

# Instrukcja

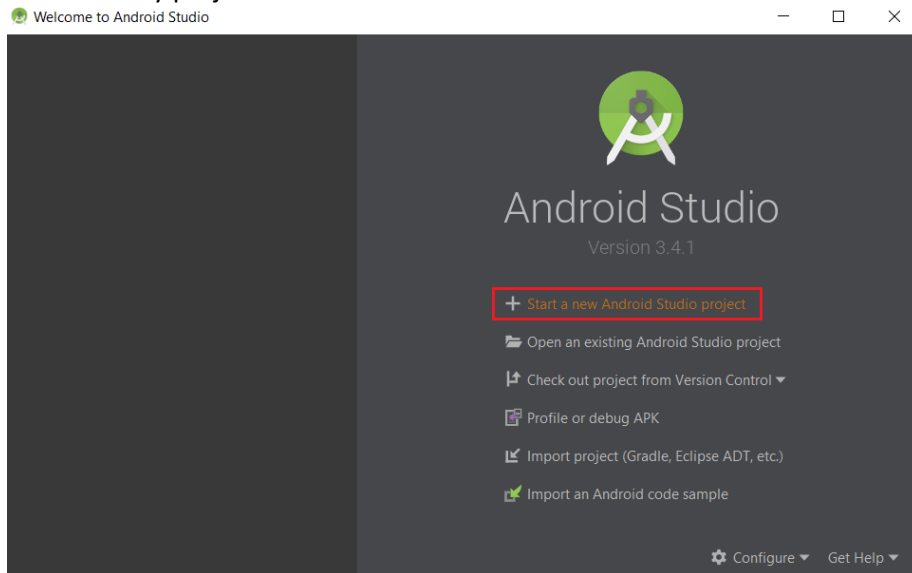
## Spis treści

1. Tworzenie projektu.....	2
2. Tworzenie klasy do przechowywania danych.....	4
3. Tworzenie danych.....	6
4. Tworzenie własnego adaptera listy .....	8
5. Tworzenie aktywności listy .....	11
6. Tworzenie własnego fragmentu tabbed.....	12
7. Tworzenie własnego adaptera tabbed .....	14
8. Tworzenie aktywności tabbed .....	15
9. Tworzenie intencji wyboru obrazu .....	16

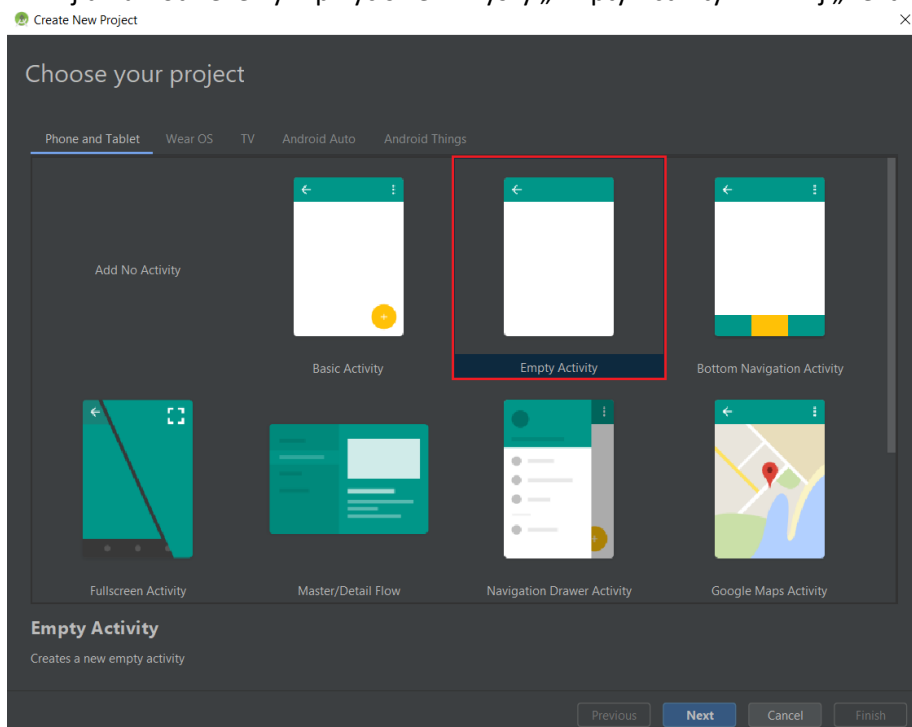
## 1. Tworzenie projektu

Jeżeli chcesz dodać galerię do już istniejącego projektu pomiń tworzenie nowego projektu i przejdź do tworzenia klasy do przechowywania danych!

