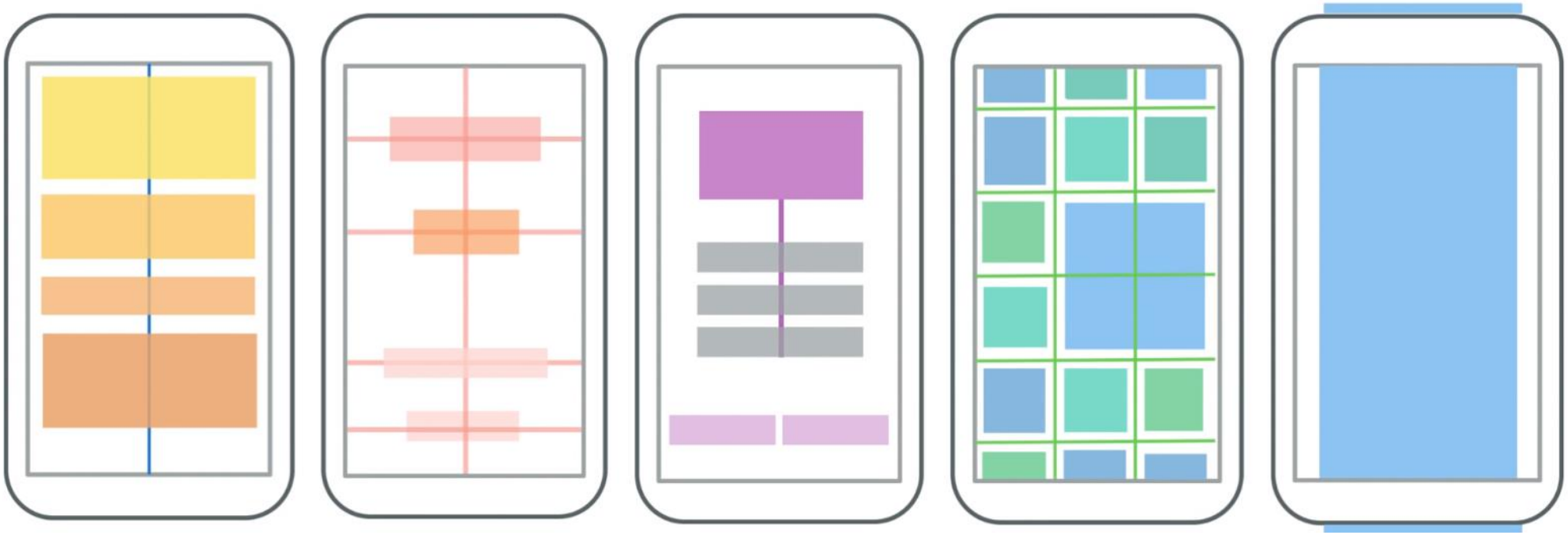
# FRAGMENTY

*dr Artur Bartoszewski*

# Fragmenty



# Fragmenty – trochę teorii

Fragmenty (ang. Fragments) w Androidzie to modułowe, samodzielne komponenty interfejsu użytkownika, które można łatwo integrować i zarządzać w ramach aktywności. Są to części aplikacji, które mogą zawierać zarówno widoki (UI), jak i logikę, i mogą być ponownie wykorzystywane w różnych miejscach aplikacji.

Fragmenty umożliwiają podzielenie interfejsu użytkownika na mniejsze, niezależne moduły, które można ponownie wykorzystać w różnych aktywnościach lub w ramach jednej aktywności. Dzięki temu można tworzyć bardziej elastyczne i skalowalne interfejsy.

Fragmenty mogą być dynamicznie dodawane, usuwane i zastępowane w czasie działania aplikacji, co jest przydatne w aplikacjach złożonych, gdzie interfejs zmienia się w zależności od akcji użytkownika. Dzięki fragmentom można łatwo realizować nawigację pomiędzy różnymi sekcjami interfejsu.

