У Android есть особенность — при смене конфигурации (например, при повороте экрана) Activity будет уничтожена и создана заново.

Давайте посмотрим на конкретном примере. Создайте новый проект.

Теряем данные при повороте экрана

Суть приложения до безумия проста: в Activity есть переменная типа int, которую мы увеличиваем на 1 при нажатии на кнопку.

Рядом есть TextView, в нём отображается значение, которое мы хранится в переменной.

Лэйаут MainActivity будет таким:

<?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:gravity="center"

android:orientation="vertical"

tools:context="com.skillberg.rotation.MainActivity">

<TextView

android:id="@+id/counter\_tv"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

<Button

android:id="@+id/counter\_btn"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="16dp"

android:text="@string/title\_increment" />

</LinearLayout>

Код MainActivity:

public class MainActivity extends AppCompatActivity {

private TextView counterTv;

private int currentNumber = 0;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

counterTv = findViewById(R.id.counter\_tv);

updateCounter();

findViewById(R.id.counter\_btn).setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

currentNumber++;

updateCounter();

}

});

}

/\*\*

\* Обновляет счётчик в соответствиии с currentNumber

\*/

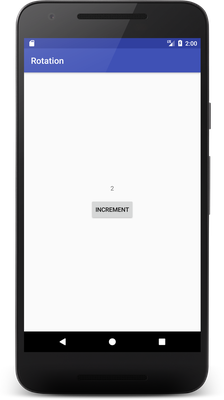
private void updateCounter() {

counterTv.setText(String.valueOf(currentNumber));

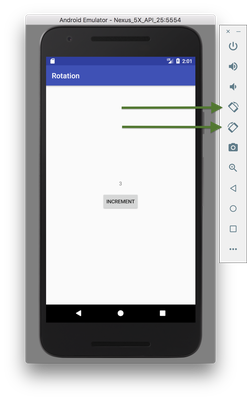
}

}

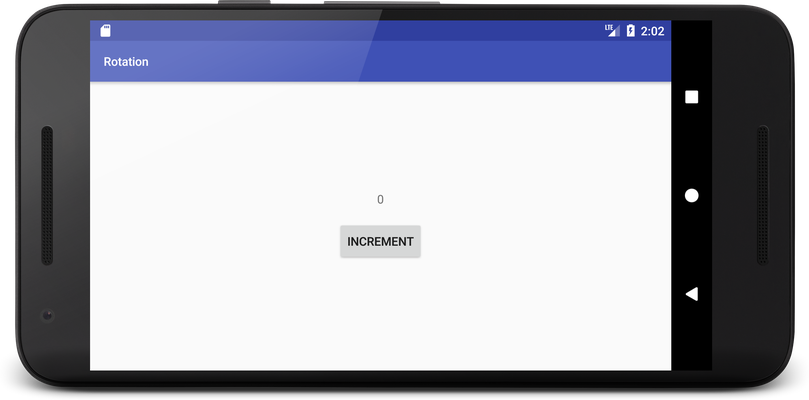
Запустите приложение и понажимайте на кнопку, чтобы счётчик увеличился:

[](https://skillberg.com/media/uploads/2017/11/17/5d603b653fd24c6cb1741fb93c697c34.png)Счётчик в портертной ориентации

А теперь поверните экран. В эмуляторе это можно сделать, нажав на одну из этих кнопок:

[](https://skillberg.com/media/uploads/2017/11/17/505ddf431c574b0cbfaf2215ae4770e6.png)Поворот экрана в эмуляторе

Итак, после поворота вы увидите это:

[](https://skillberg.com/media/uploads/2017/11/17/1ee967530e164892a2a62acd7fb882f9.png)Счётчик в ландшафтной ориентации

Упс! Счётчик сбросился.

Это произошло потому что Activity пересоздалась и, соответственно, значение переменной currentNumberсбросилось.

Что же делать?

Метод onSaveInstanceState()

У Activity есть метод onSaveInstanceState(). Перед тем, как Activity уничтожится для пересоздания, система вызовет этот метод, чтобы дать нам возможность сохранить данные.

**Обратите внимание:** этот метод будет вызван только в случае пересоздания Activity. При безвозвратном уничтожении (например, если пользователь нажал на кнопку **назад**) этот метод вызван не будет.

У метода onSaveInstanceState() есть параметр Bundle outState. Именно в него мы и должны сохранить те данные, которые нужно будет восстановить позже.