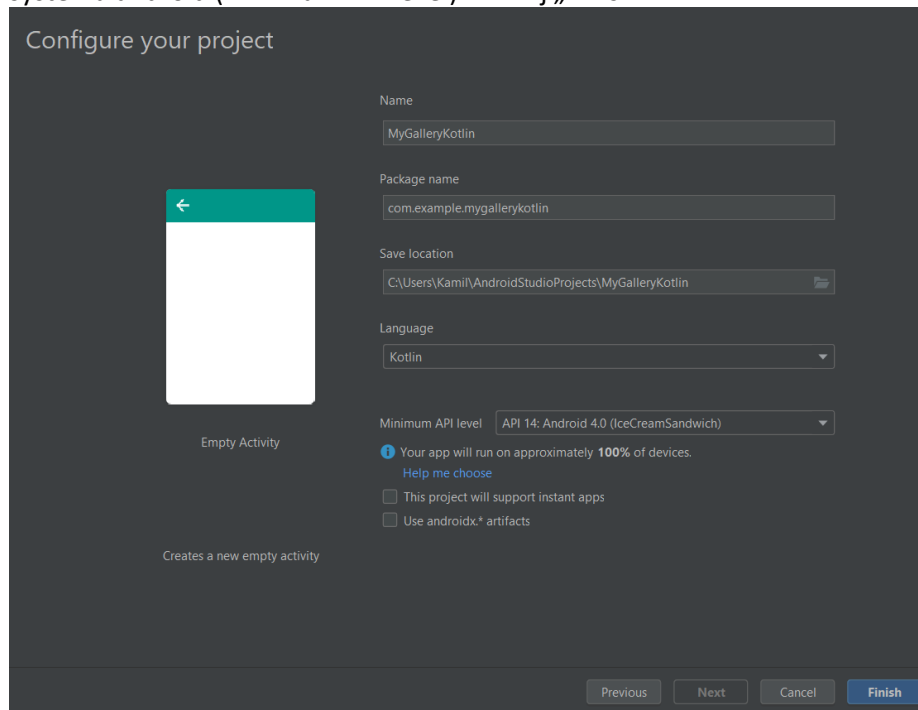
1. Uruchom android studio
2. Utwórz nowy projekt w Android Studio



3. Kliknij dwukrotnie lewym przyciskiem myszy „Empty Activity” i kliknij „next”



4. Wprowadź nazwę projektu (Name), wybierz język (Language) Kotlin i ustaw minimalną wersję systemu android (Minimum API level) i kliknij „Finish”



Configure your project

Name  
MyGalleryKotlin

Package name  
com.example.mygallerykotlin

Save location  
C:\Users\Kamil\AndroidStudioProjects\MyGalleryKotlin

Language  
Kotlin

Minimum API level  
API 14: Android 4.0 (IceCreamSandwich)