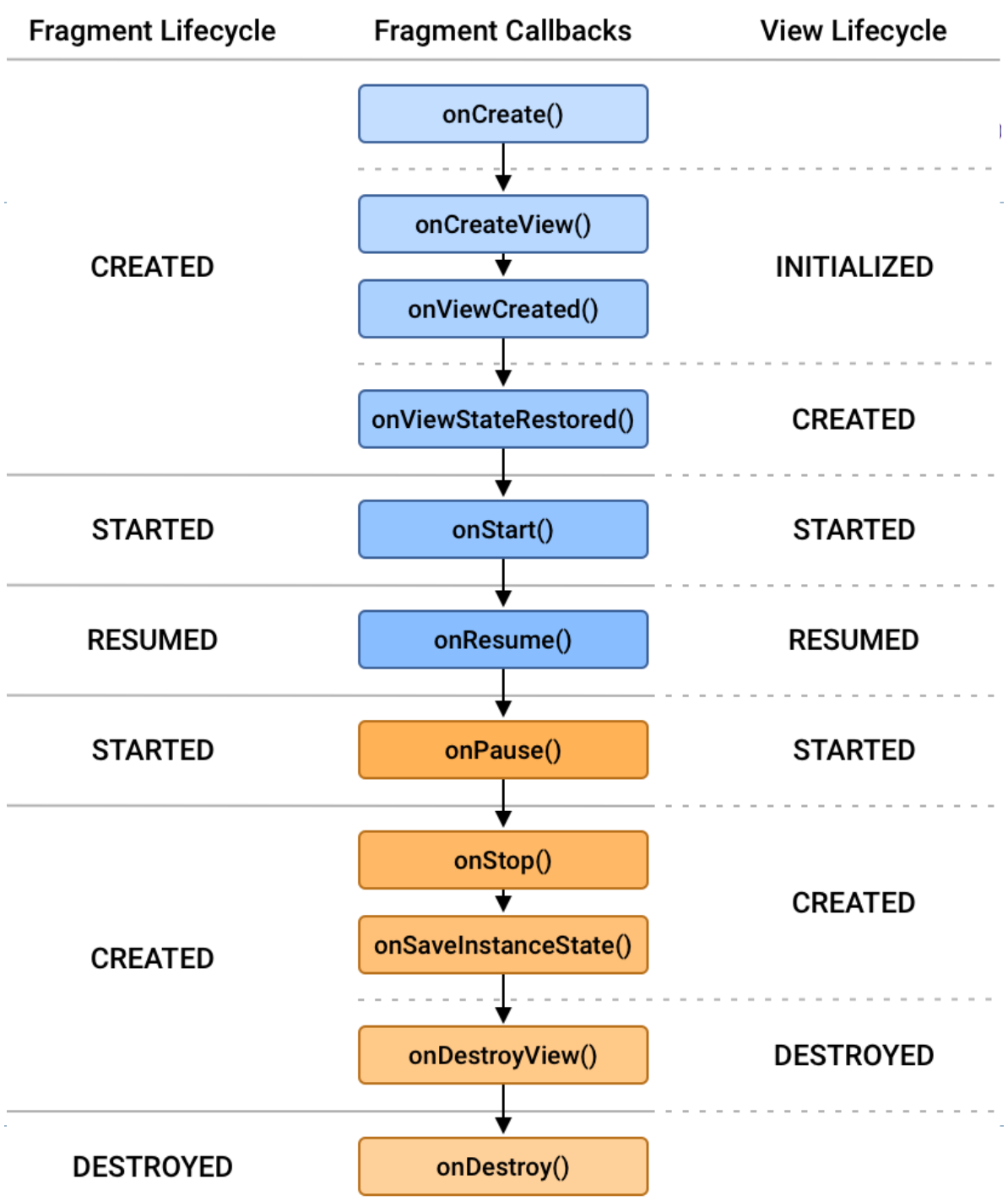
# Fragmenty – trochę teorii

Fragmenty mają własny cykl życia, który jest powiązany z cyklem życia aktywności.

