HW18testCaseJSON

jueves, 2 de febrero de 2023 22:17

FrmSubstract.java

```
:ivate void btnCancelActionPerformed(java.awt.event.ActionEvent
    cleanForm();


:ivate void btnAcceptActionPerformed(java.awt.event.ActionEvent
    Operation addOperation = new Operation( value1: Double.parseDou
            value2: Double.parseDouble( s: txtfSubtrahend.getText())
            expectedValue: Double.parseDouble( s: txtfDifference.getTe

    FileManager.editJson( operation: addOperation,  jsonName: "Subtract
```

FrmDivide.java

```
void txtfDividentActionPerformed(java.awt.event.ActionEvent evt
ODO add your handling code here:


void btnAcceptActionPerformed(java.awt.event.ActionEvent evt) {
ation addOperation = new Operation( value1: Double.parseDouble( s: t
    value2: Double.parseDouble( s: txtfDivisor.getText()),
    expectedValue: Double.parseDouble( s: txtfQuotient.getText()));

Manager.editJson( operation: addOperation,  jsonName: "Divide Test Case
```

Start Page / FrmMultiply.java

```
ed(java.awt.event.ActionEvent evt) {
eration( value1: Double.parseDouble( s: txtfMultiplicand.getText()),
e( s: txtfMultiplier.getText()),
Double( s: txtfProduct.getText()));

dOperation,  jsonName: "Multiply Test Cases.json");


uments
```

FrmAdd.java

```
ceptActionPerformed(java.awt.event.ActionEvent evt) {
peration = new Operation( value1: Double.parseDouble( s: txtfAddent1
Double.parseDouble( s: txtfAddent2.getText()),
Value: Double.parseDouble( s: txtfSum.getText()));

itJson( operation: addOperation,  jsonName: "Add Test Cases.json");


umActionPerformed(java.awt.event.ActionEvent evt) {
ur handling code here:
```

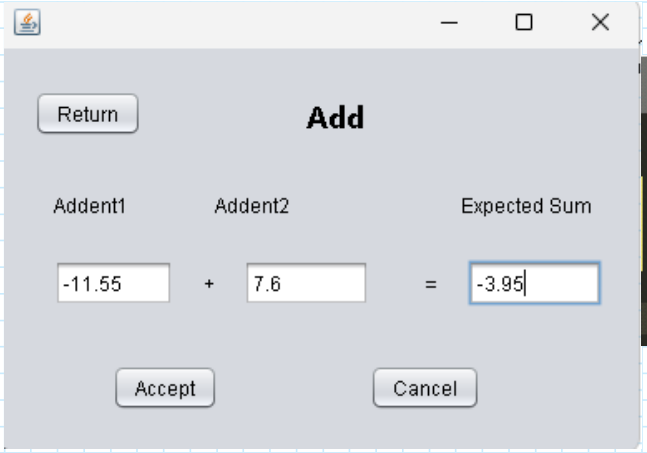
Start Page / BasicOperationTest.java / FileManager.java

```
    private static Gson gson = new GsonBuilder().setPrettyPrinting().create();


    public static void editJson(Operation operation, String jsonName) {
        File fileJson = new File( pathname: jsonName);
        ArrayList<Operation> operations = new ArrayList<>();
        operations = readJson(operations,  fileJson: jsonName);
        operations.add( e: operation);
        String json = gson.toJson( src: operations);

        try ( FileWriter writer = new FileWriter( file: fileJson)) {
            writer.write( str: json);
        } catch (FileNotFoundException e) {
            e.printStackTrace( s: System.out);
        } catch (IOException e) {
            e.printStackTrace( s: System.out);
        }
    }

    public static ArrayList<Operation> readJson(ArrayList<Operation> operations, String fileJson) {
        Gson gson = new Gson();
        File file = new File( pathname: fileJson);
        if (file.length() == 0) {
            try ( FileWriter writer = new FileWriter( fileName: fileJson)) {
                writer.write( str: "[]");
            } catch (IOException ex1) {
                Logger.getLogger( name: FileManager.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex1);
            }
        } else {
            try {
                Reader reader = Files.newBufferedReader( path: Paths.get( first: fileJson));
                TypeToken<ArrayList<Operation>> type = new TypeToken<ArrayList<Operation>>() {
                };
                operations = gson.fromJson( json: reader,  typeOfT: type.getType());
                reader.close();
            } catch (FileNotFoundException ex) {
                System.out.println( x: "File not Found");
            } catch (IOException ex) {
```