Empty Activity

Creates a new empty activity

Help me choose

☐ This project will support instant apps

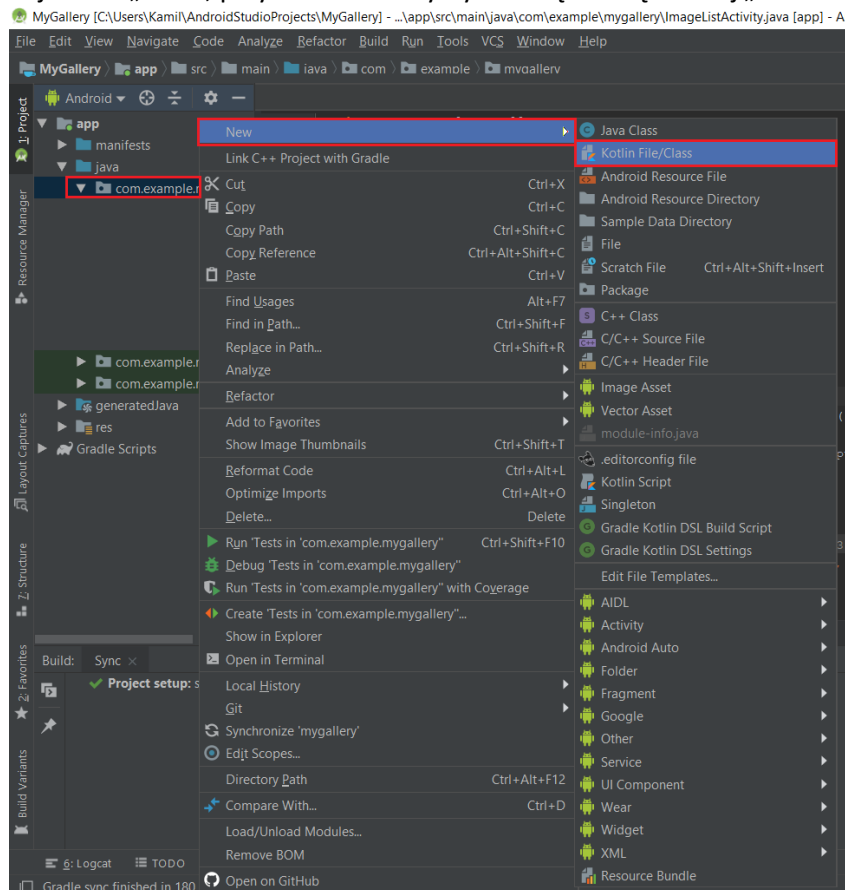
☐ Use androidx\* artifacts

Previous Next Cancel Finish

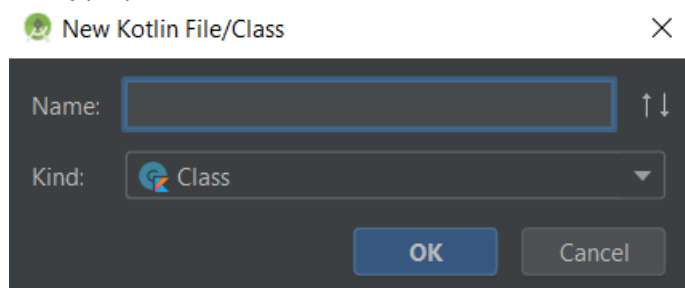
## 2. Tworzenie klasy do przechowywania danych

1. Stwórz nową klasę „ImageEntity” w pakiecie „com.example.[nazwa projektu]”. Zadaniem klasy będzie przechowywać informację o Obrazie.

Po lewej stronie kliknij prawym przyciskiem myszy „com.example.nazwaprojektu”, następnie najedź na „new”, przysuń kursor myszy w lewą stronę i kliknij „Kotlin Class”



W nowo otwartym oknie wpisz nazwę klasy „ImageEntity”, zmień wartość pola „Kind” na „Class” i kliknij przycisk ok