W odróżnieniu od aktywności utworzenie fragmentu nie jest jednoznaczne z jego wyświetleniem.

Fragment może być utworzony w pamięci i dopiero później dodany do aktywności.

Widoki (kontrolki opisane w pliku XML) dodawane są do fragmentu dopiero w momencie jego wyświetlenie. Dlatego nie wolno odwoływać się do nich w metodzie onCreate(), a dopiero w metodach onCreateView() i onViewCreated().

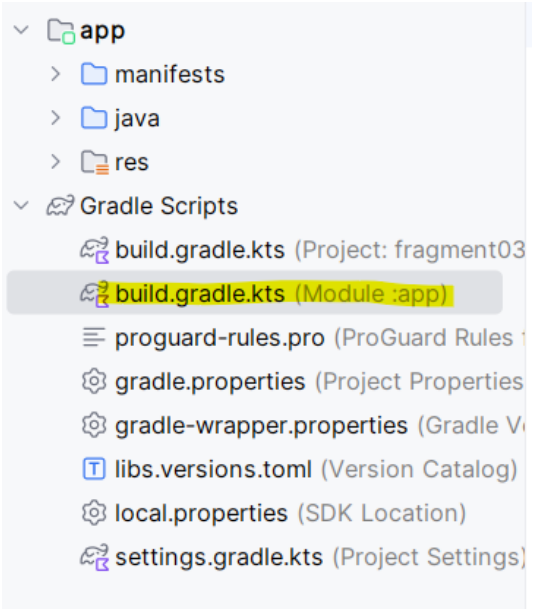


# Konfigurowanie środowiska

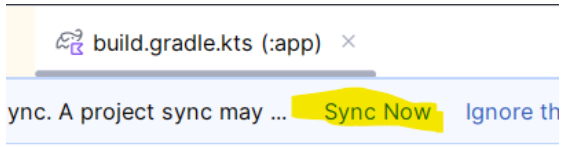
Fragmenty wymagają zależności od biblioteki fragmentów AndroidX. Aby ją dołączyć do projektu, dodaj następujące elementy zależności w pliku aplikacji:build.gradle:

```
dependencies {
    val fragment_version = "1.8.2"

    // Java language implementation
    implementation("androidx.fragment:fragment:$fragment_version")
    // Kotlin
    implementation("androidx.fragment:fragment-ktx:$fragment_version")
}
```



