invoice__c>{
            new Invoice__c(Amount__c = 100, Country__c = 'Poland')
        };
        Boolean isException = false;
        Test.startTest();
        try {
            insert invoices;
        } catch(Exception e) {
            isException = true;
        }
        Test.stopTest();
        List<Invoice__c> invoicesToAssert = [SELECT Id, Tax__c, Tax_Amount__c FROM Invoice__c];
        System.assertEquals(false, isException);
        System.assertNotEquals(null, invoicesToAssert[0].Tax__c);
        System.assertNotEquals(null, invoicesToAssert[0].Tax_Amount__c);
    }
```



```
    @isTest
    static void shouldNotCalcualteTaxWhenOneRecordIsInserted() {
        List<Invoice__c> invoices = new List<Invoice__c>{
            new Invoice__c(Amount__c = 100, Country__c = 'Test')
        };
        Boolean isException = false;
        Test.startTest();
        try {
            insert invoices;
        } catch(Exception e) {
            isException = true;
        }
        Test.stopTest();
        System.assertEquals(true, isException);
    }

    @isTest
    static void shouldCalcualteTaxWhenManyRecordsAreInserted() {
        List<Invoice__c> invoices = new List<Invoice__c>{
            new Invoice__c(Amount__c = 100, Country__c = 'Poland'),
            new Invoice__c(Amount__c = 100, Country__c = 'Australia'),
            new Invoice__c(Amount__c = 100, Country__c = 'Brazil'),
            new Invoice__c(Amount__c = 100, Country__c = 'Egypt'),
            new Invoice__c(Amount__c = 100, Country__c = 'France')
        };
        Boolean isException = false;
        Test.startTest();
        try {
            insert invoices;
        } catch(Exception e) {
            isException = true;
        }
        Test.stopTest();
        List<Invoice__c> invoicesToAssert = [SELECT Id, Tax__c, Tax_Amount__c FROM Invoice__c];
        System.assertEquals(false, isException);
        for(Invoice__c invoiceToAssert: invoicesToAssert) {
            System.assertNotEquals(null, invoiceToAssert.Tax__c);
            System.assertNotEquals(null, invoiceToAssert.Tax_Amount__c);
        }
    }
```



```
    @isTest
    static void shouldNotCalcualteTaxWhenManyRecordsAreInserted() {
        List<Invoice__c> invoices = new List<Invoice__c>{
            new Invoice__c(Amount__c = 100, Country__c = 'Test'),
            new Invoice__c(Amount__c = 100, Country__c = 'Australia'),
            new Invoice__c(Amount__c = 100, Country__c = 'Brazil'),
            new Invoice__c(Amount__c = 100, Country__c = 'Egypt'),
            new Invoice__c(Amount__c = 100, Country__c = 'France')
        };
        Boolean isException = false;
        Test.startTest();
        try {
            insert invoices;
        } catch(Exception e) {
            isException = true;
        }
        Test.stopTest();
        System.assertEquals(true, isException);
    }
}
```