View Models: przywracanie stanu aplikacji po obrocie ekranu / powrocie z tła

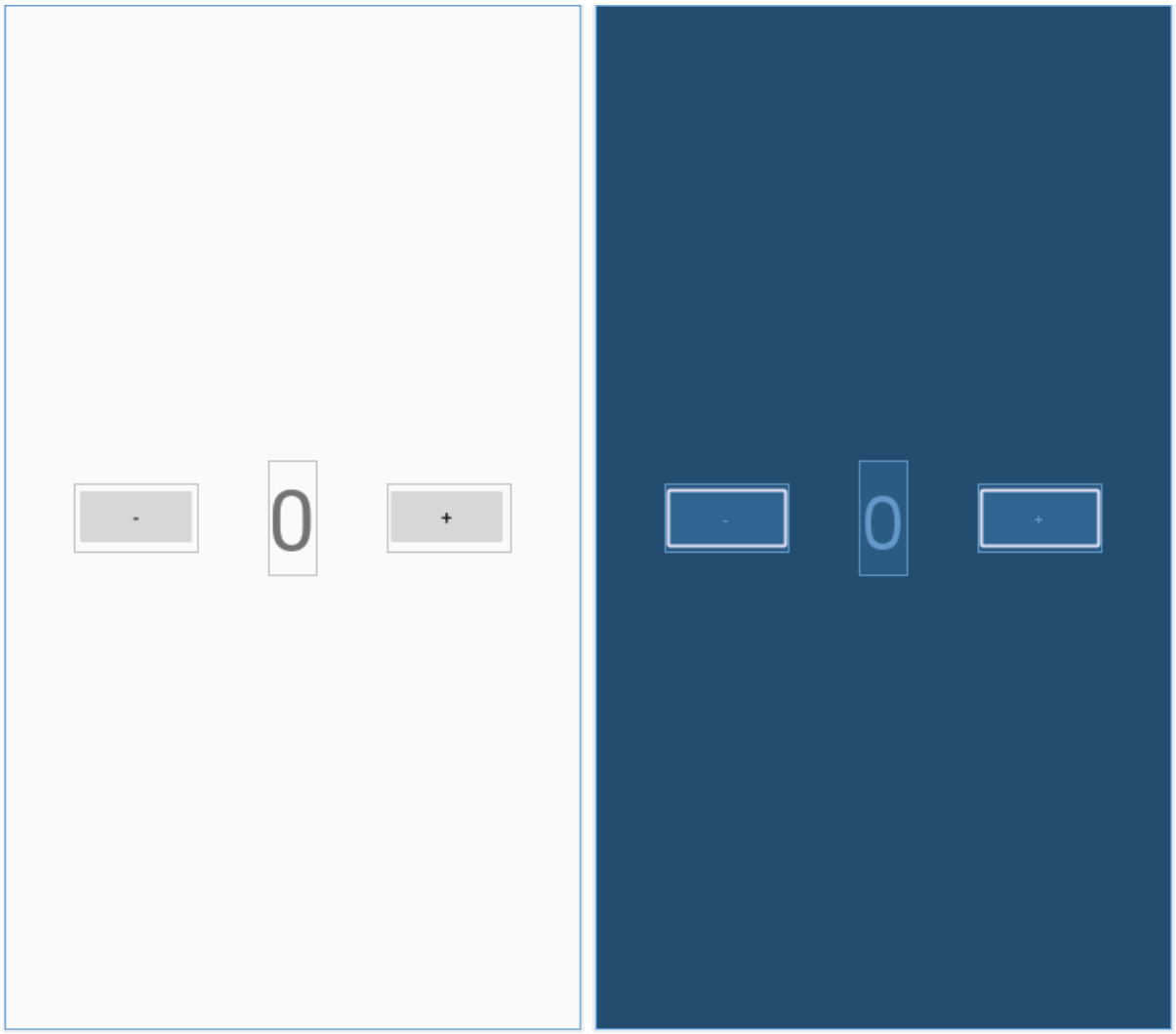
Klasa ViewModel służy do przechowywania i zarządzania danymi związanymi z interfejsem użytkownika. Dzięki niej nie utracimy naszych danych po obrocie ekranu czy powrocie z tła.

Oto prosty przykład, jak krok po kroku wykorzystać jej możliwości.

1. Wygeneruj swój projekt.

```
package prezentacja.viewmodels;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

}
```

2. Stwórz przykładowy layout.



```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/counter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:textAlignment="center"
        android:textSize="60sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/minus"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:text="-"
        app:layout_constraintBottom_toBottomOf="@+id/counter"
        app:layout_constraintEnd_toStartOf="@+id/counter"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/counter" />

    <Button
        android:id="@+id/plus"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:text="+"
        app:layout_constraintBottom_toBottomOf="@+id/counter"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/counter"
        app:layout_constraintTop_toTopOf="@+id/counter" />

</android.support.constraint.ConstraintLayout>
```

3. Stwórz klasę z danymi, które chcesz przechowywać.

```
package prezentacja.viewmodels;

import android.arch.lifecycle.ViewModel;

public class CounterViewModel extends ViewModel {

    private int counter;

    public int getCounter() {
        return counter;
    }

    public void setCounter(int counter) {
        this.counter = counter;
    }

}
```

4. Przypisz elementy layoutu do zmiennych.

```
public class MainActivity extends AppCompatActivity {

    private Button plus;
    private Button minus;
    private TextView counter;

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

        plus = findViewById(R.id.plus);
        minus = findViewById(R.id.minus);
        counter = findViewById(R.id.counter);
    }

}
```

5. Utwórz w klasie pole CounterViewModel.

```
private CounterViewModel mViewModel;
```

6. W pliku build.grandle dodaj dwie zależności.

```
dependencies {
  (...)
  implementation 'android.arch.lifecycle:extensions:1.1.0'
  implementation 'android.arch.lifecycle:viewmodel:1.1.0'
  (...)
}
```

7. W metodzie onCreate() zainicjalizuj zmienną w ten sposób.

```
mViewModel = ViewModelProviders.of(this).get(CounterViewModel.class);
```

8. Teraz już odwoływać się do swoich zmiennych za pomocą metod, które utworzyłeś w swojej klasie.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    (…)
    setCounter(mViewModel.getCounter());

    plus.setOnClickListener(v -> increaseCounter());
    minus.setOnClickListener(v -> decreaseCounter());
}

private void increaseCounter(){
    mViewModel.setCounter(mViewModel.getCounter() + 1);
    counter.setText(mViewModel.getCounter() + "");
}

private void decreaseCounter(){
    mViewModel.setCounter(mViewModel.getCounter() - 1);
    counter.setText(mViewModel.getCounter() + "");
}

private void setCounter(int number){
    counter.setText(number + "");
}
```

Cały kod źródłowy można znaleźć pod tym adresem: https://github.com/skamila/ViewModels