Fondamentaux d'Android

7. Tests unitaires

Hatem Aziza - Octobre 2024

Developers

Introduction

Tester votre code peut vous aider à détecter les bogues dès le début du développement. Les tests améliorent la robustesse de votre code. Avec des tests dans votre code, vous pouvez exercer de petites parties de votre application de manière isolée et vous pouvez tester de manière automatisable et reproductible.

Android Studio et la bibliothèque de support de tests Android prennent en charge plusieurs types différents de tests et de frameworks de test.

Dans cet atelier, vous explorez la fonctionnalité de test intégrée d'Android Studio et vous apprenez à écrire et à exécuter des tests unitaires locaux.

Les tests unitaires locaux sont des tests compilés et exécutés entièrement sur votre machine locale avec la machine virtuelle Java (JVM). Vous utilisez des tests unitaires locaux pour tester les parties de votre application qui n'ont pas besoin d'accéder au framework Android ou à un appareil ou émulateur Android.

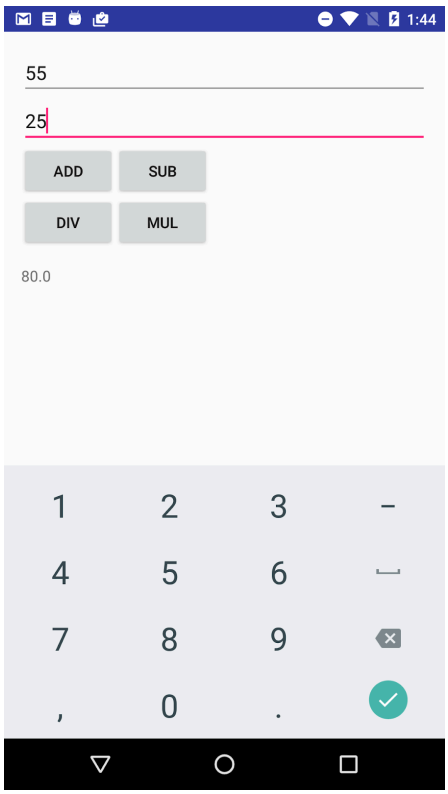
Les tests unitaires sont écrits avec **JUnit**, un framework de tests unitaires commun pour Java.

Ce que vous apprendrez

- Comment organiser et exécuter des tests dans Android Studio.
- Comprenez ce qu'est un test unitaire.
- Écrivez des tests unitaires pour votre code.

Aperçu de l'application

Vous allez développer une application de calcul simple, avec quelques problèmes.



Créer une activité décrite comme suit :
Le layout contient deux **EditText** pour la saisie, quatre **Button** pour les calculs et un **TextView** pour afficher le résultat.
Chaque **Button** possède son propre gestionnaire de clics **android:onClick** (onAdd, OnSub, etc.)
Le TextView résultat ne contient aucun texte par défaut.
Les deux EditText ont l'attribut **android:inputType** et la valeur "**numberDecimal**". Cet attribut indique que l'EditText n'accepte que des nombres en entrée. Le clavier qui apparaît à l'écran ne contiendra que des chiffres.

Créer une classe Java nommée **Calculator** ayant le code suivant:

```
public class Calculator {
    public enum Operator {ADD, SUB, DIV, MUL}

    public double add(double firstOperand, double secondOperand) {
        return firstOperand + secondOperand;
    }

    public double sub(double firstOperand, double secondOperand) {
```

```
        return firstOperand - secondOperand;
    }

    public double div(double firstOperand, double secondOperand) {
        return firstOperand / secondOperand;
    }

    public double mul(double firstOperand, double secondOperand) {
        return firstOperand * secondOperand;
    }
}
```

Créer une activité **MainActivity** ayant le code suivant:

```
public class MainActivity extends Activity {

  private static final String TAG = "CalculatorActivity";
  private Calculator mCalculator;
  private EditText mOperandOneEditText;
  private EditText mOperandTwoEditText;
  private TextView mResultTextView;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // Initialize the calculator class and all the views
    mCalculator = new Calculator();
    mResultTextView = findViewById(R.id.operation_result_text_view);
    mOperandOneEditText = findViewById(R.id.operand_one_edit_text);
    mOperandTwoEditText = findViewById(R.id.operand_two_edit_text);
  }

  public void onAdd(View view) {
    compute(Calculator.Operator.ADD);
  }

  public void onSub(View view) {
    compute(Calculator.Operator.SUB);
  }

  public void onDiv(View view) {
    try {
```

```
            compute(Calculator.Operator.DIV);
        } catch (IllegalArgumentException iae) {
            Log.e(TAG, "IllegalArgumentException", iae);
            mResultTextView.setText(getString(R.string.computationError));
        }
    }

    public void onMul(View view) {
        compute(Calculator.Operator.MUL);
    }

    private void compute(Calculator.Operator operator) {
        double operandOne;
        double operandTwo;
        try {
            operandOne = getOperand(mOperandOneEditText);
            operandTwo = getOperand(mOperandTwoEditText);
        } catch (NumberFormatException nfe) {
            Log.e(TAG, "NumberFormatException", nfe);
            mResultTextView.setText(getString(R.string.computationError));
            return;
        }

        String result;
        switch (operator) {
            case ADD:
                result = String.valueOf(
                        mCalculator.add(operandOne, operandTwo));
                break;
            case SUB:
                result = String.valueOf(
                        mCalculator.sub(operandOne, operandTwo));
                break;
            case DIV:
                result = String.valueOf(
                        mCalculator.div(operandOne, operandTwo));
                break;
            case MUL:
                result = String.valueOf(
                        mCalculator.mul(operandOne, operandTwo));
                break;
            default:
                result = getString(R.string.computationError);
                break;
```

```
        }
        mResultTextView.setText(result);
    }

    private static Double getOperand(EditText operandEditText) {
        String operandText = getOperandText(operandEditText);
        return Double.valueOf(operandText);
    }

    private static String getOperandText(EditText operandEditText) {
        return operandEditText.getText().toString();
    }
}
```