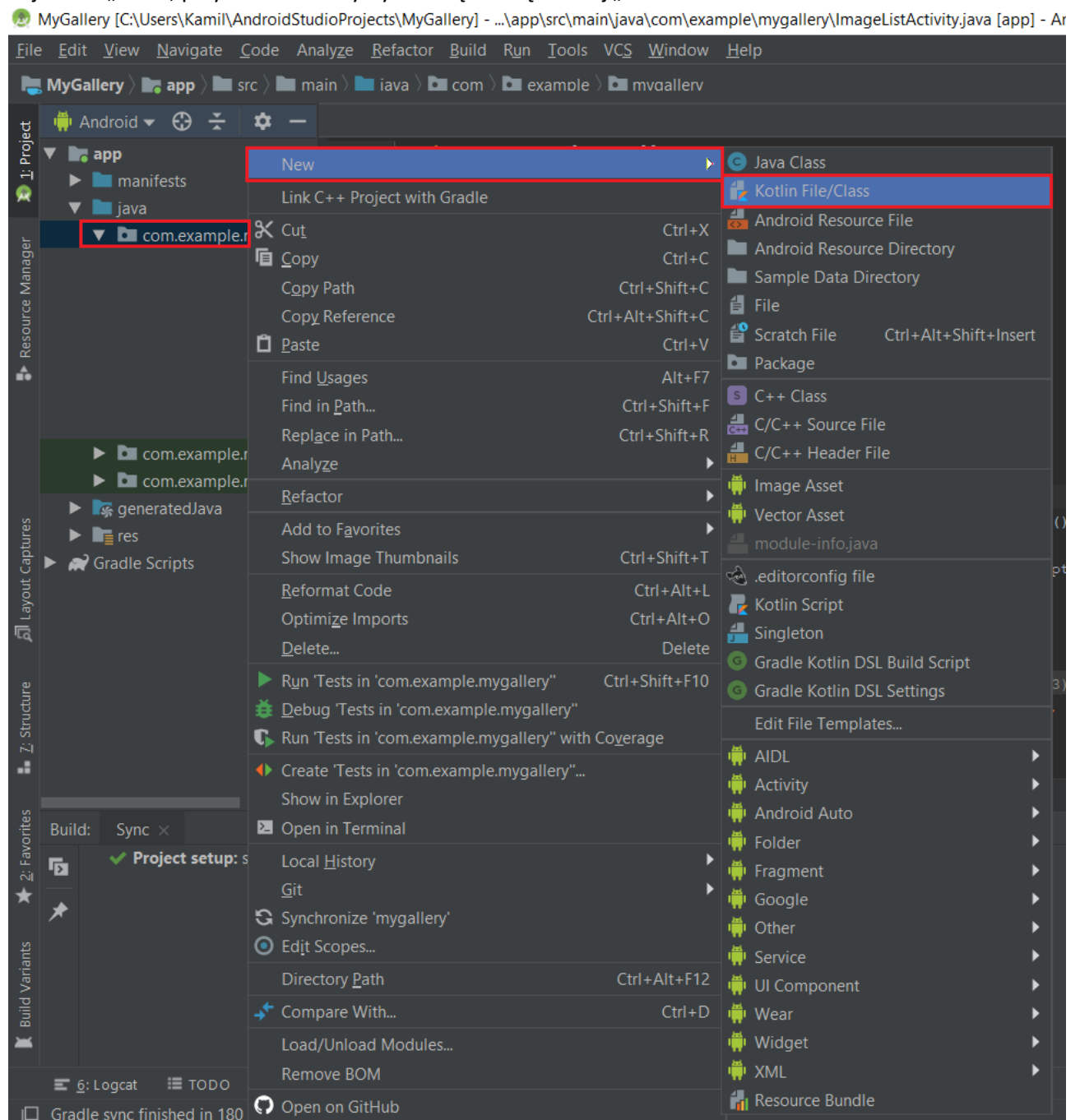
2. Utwórz wewnątrz konstruktora klasy pola typu value przechowujące wybrane informacje o obrazie. Możesz swobodnie dodać nowe informacje np. rok powstania, wymiary, itp. W języku kotlin w odróżnieniu od javy nie musisz deklarować oddzielnego konstruktora ani getterów. Sprawia to że kod jest czytelniejszy i łatwiejszy w edycji. Javowa klasa „ImageEntity” posiadała 37 linijek kodu, kotlinowa wersja posiada jedynie 9 linijek kodu.

```
class ImageEntity internal constructor(  
    val title: String,  
    val author: String,  
    val place: String,  
    val description: String,  
    val image: Int  
)
```

### 3. Tworzenie danych

1. Stwórz nową klasę „ImageGenerator” w pakiecie „com.example.[nazwa projektu]”. Zadaniem klasy będzie stworzyć dane o obrazach.

Po lewej stronie kliknij prawym przyciskiem myszy „com.example.nazwaprojektu”, następnie najedź na „new”, przesuń kursor myszy w lewą stronę i kliknij „Java Class”



2. Wrzuć do pakietu „res .drawable” kilka obrazków. W moim przypadku są to obrazy image1, image2, image3 i image4. Możesz umieścić dowolne obrazy i nazwać je inaczej.

