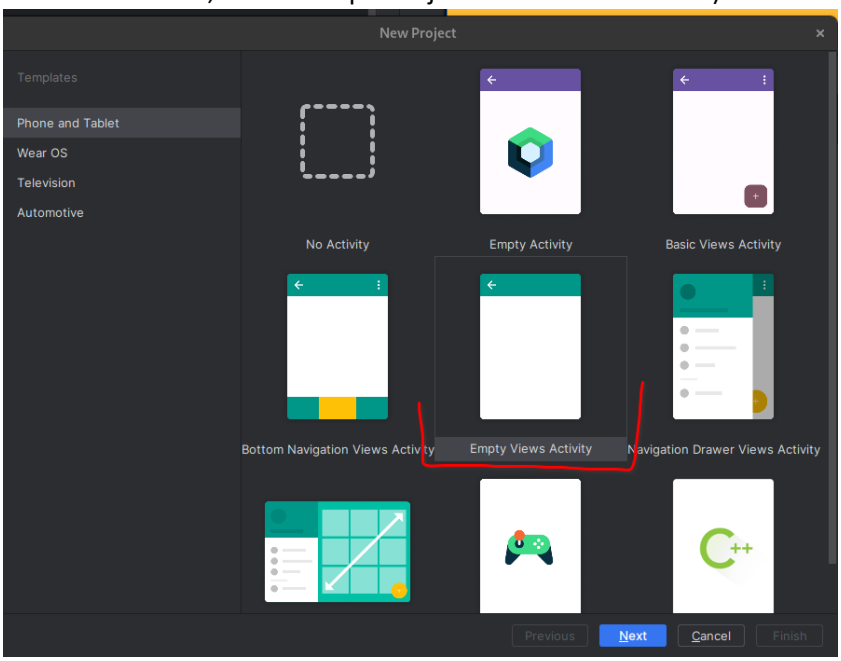
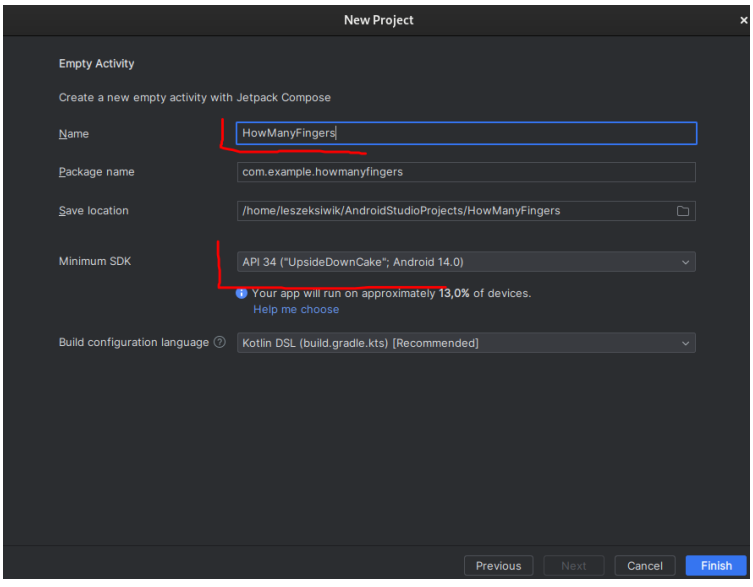
Bezpieczeństwo urządzeń mobilnych – Lab 2