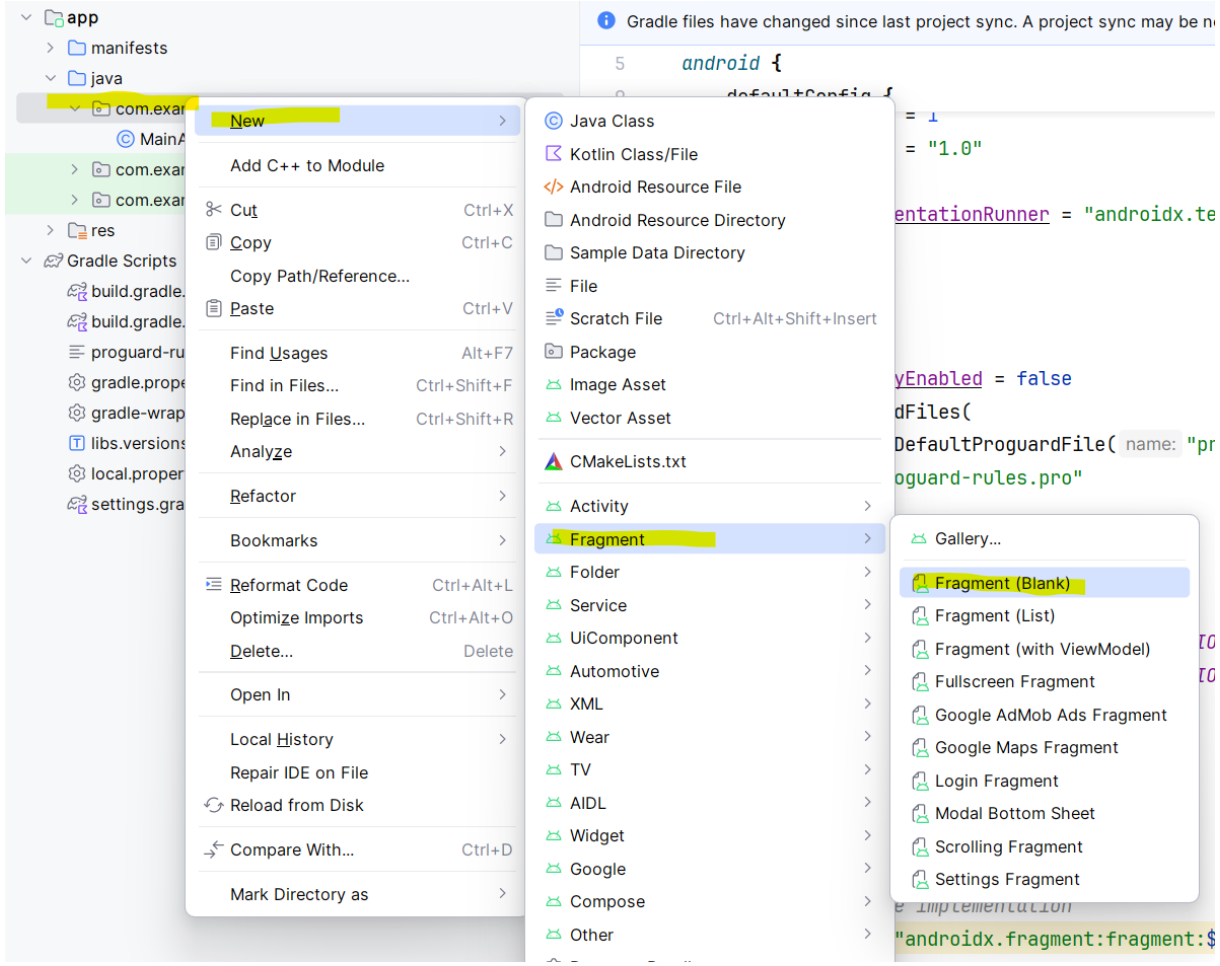

Pamiętaj o zsynchronizowaniu projektu.



# Tworzenie fragmentu

## Tworzenie fragmentu:

Fragment, podobnie jak aktywność, opisany jest w pliku xml oraz w pliku kodu java



# Tworzenie fragmentu

Przykładowe kod umieszczony w szablonie można usunąć. Najprostsza wersja kodu JAVA dla fragmentu wygląda tak:

```
package com.example.fragment03;

import android.os.Bundle;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class PierwszyFragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_pierwszy, container, false);
    }
}
```

Tworzymy własną klasę dziedziczącą są po klasie Fragment.

Zwróćmy uwagę na jej nazwę, gdyż za chwilę będziemy tworzyć obiekt tej klasy.

# Tworzenie fragmentu

Fragment podobnie jak ta aktywność posiada swój layout opisany w XML oraz kod. Uchwyty do kontrolek podobnie jak w zwykłej aktywności uzyskujemy za pomocą findViewById(). Zauważyć trzeba jednak pewną różnicę:

Metoda inflate() „nadmuchująca” fragment zwraca referencję do obiektu typu View.