Start Page × BasicOperationTest.java × FileManager.java ×

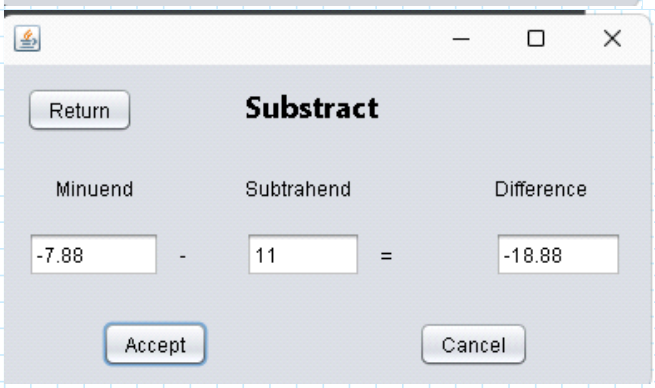
Source History

```java
    public BasicOperationTest() {
    }

    ArrayList<Operation> operations = new ArrayList<>();

    /**
     * Test of add method, of class BasicOperation.
     */
    @Test
    public void testAdd() {
        operations = FileManager.readJson(operations, fileJson:"Add Test Cases.json");
        System.out.println( x:"add");
        for (int i = 0; i < operations.size(); i++) {
            double addend1 = operations.get( index:i).getValue1();
            double addent2 = operations.get( index:i).getValue2();
            double expResult = operations.get( index:i).getExpectedValue();
            double result = BasicOperation.add(addend1, addend2:addent2);
            assertEquals( expected:expResult, actual:result, delta:0);
        }
    }

    /**
     * Test of substract method, of class BasicOperation.
     */
    @Test
    public void testSubstract() {

        operations = FileManager.readJson(operations, fileJson:"Subtract Test Cases.json");
        System.out.println( x:"substract");
        for (int i = 0; i < operations.size(); i++) {
            double minuend = operations.get( index:i).getValue1();
            double subtrahend = operations.get( index:i).getValue2();
            double expResult = operations.get( index:i).getExpectedValue();
            double result = BasicOperation.substract(minuend, subtrahend);
            assertEquals( expected:expResult, actual:result, delta:0);
        }
    }

    /**
     * Test of multiply method, of class BasicOperation.
     */
    @Test
    public void testMultiply() {...12 lines }

    /**
     * Test of divide method, of class BasicOperation.
     */
    @Test
    public void testDivide() {...11 lines }
}
```



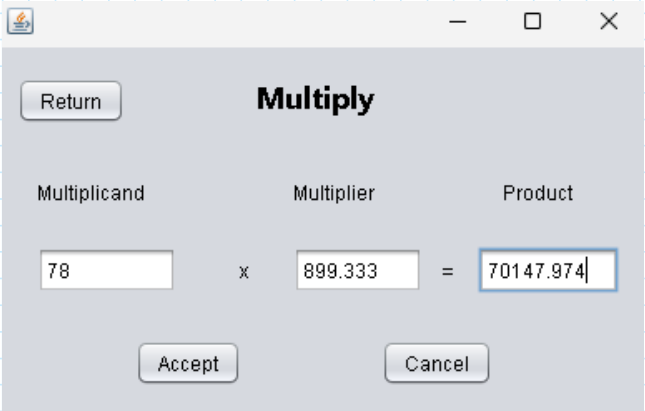
Add Test Cases.json

```
[
  {
    "value1": -11.55,
    "value2": 7.6,
    "expectedValue": -3.95
  }
]
```



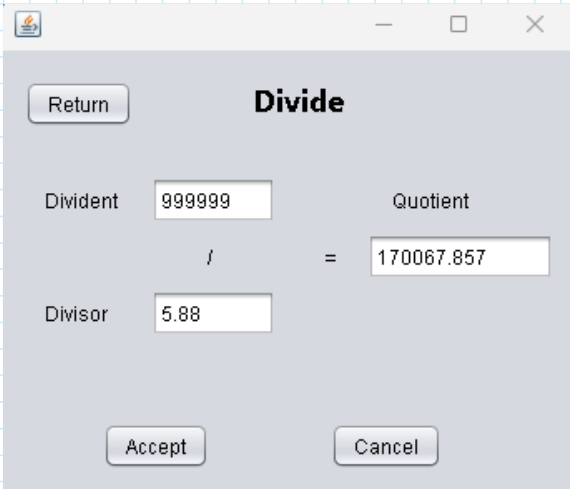
Subtract Test Cases.json

```
[
  {
    "value1": -7.88,
    "value2": 11.0,
    "expectedValue": -18.88
  }
]
```



Multiply Test Cases.json

```
[
  {
    "value1": 78.0,
    "value2": 899.333,
    "expectedValue": 70147.974
  }
]
```




```
[
  {
    "value1": 999999.0,
    "value2": 5.88,
    "expectedValue": 170067.857
  }
]
```

Add Test Cases.json	05/02/2023 21:09
build.xml	31/01/2023 15:03
Divide Test Cases.json	05/02/2023 21:15
manifest.mf	31/01/2023 9:03
Multiply Test Cases.json	05/02/2023 21:12
Subtract Test Cases.json	05/02/2023 21:11

```
26            for (int i = 0; i < operations.size(); i++) {
27                double addend1 = operations.get( index: i).getValue1();
28                double addent2 = operations.get( index: i).getValue2();
29                double expResult = operations.get( index: i).getExpectedValue();
```

est Results × | Output - UnitTestWithAnd (test)

ec.edu.espe.calculator.controller.BasicOperationTest ×

Tests passed: 100,00 %

All 4 tests passed. (0,073 s)

- ec.edu.espe.calculator.controller.BasicOperationTest passed
  - testAdd passed (0,023 s)
  - testSubstract passed (0,0 s)
  - testDivide passed (0,001 s)
  - testMultiply passed (0,001 s)

```
add
substract
divide
multiply
```