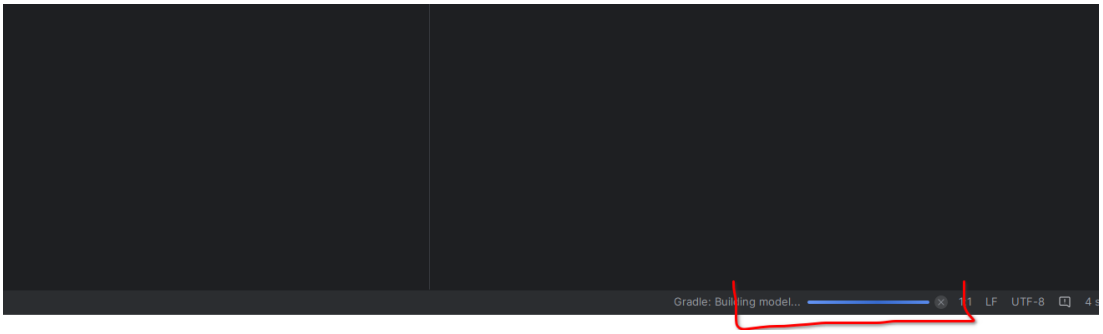
- Przyjrzymy się dziś w jaki sposób zmodyfikować istniejącą aplikację androidową
- Najpierw ją sobie zaimplementujemy i przy tej okazji przyjrzymy się jak budować aplikację w nieco odmienny sposób niż robiliśmy to ostatnio
- Na ostatnich zajęciach korzystaliśmy z pakietu Compose który pozwalał na budowanie interfejsu użytkownika bezpośrednio w kodzie.
- Dzisiaj, przyjrzymy się jak można to robić z wykorzystaniem tzw. Layout'ów.
- Noto do dzieła :)
- Uruchamiamy Android Studio
- Tworzymy nowy projekt File - > New Project -> Phone and Tablet
- Ale uwaga, tym razem jak template projektu wybieramy Empty **Views** activity (w odróżnieniu od Empty Activity którego używaliśmy ostatnio - trzeba na to zwrócić uwagę, bo w nazwie róznica niewielka, natomiast później w kodzie dość istotna :)



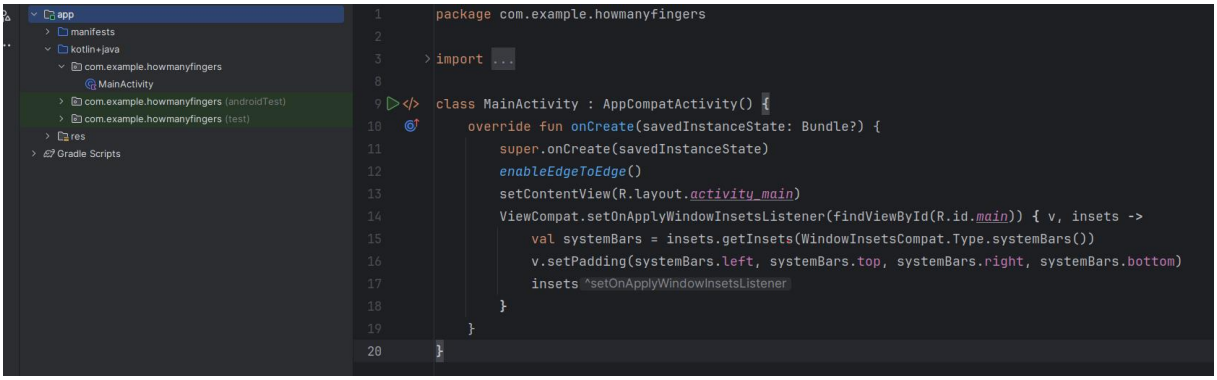
- Nazwijmy projekt HowManyFingers a jako minimum API Level wybierzmy API 34