Referencja ta będzie nam potrzebna aby wskazać, gdzie należy szukać poszczególnych widoków składających się na layout fragmentu

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_pierwszy, container, false);
    TextView textView = view.findViewById(R.id.textView02);
    textView.setText("Tekst wstawiony z poziomu kodu");
    return view;
}
```

view.findViewById() znajduje kontrolkę w widoku o referencji „view” czyli naszym fragmencie. (Inaczej szukała by jej w aktywności hostującej fragment)

# Dodawanie fragmentu

Pierwszym krokiem jest przygotowanie kontenera w którym fragment zostanie wyświetlony

```
<androidx.fragment.app.FragmentContainerView
   android:id="@+id/fragmentContainer"
   android:layout_width="0dp"
   android:layout_height="0dp"
   app:layout_constraintHeight_percent="0.33"

app:layout_constraintBottom_toBottomOf="parent"
   app:layout_constraintEnd_toEndOf="parent"
   app:layout_constraintStart_toStartOf="parent„
   tools:layout="@layout/fragment_pierwszy"
   />
```

FragmentConteiner pozycjonujemy jak każdy inny widok. W tym przykładzie zajmie on dolną 1/3 ekranu

To polecenie (tak jak wszystkie polecenia z przestrzeni nazw tools:) służy tylko do uzyskania podglądu w Android Studio. Po uruchomieniu aplikacji kontener będzie pusty.

Nie ma możliwości załadowania fragmentu do kontenera z poziomu XML.

# Dodawanie fragmentu

Dodawanie fragmentu rozpoczynamy od przygotowania referencji ta fragmentu oraz dla obiektu FragmentManager.

FragmentManager zarządza fragmentami umożliwia ich dodawanie oraz usuwanie.

Tworzymy referencję do naszego fragmentu. Nazwą typu jest nazwa klasy którą utworzyliśmy (patrz 2 slajdy wyżej)

```
PierwszyFragment myFragment;

FragmentManager fragmentManager;
```

Referencje do menadżera fragmentów

Obie referencje powinny być globalne (pola klasy MainActivity)

# Dodawanie fragmentu

```
fragmentManager = getSupportFragmentManager();

myFragment = new PierwszyFragment();

// Rozpoczynamy transakcję - proces dodawania fragmentu
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
// Dodajemy fragment
fragmentTransaction.add(R.id.fragmentContainer, myFragment);
// Dodajemy fragment do backStacku
fragmentManager.restoreBackStack("fragment01");
// Zatwierdzamy transakcję
fragmentTransaction.commit();
```

Pobieramy instancję obiektu FragmentManager.

Tworzymy fragment. W tym momencie jest on tworzony w pamięci i nie jest jeszcze wyświetlany.

Dodając fragment podajemy w jakim kontenerze ma być wyświetlony. Można dodać wiele fragmentów do różnych kontenerów.

Back Stack (stos wsteczny) to struktura danych, która przechowuje sekwencję aktywności lub fragmentów w miarę, jak użytkownik przechodzi z jednego ekranu na inny. Kiedy użytkownik naciska przycisk „Wstecz” na urządzeniu, system przemieszcza się wstecz przez ten stos.

Fragmentu nie musimy dodawać do stosu. Jeżeli tego nie zrobimy będzie on połączony z aktywnością którego hostuje.

# Usuwanie fragmentu

Istnieje kilka metod wyszukiwania i usuwania fragmentu. Najczęściej posiadamy jednak referencje do fragmentu który chcemy usunąć.

Rozpoczynamy nową transakcję

```
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

fragmentTransaction.remove(myFragment);

fragmentTransaction.commit();
```

Usuwamy wskazany fragment. Parametrem jest referencja do obiektu.

Zatwierdzamy transakcję.

Uwaga: należy upewnić się, że fragment do którego prowadzi referencja jest aktualnie wyświetlany inaczej spowodujemy błąd.

# Usuwanie fragmentu

Przykład aktywności która pokazuje i ukrywa fragment po naciśnięciu przycisków.