**Pamiętaj aby nazwy nie zawierały spacji.**

3. Stwórz i zaimplementuj metodę „generateImages” która zwróci listę obrazów. Pamiętaj żeby w „R.drawable.” po kropce podać nazwę pliku z obrazem, który umieściliśmy w folderze „drawable”. Dla lepszego efektu możesz zamienić teks "Tutaj wstaw długi opis..." na kilkuset znakowy teks. Możesz skorzystać z generatora tekstu lub skopiować go z Wikipedii.

**W prawdziwej aplikacji dane pobiera się z bazy danych lub pamięci urządzenia.  
Nie należy ich hard kodować jak w poniższym przykładzie!**

```
fun generateImages(): Array<ImageEntity> {
    return arrayOf(

        ImageEntity(
            "Trzy słoneczniki w wazonie",
            "Vincent van Gogh",
            "Muzeum Vincenta van Gogha",
            "...",
            R.drawable.image1
        ),

        ImageEntity(
            "Wazon z piętnastoma słonecznikami",
            "Vincent van Gogh",
            "National Gallery w Londynie",
            "...",
            R.drawable.image2
        ),

        ImageEntity(
            "Wazon z dwunastoma słonecznikami",
            "Vincent van Gogh",
            "Nowa Pinakoteka",
            "...",
            R.drawable.image3
        ),

        ImageEntity(
            "Czaszka z palącym się papierosem",
            "Vincent van Gogh",
            "Muzeum Vincenta van Gogha",
            "...",
            R.drawable.image4
        )
    )
}
```

## 4. Tworzenie własnego adaptera listy

1. Stwórz nową klasę „MyImageListAdapter” w pakiecie „com.example.[nazwa projektu]”. Zadaniem klasy jest tworzenie pojedynczego elementu listy obrazów.
2. Utwórz wewnątrz konstruktora klasy prywatne pola. Typ „Array<ImageEntity>” określa kolekcję obiektów które chcemy wyświetlić. W przypadku tworzenia własnej listy możesz stworzyć własną kolekcję obiektów np. „Array<Animal>”, „List<Car> cars”, „Buildings buildings”

```
class MyImageListAdapter constructor(  
    private val appContext: Context,  
    private val layoutResourceId: Int,  
    private val images: Array<ImageEntity>  
)
```

3. Jeżeli Android studio zapyta się o import klas to zawsze wyrażaj zgodę. Jeżeli jakaś klasa będzie zaznaczona na czerwono, kliknij na nią lewym przyciskiem myszy. Android Studio zasugeruje import klasy. Kliknij ALT + ENTER, w razie konieczności wybierz z listy „Import class”. W razie potrzeby importuj brakujące klasy.
4. Rozszerz klasę o „ArrayAdapter<ImageEntity>”, w razie potrzeby zaimportuj brakującą klasę. W przypadku tworzenia własnego adaptera należy zamienić „<ImageEntity>” na klasę której obiekty chcemy wyświetlić.

```
class MyImageListAdapter constructor(  
    private val appContext: Context,  
    private val layoutResourceId: Int,  
    private val images: Array<ImageEntity>  
) : ArrayAdapter<ImageEntity>(appContext, layoutResourceId, images) {  
    ...  
}
```

5. Stwórz nowy layout „fragment\_image\_list” w pakiecie „res.layout”, określający wygląd pojedynczego elementu listy.
4. Uzupełnij layout poniższym kodem lub stwórz własny layout. Spróbuj poeksperymentować z różnymi układami aż osiągniesz satysfakcjonujący efekt.