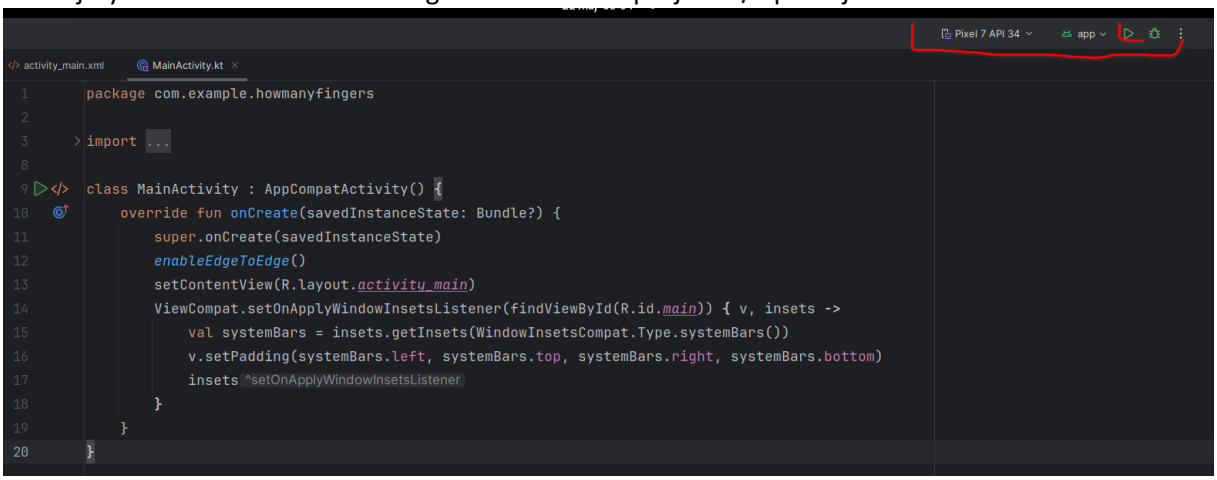
- Tradycyjnie, musimy poczekać (co może chwilkę zająć 🙂) az Gradle wykona wszystkie czynności związane ze stworzeniem projektu



- Po czym powinnismy uzyskac środowisko gotowe do pracy

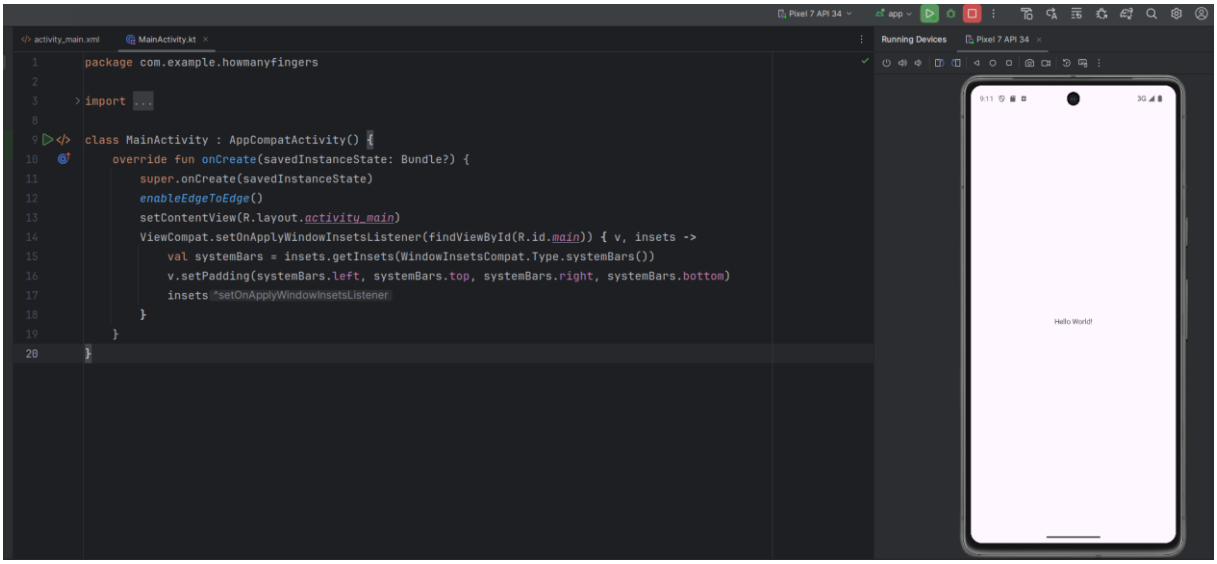


- Zacznijmy standardowo od testowego uruchomienia projektu / aplikacji na emulatorze.