```
public class MainActivity extends AppCompatActivity {

    PierwszyFragment myFragment;
    FragmentManager fragmentManager;

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

        fragmentManager = getSupportFragmentManager();
        myFragment = null;

        Button pokaz = findViewById(R.id.button01);
        pokaz.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (myFragment == null){
                    myFragment = new PierwszyFragment();
                    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
                    fragmentTransaction.add(R.id.fragmentContainer, myFragment);
                    fragmentTransaction.commit();
                }
            }
        });

        Button schowaj = findViewById(R.id.button02);
        schowaj.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (myFragment != null) {
                    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
                    fragmentTransaction.remove(myFragment);
                    fragmentTransaction.commit();
                    myFragment = null;
                }
            }
        });
    }
}
```

# Przesyłanie danych do fragmentu

Otwierając fragment możemy przesłać do niego dane dołączone jako zmienne typuBundle (analogicznie jak było to w przypadku otwierania aktywności)

```
myFragment = new PierwszyFragment();

Bundle paczka = new Bundle();
paczka.putString("tekst", "Przesłany tekst do wyświetlenia");
myFragment.setArguments(paczka);

FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
fragmentTransaction.add(R.id.fragmentContainer, myFragment);
fragmentTransaction.addToBackStack("fragment01");
fragmentTransaction.commit();
```

Danych może być więcej aby je odebrać dane musimy znać ich typy oraz klucze.

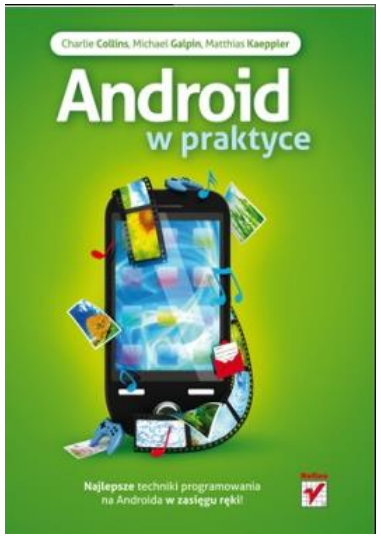
# Przesyłanie danych do fragmentu

Przesłane to fragmentu dane możemy odebrać w metodzie onCeeateView()

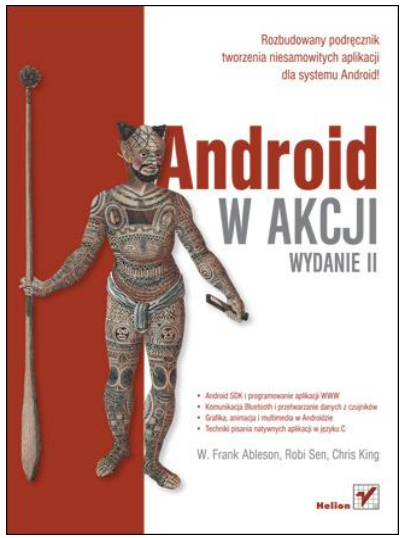
Funkcja getArguments() zwróci wartość nul i jeżeli do fragmentu nie zostały przesłane żadne dane

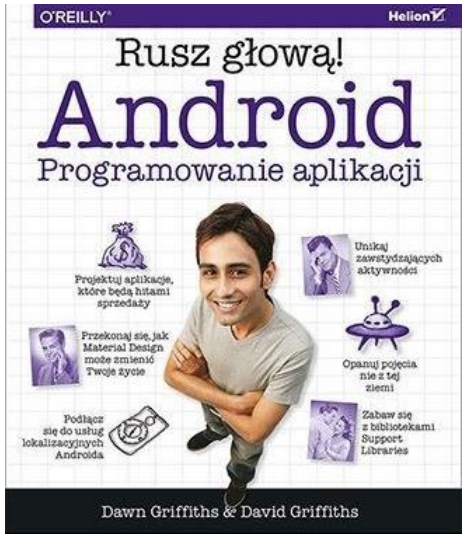
```
if (getArguments() != null) {
    String wartosc = getArguments().getString("tekst");
    // Użycie przesłanej wartości
   textView.setText(wartosc);
}
else
    textView.setText("Brak danych");
```

# Literatura




https://developer.android.com




https://javastart.pl/baza-wiedzy/android/

https://forum.android.com.pl