В нашем примере нам нужно будет сохранить лишь одно поле:

@Override

protected void onSaveInstanceState(Bundle outState) {

outState.putInt(KEY\_NUMBER, currentNumber);

super.onSaveInstanceState(outState);

}

Константа KEY\_NUMBER:

public class MainActivity extends AppCompatActivity {

private static final String KEY\_NUMBER = "number";

**Обратите внимание:** нужно обязательно вызвать метод суперкласса после сохранения данных, чтобы Activity могла самостоятельно сохранить некоторые другие данные.

Какие "другие данные"?

Дело в том, что этот механизм автоматически сохраняет такие вещи, как, например, содержимое EditText. Единственное условие — у этих View должны быть заданы идентификаторы.

Это серьёзно упрощает жизнь, так как вам не нужно выполнять лишнюю работу.

Восстановление из savedInstanceState.

Все те данные, которые вы сохранили в onSaveInstanceState() будут переданы обратно после пересоздания Activity.

Есть два варианта восстановления данных:

1. В методе onRestoreInstanceState().
2. В методе onCreate().

Разница между ними несущественна: onRestoreInstanceState() будет вызван после вызова метода onStart(), то есть существенно позже onCreate(). В целом — это вся разница.

Зачем тогда два метода?

Представьте, что у вас есть **Activity**, наследующаяся от другой вашей же **Activity**. У каждой из этих Activityможет быть свой onCreate(), тогда может не получиться реализовать восстановление один раз для обоих классов, либо наоборот, не получится реализовать раздельное восстановление. В таких ситуациях рекомендуется использовать onRestoreInstanceState().

В остальных случаях лучше восстанавливать состояние в onCreate().

Как вы могли заметить, в onCreate() передаётся параметр Bundle savedInstanceState. Именно тут будут лежать сохранённые данные.

Восстановление будет невероятно простым:

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

counterTv = findViewById(R.id.counter\_tv);

if (savedInstanceState != null) {

currentNumber = savedInstanceState.getInt(KEY\_NUMBER, 0);

}

updateCounter();

findViewById(R.id.counter\_btn).setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// ...

}

});

}

Как видите, я добавил всего лишь три строчки кода и восстановил данные.

Восстановление объектов с использованием Parcelable.

А что, если нам нужно сохранить не какой-то примитивный тип, а сложный объект? Давайте создадим класс Counter и добавим в него пару дополнительных полей (которые по факту не нужны, но будут присутствовать для примера):

public class Counter {

private int currentNumber = 0;

private final boolean someBoolean;

private final String someString;

public Counter() {

SecureRandom secureRandom = new SecureRandom();

someBoolean = secureRandom.nextBoolean();

someString = "Test" + secureRandom.nextInt();

}

public int getCurrentNumber() {

return currentNumber;

}

public boolean isSomeBoolean() {

return someBoolean;

}

public String getSomeString() {

return someString;

}

public void incrementCounter() {

currentNumber++;

}

@Override

public String toString() {

return "Counter{" +

"currentNumber=" + currentNumber +

", someBoolean=" + someBoolean +

", someString='" + someString + '\'' +

'}';

}

}

Два поля, добавленных для примера, генерируются случайным образом.

Теперь в MainActivity заменим "цифровой" счётчик на этот:

public class MainActivity extends AppCompatActivity {

private static final String KEY\_NUMBER = "number";

private TextView counterTv;

private final Counter counter = new Counter();

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

counterTv = findViewById(R.id.counter\_tv);

updateCounter();

findViewById(R.id.counter\_btn).setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

counter.incrementCounter();

updateCounter();

}

});

}

/\*\*

\* Обновляет счётчик в соответствиии с currentNumber

\*/

private void updateCounter() {

counterTv.setText(String.valueOf(counter.getCurrentNumber()));

}

@Override

protected void onRestoreInstanceState(Bundle savedInstanceState) {

super.onRestoreInstanceState(savedInstanceState);

}

@Override

protected void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

}

}

Собственно, проблема в том, что мы не можем вот так просто передать произвольный объект в Bundle. У нас есть три варианта:

1. Сохранять каждое поле по отдельности. Такой себе вариант.
2. Реализовать интерфейс Serializable. Неплохой вариант, "встроенный" в Java, но он не очень хорошо подходит для мобильных платформ из-за низкой скорости.
3. Реализовать интерфейс Parcelable. Это идеальный вариант, отлично работающий на Android.