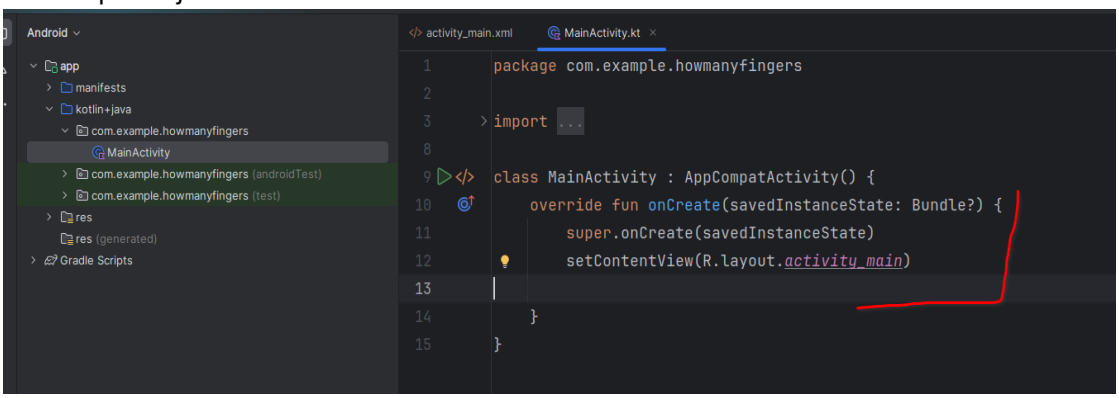
- Przy czym pamiętamy o wskazaniu emulatora (w moim przypadku Pixel 7 API 34) na którym chcemy uruchomić naszą aplikację.
- Jeśli nie mamy zdefiniowanego żadnego emulatora (tzw. AVD – Android Virtual Device), podobnie jak poprzednio musimy go utworzyć.
- Żeby nie zaciemniać bieżącej instrukcji, nie wstawiałem tutaj potrzebnych kroków, prosze w takim przypadku zerknąć do instrukcji z poprzednich zajęć

- Finalnie, powinniśmy zobaczyć naszą aplikację uruchomioną na emulatorze, jak poniżej:

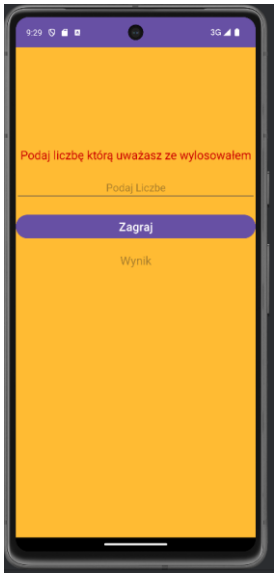
- 

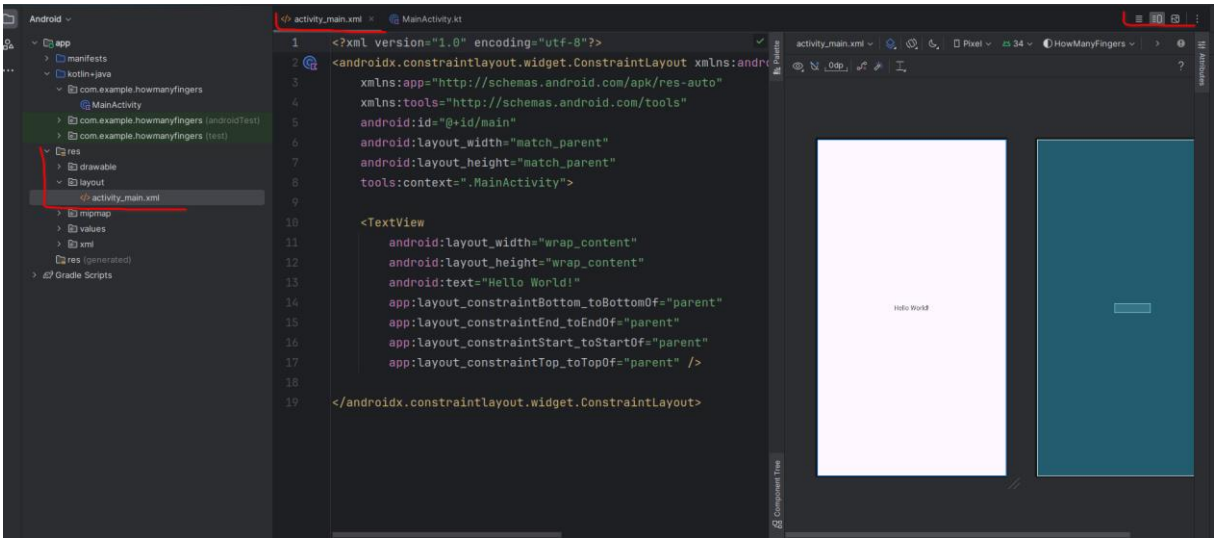
- Jeśli na tym etapie pojawiły się jakieś problemy / błędy – proszę o sygnał, postaram się pomóc je rozwiązać bo bez tego nie będziemy mogli iść dalej :)