**Pamiętaj aby pola które chcesz uzupełnić wartościami z obiektu posiadały unikatowy ID. Pozostałe elementy layoutu nie muszą ich posiadać.**

```
<?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"  
    android:id="@+id/row"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
  
    <android.support.constraint.Guideline  
        android:id="@+id/guideline2"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:orientation="vertical"  
        app:layout_constraintGuide_begin="156dp" />
```



```

<android.support.constraint.Guideline
    android:id="@+id/guideline1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_begin="226dp" />

<ImageView
    android:id="@+id/imageView"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginBottom="8dp"
    android:contentDescription="@string/app_name"
    app:layout_constraintBottom_toTopOf="@+id/guideline1"
    app:layout_constraintEnd_toStartOf="@+id/guideline2"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:srcCompat="@android:drawable/ic_menu_crop" />

<LinearLayout
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginBottom="8dp"
    android:orientation="vertical"
    app:layout_constraintBottom_toTopOf="@+id/guideline1"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="@+id/guideline2"
    app:layout_constraintTop_toTopOf="parent">

    <TextView
        android:id="@+id/textViewTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@android:string/unknownName"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textViewAuthor"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@android:string/unknownName" />

    <TextView
        android:id="@+id/textViewPlace"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@android:string/unknownName" />

    <TextView
        android:id="@+id/textViewDescription"
        android:layout_width="match_parent"

```

```

        android:layout_height="match_parent"
        android:text="@android:string/unknownName" />
    </LinearLayout>

</android.support.constraint.ConstraintLayout>

```

6. Teraz przypisz wartości pól obrazów do pól w layoucie. Wróć do klasy „MyImageListAdapter” i nadpisz metodę „getView”. Zwróć uwagę na:  
 „(currentView.findViewById<View>( R.id.idPolaTekstowego) as TextView).text = „Mój tekst””  
 Pierwszy nawias ma za zadanie znaleźć pole tekstowe o id „idPolaTekstowego” znajdującego się w layoucie. Następnie odwołujesz się do jego wartości „text”, jak nazwa wskazuje służy ona przechowywaniu tekstu w polu. Jako parametr wprowadzamy „image.pole” jest to odwołanie się do getera zwracającego wartość wskazanego pola.  
 W razie potrzeby zaimportuj brakujące klasy.

```

@SuppressLint("ViewHolder")
override fun getView(position: Int, convertView: View?, parent: ViewGroup):
View {
    val inflater = (appContext as Activity).layoutInflater
    val currentView = inflater.inflate(layoutResourceId, parent, false)
    val image = images[position]

    (currentView.findViewById<View>(R.id.textViewTitle) as TextView).text =
image.title
    (currentView.findViewById<View>(R.id.textViewAuthor) as TextView).text =
image.author
    (currentView.findViewById<View>(R.id.textViewPlace) as TextView).text =
image.place
    (currentView.findViewById<View>(R.id.textViewDescription) as
TextView).text = image.description
    (currentView.findViewById<View>(R.id.imageView) as
ImageView).setImageResource(image.image)

    return currentView
}

```

## 5. Tworzenie aktywności listy

1. Jeżeli rozszerzasz istniejącą aplikację stwórz nową aktywność „ImageListActivity”  
**Pamiętaj aby ustawić intencję umożliwiającą przejście z twojej aplikacji do nowej aktywności!**  
Jeżeli tworzysz aplikację od początku wykorzystaj już istniejącą aktywność „MainActivity”. W pakiecie „com.example.[nazwa projektu]” kliknij prawym przyciskiem myszy na „MainActivity”, wybierz „Refactor” i „Rename”. Zmień nazwę aktywności na „ImageListActivity”. Analogicznie zmień nazwę layoutu „activity\_main” w pakiecie „res.layout” na „activity\_image\_list”
2. Przejdź do edycji layoutu „activity\_image\_list”. Usuń pole tekstowe „Hello World!”, zamień „ConstraintLayout” na „LinearLayout”, zmieść w środku „ListView”. Rozciągnij go na cały ekran i ustaw id na „imagesList”. **Możesz też zastąpić kod poniższym kodem**

```
<?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"
    tools:context=".ImageListActivity">

    <ListView
        android:id="@+id/imagesList"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
</LinearLayout>
```

3. Przejdź do edycji klasy „ImageListActivity”. Do metody „onCreate()” dopisz poniższy kod. W razie potrzeby zaimportuj brakujące klasy.  
Pierwsza linia tworzy kolekcję obiektów „ImagesEntity” (obrazy)  
Druga linia tworzy adapter który deklarowaliśmy w poprzednim punkcie.  
Trzecia linia pobiera „ListView” znajdujący się w layoutcie „activity\_image\_list”.  
Czwarta linia dodaje do „ListView” adapter, co powoduje zapełnienie „ListView” elementami według reguł określonych w adapterze.

```
val images = ImageGenerator().generateImages()

val myImageListAdapter = MyImageListAdapter(this,
    R.layout.fragment_image_list, images)
val listView = findViewById<ListView>(R.id.imagesList)

listView.adapter = myImageListAdapter
```

4. Uruchom aplikację i przetestuj działanie listy. Możesz spróbować edytować layout „fragment\_image\_list”, żeby uzyskać inny wygląd.