Voici le code de **activity_main.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:paddingBottom="16dp"
        android:paddingLeft="16dp"
        android:paddingRight="16dp"
        android:paddingTop="16dp"
        tools:context=".MainActivity">

    <EditText
        android:id="@+id/operand_one_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/type_operand_one_hint"
        android:inputType="numberDecimal"/>

    <EditText
        android:id="@+id/operand_two_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/type_operand_two_hint"
        android:inputType="numberDecimal"/>

    <RelativeLayout
        android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content">

    <Button
        android:id="@+id/operation_add_btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onAdd"
        android:text="@string/add_operation_text"/>

    <Button
        android:id="@+id/operation_sub_btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toEndOf="@id/operation_add_btn"
        android:layout_toRightOf="@id/operation_add_btn"
        android:onClick="onSub"
        android:text="@string/sub_operation_text"/>

    <Button
        android:id="@+id/operation_div_btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/operation_add_btn"
        android:onClick="onDiv"
        android:text="@string/div_operation_text"/>

    <Button
        android:id="@+id/operation_mul_btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/operation_add_btn"
        android:layout_toEndOf="@id/operation_add_btn"
        android:layout_toRightOf="@id/operation_add_btn"
        android:onClick="onMul"
        android:text="@string/mul_operation_text"/>

</RelativeLayout>

<TextView
    android:id="@+id/operation_result_text_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"/>
```

```
</LinearLayout>
```

Explorer et exécuter CalculatorTest

Vous écrivez et exécutez vos tests dans Android Studio, parallèlement au code de votre application. Chaque nouveau projet Android comprend des exemples de classes de base pour les tests que vous pouvez étendre ou remplacer pour vos propres usages.

1. Créez une classe de tests unitaires de base.

```
import static org.junit.Assert.assertEquals;

import org.junit.Before;
import org.junit.Test;

public class CalculatriceTest {

   private Calculatrice calculator;

   @Before
   public void setUp() {
      calculator = new Calculatrice();
   }

   @Test
   public void testAddition() {
      double result = calculator.add(5, 3);
      assertEquals(8, result, 0.001);
   }

   @Test
   public void testSubtraction() {
      double result = calculator.sub(10, 4);
      assertEquals(6, result, 0.001);
   }

   @Test
   public void testDivision() {
      double result = calculator.div(15, 3);
      assertEquals(5, result, 0.001);
   }

   @Test
```

```
    public void testMultiplication() {
        double result = calculator.mul(4, 7);
        assertEquals(28, result, 0.001);
    }
}
```

Ce code est une classe de **test unitaire** pour la classe **Calculatrice** qui effectue des opérations mathématiques telles que l'addition, la soustraction, la division et la multiplication.

- **private Calculatrice calculator :**
  Cette ligne déclare une variable membre calculator de type Calculatrice. Elle sera utilisée pour effectuer les opérations de test.
- **@Before :**
  Cette annotation marque une méthode **setUp()** qui sera exécutée avant chaque méthode de test. Dans cette méthode, une nouvelle instance de Calculatrice est créée et assignée à la variable calculator. Cela garantit que chaque test commence avec une calculatrice propre.
- **@Test :**
  Cette annotation marque chaque méthode de test. Chaque méthode de test est une fonction qui effectue une opération sur la calculatrice et vérifie si le résultat est correct.
- **public void testAddition() :**
  C'est une méthode de test qui vérifie l'opération d'addition de la calculatrice. Elle exécute calculator.add(5, 3) pour ajouter 5 et 3, puis utilise assertEquals pour vérifier si le résultat est égal à 8 avec une tolérance de 0.001.
- **public void testSubtraction() :**
  C'est une méthode de test similaire à testAddition, mais elle vérifie l'opération de soustraction.
- **public void testDivision() :**
  Cette méthode de test vérifie l'opération de division de la calculatrice.
- **public void testMultiplication() :**
  Cette méthode de test vérifie l'opération de multiplication de la calculatrice.

L'objectif de ce code est de vérifier que les opérations de la calculatrice fonctionnent correctement en utilisant des méthodes de test dédiées pour chaque opération. Si les résultats correspondent aux valeurs attendues avec une précision de 0.001, les tests réussiront. Sinon, ils signaleront des erreurs.

2. Exécutez les tests unitaires dans le dossier de tests et afficherez le résultat des tests réussis et échoués.