Implementacja “Gry w Marynarza”

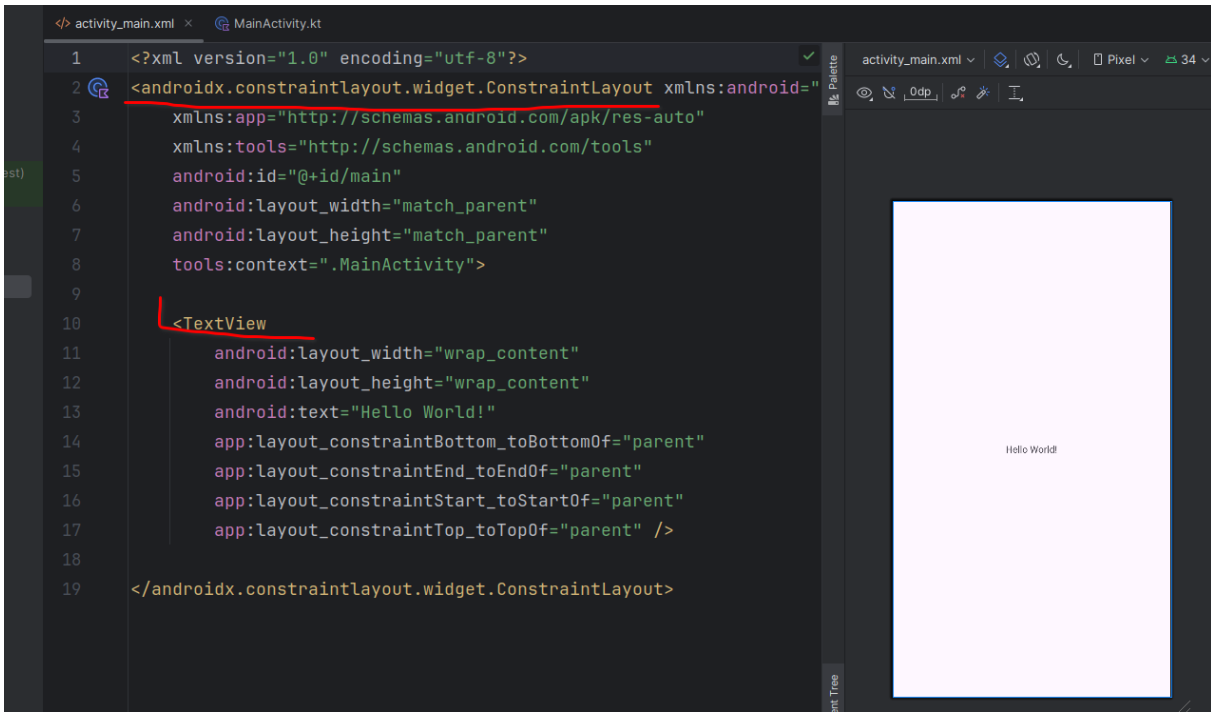
- Na potrzeby dalszych ćwiczeń, zaimplemenujemy sobie prostą “Gry w Marynarza”, w której aplikacja wylosuje liczbę z jakiegoś zakresu (powiedzmy od 1 do 10),i poprosi użytkownika o odgadnięcie jaka liczba została wylosowana.
- Jeśli użytkownik zgadnie - zdobywa punkt, jeśli nie -gramy dalej.
- Przejdźmy zatem do funkcji onCreate klasy MainActivity. Trzeba tam ewentualnie usunąć rzeczy które dodane zostały przez kreatora projektu, tak żeby zostało tam tylko to co na ekranie poniżej:
- 