## 6. Tworzenie własnego fragmentu tabbed

1. Stwórz nowy layout „fragment\_image\_tabbed” określający wygląd pojedynczego taba
2. Uzupełnij layout poniższym kodem lub stwórz własny layout. Spróbuj poeksperymentować z różnymi układami aż osiągniesz satysfakcjonujący efekt.

Pamiętaj aby pola które chcesz uzupełnić wartościami z obiektu posiadały unikatowy ID. Z tego powodu każde id w poniższym kodzie ma na końcu „2”. Pozostałe elementy layoutu nie muszą ich posiadać.

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

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <ImageView
                android:id="@+id/imageView2"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:contentDescription="@string/app_name"
                android:adjustViewBounds="true"
                app:srcCompat="@android:drawable/ic_menu_crop" />

            <TextView
                android:id="@+id/textViewTitle2"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@android:string/unknownName"
                android:layout_marginTop="8dp"
                android:textSize="24sp" />

            <TextView
                android:id="@+id/textViewAuthor2"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="8dp"
                android:text="@android:string/unknownName" />

            <TextView
                android:id="@+id/textViewPlace2"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="8dp"
                android:text="@android:string/unknownName" />

            <TextView
                android:id="@+id/textViewDescription2"
                android:layout_width="match_parent"
```

```

        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="@android:string/unknownName" />
    </LinearLayout>
</ScrollView>

</RelativeLayout>

```

3. Stwórz nową klasę „MyImageListTabbedFragment” w pakiecie „com.example.[nazwa projektu]”. Zadaniem klasy jest tworzenie widoku poszczególnych tabów.
4. Stwórz wewnątrz klasy leniwe pole przechowujące obiekt na podstawie którego ma powstać fragment. W przypadku tworzenia własnego fragmentu należy użyć własnego obiektu który będzie reprezentacją danych.

```
internal lateinit var image: ImageEntity
```

5. Rozszerz klasę „MyImageTabbedFragment” o klasę „Fragment”

```
class MyImageTabbedFragment : Fragment() {
    ...
}
```

6. Nadpisz metodę „onCreate”, zasada taka sama jak w klasie „MyImageListAdapter”.

```

override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?): View? {
    val currentView = inflater.inflate(R.layout.fragment_image_tabbed,
        container, false)

    (currentView.findViewById<View>(R.id.textViewTitle2) as TextView).text =
        image.title
    (currentView.findViewById<View>(R.id.textViewAuthor2) as TextView).text =
        image.author
    (currentView.findViewById<View>(R.id.textViewPlace2) as TextView).text =
        image.place
    (currentView.findViewById<View>(R.id.textViewDescription2) as
        TextView).text = image.description
    (currentView.findViewById<View>(R.id.imageView2) as
        ImageView).setImageResource(image.image)

    return currentView
}

```

## 7. Tworzenie własnego adaptera tabbed

1. Stwórz nową klasę „MyImageTabbedAdapter” w pakiecie „com.example.[nazwa projektu]”.
2. Utwórz wewnątrz klasy prywatne pole przechowujące listę fragmentów.

```
private val mFragmentManager = ArrayList<Fragment>()
```

3. Rozszerz klasę o „FragmentManagerAdapter” w razie potrzeby zaimportuj brakującą klasę oraz zadeklaruj konstruktor. Deklaracja klasy zos

```
class MyImageTabbedAdapter constructor (
    fragmentManager: FragmentManager,
    images: Array<ImageEntity>
) : FragmentStatePagerAdapter(fragmentManager) {
    ...
}
```

4. Deklaracje klasy zostanie podkreślona na czerwono, ponieważ brakuje metod implementowanej klasy. Nadpisz metodę „getItem” zwracającą fragment dla danej pozycji i „getCount” zwracającą ilość tabów to stworzenia.