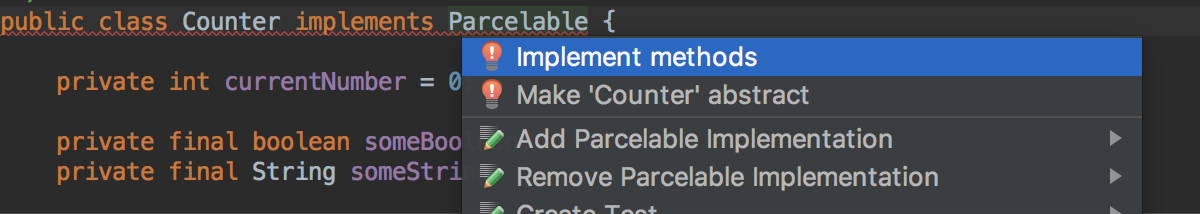
Объект класса, реализовавшего интерфейс Parcelable, может быть сериализован в специальный объект класса Parcel. При этом каждое нужное нам поле будет сохранено в компактном виде, и потом может быть с лёгкостью восстановлено.

В современных версиях Android Studio реализовать интерфейс Parcelable очень легко — просто добавьте его к классу:

public class Counter implements Parcelable {

// ...

Затем нажмите Alt + Enter и выберите **Implement Methods**:

[](https://skillberg.com/media/uploads/2017/11/17/4c5a1828e8a4473aa66ed673da3d324e.png)

После этого в класс будут добавлены два метода:

@Override

public int describeContents() {

return 0;

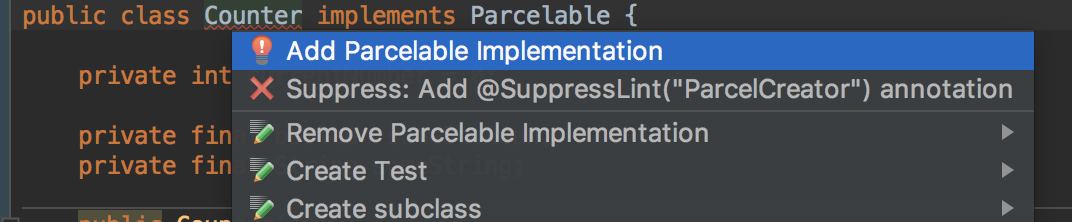
}

@Override

public void writeToParcel(Parcel dest, int flags) {

}

Теперь поставьте курсор в класс, нажмите Alt + Enter ещё раз и выберите **Add Parcelable Implementation**:

[](https://skillberg.com/media/uploads/2017/11/17/f50b7dbbb26148baaa2e0d5109ee004b.png)

После этого добавится новый конструктор:

protected Counter(Parcel in) {

currentNumber = in.readInt();

someBoolean = in.readByte() != 0;

someString = in.readString();

}

Метод writeToParcel() будет реализован автоматически:

@Override

public void writeToParcel(Parcel dest, int flags) {

dest.writeInt(currentNumber);

dest.writeByte((byte) (someBoolean ? 1 : 0));

dest.writeString(someString);

}

И создастся новый объект CREATOR:

public static final Creator<Counter> CREATOR = new Creator<Counter>() {

@Override

public Counter createFromParcel(Parcel in) {

return new Counter(in);

}

@Override

public Counter[] newArray(int size) {

return new Counter[size];

}

};

Как видите, мы просто по очереди записываем в Parcel нужные нам поля, а при извлечении читаем их оттуда в том же порядке, что и при записи. Всё элементарно, и Android Studio сделает всё за нас!

**Обратите внимание:** не передавайте большие объекты (например, с изображениями) в Parcelable.

**Обратите внимание:** используйте Parcelable только для временной передачи данных, например, при пересоздании Activity. Нельзя хранить данные в Parcelable в течение долгого времени или передавать его между разными устройствами, так как реализации этого класса могут отличаться.

Вернёмся к MainActivity.

Метод onSaveInstanceState() теперь выглядит так:

@Override

protected void onSaveInstanceState(Bundle outState) {

outState.putParcelable(KEY\_NUMBER, counter);

super.onSaveInstanceState(outState);

}

А восстановление будет выглядеть так:

public class MainActivity extends AppCompatActivity {

// ...

private Counter counter;

@Override

protected void onCreate(Bundle savedInstanceState) {

// ...

counterTv = findViewById(R.id.counter\_tv);

if (savedInstanceState != null) {

counter = savedInstanceState.getParcelable(KEY\_NUMBER);

} else {

counter = new Counter();

}

updateCounter();

// ..

}

Запустите приложение и убедитесь, что всё работает! :)

Как быть с фрагментами?

У фрагментов сохранение и восстановление данных работает точно так же — проверьте сами!

ДОМАШНЕЕ ЗАДАНИЕ

Реализуйте то же самое, включая обработку поворота экрана, используя фрагменты.