```
override fun getItem(position: Int): Fragment {
    return mFragmentManager[position]
}

override fun getCount(): Int {
    return mFragmentManager.size
}
```

5. Stwórz inicjalizator oraz wydziel metodę tworzącą fragment do prywatnej metody „createFragment”

```
init {
    for (image in images) {
        mFragmentManager.add(createFragment(image))
    }
}

private fun createFragment(image: ImageEntity): MyImageTabbedFragment {
    val fragment = MyImageTabbedFragment()
    fragment.image = image
    return fragment
}
```

## 8. Tworzenie aktywności tabbed

1. Utwórz nową aktywność „ImageTabbedActivity”
2. Przejdź do edycji layoutu „activity\_image\_tabbed”. Usuń pole tekstowe „Hello World!” (jeżeli zostało wygenerowane), zamień „ConstraintLayout” na „LinearLayout”, umieść w środku „TabLayout”, ustaw id na „TabLayout”, dodaj „ViewPager” rozciągnij go na cały ekran i ustaw id na „viewPager”. **Możesz też zastąpić kod poniższym kodem**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.design.widget.TabLayout
        android:id="@+id/tabLayout"
        android:layout_width="0dp"
        android:layout_height="0dp"/>

    <android.support.v4.view.ViewPager
        android:id="@+id/viewPager"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

    </android.support.v4.view.ViewPager>
</LinearLayout>
```

3. Przejdź do edycji klasy „ImageTabbedActivity”. Do metody „onCreate()” dopisz poniższy kod. W razie potrzeby zaimportuj brakujące klasy.

Pierwsza linia tworzy kolekcję obiektów „ImagesEntity” (obrazy)

Druga linia tworzy adapter który deklarowaliśmy w poprzednim punkcie.

Trzecia i czwarta linia pobiera „ViewPager” i „TabLayout” z layoutu „activity\_image\_tabbed”.

Piąta linia pobiera z intencji numer wybranego obrazu.

Szusta linia ustawienie adaptera

Siódma linia ustawianie wybranego obrazu jako aktywny

Ósma linia dodanie do layoutu viewPager

```
val images = ImageGenerator().generateImages()

val tabAdapter = MyImageTabbedAdapter(supportFragmentManager, images)
val viewPager = findViewById<ViewPager>(R.id.viewPager)
val tabLayout = findViewById<TabLayout>(R.id.tabLayout)
val selectedImage = intent.getIntExtra("imageSelected", 0)

viewPager.adapter = tabAdapter
viewPager.currentItem = selectedImage
tabLayout.setupWithViewPager(viewPager)
```

## 9. Tworzenie intencji wyboru obrazu

1. Przejdź do klasy ImageListActivity i w metodzie onCreate dopisz deklarację intencji przejścia do nowej aktywności z przekazaniem parametru określającego numer wybranego obrazu

```
listView.setOnItemClickListener = AdapterView.OnItemClickListener { _, _,  
position, _ ->  
    val intent = Intent(applicationContext, ImageTabbedActivity::class.java)  
    intent.putExtra("imageSelected", position)  
    startActivity(intent)  
}
```

2. Finalnie klasa ImageListActivity powinna wyglądać tak

```
class ImageListActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_image_list)  
  
        val images = ImageGenerator().generateImages()  
  
        val myImageListAdapter = MyImageListAdapter(this,  
R.layout.fragment_image_list, images)  
        val listView = findViewById<ListView>(R.id.imagesList)  
  
        listView.adapter = myImageListAdapter  
  
        listView.setOnItemClickListener = AdapterView.OnItemClickListener { _,  
_, position, _ ->  
            val intent = Intent(applicationContext,  
ImageTabbedActivity::class.java)  
            intent.putExtra("imageSelected", position)  
            startActivity(intent)  
        }  
    }  
}
```

3. Uruchom aplikację i przetestuj działanie. Możesz edytować layouty fragment\_image\_list i fragment\_image\_tabbed aby uzyskać inny efekt wizualny.