- Załóżmy, że nasza aplikacja powinna wyglądać jak poniżej.

- 

- Wędrujemy zatem do pliku activity_main.xml zlokalizowanego w podkatalogu res/layout/ naszego projektu
- 

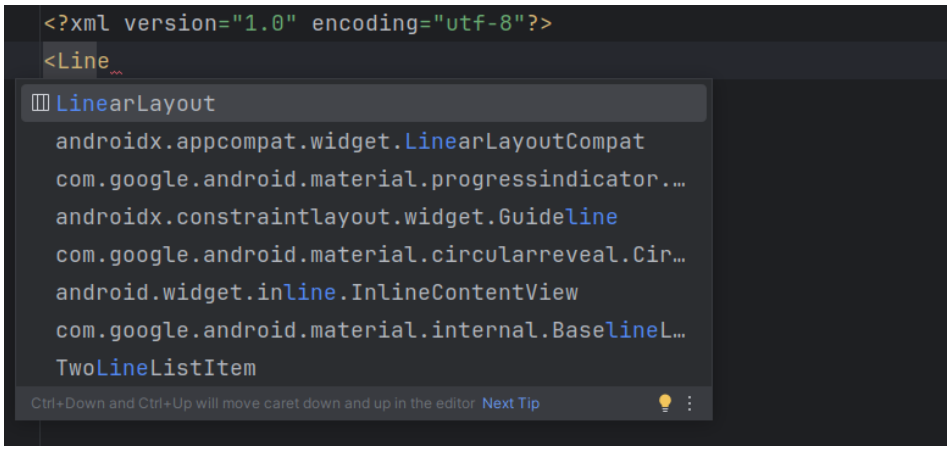
- I to tutaj znajduje się definicja wyglądu naszej aktywności
- Aktualnie mamy tam zdefiniowane dwa elementy
    - Layout (konkretnie ConstraintLayout) - rodzaj “kontenera” który we właściwy sobie sposób rozmieszcza umieszczone w nim elementy na ekranie
    - Oraz umieszczony wewnątrz layoutu element TextView który odpowiada za wyświetlenie napisu Hello World



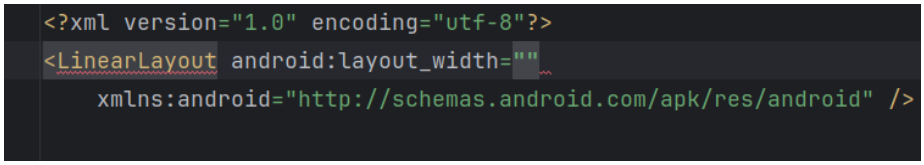
- Prosze zauważyć, że klasa naszej aktywności (plik MainActivity.kt), ma w tej chwili jedną metodę (onCreate - która wywoływana jest w momencie instancjonowania / uruchamiania naszej aktywności i - poza wywołaniem onCreate z klasy po której dziedziczy: super.onCreate(savedInstanceState) - jedyne co na razie robi to wywołuje metodę setContentView - odpowiadającą za ustawienie “zawartości” ekranu, używając w parametrze R.layout.activity_main.
- R - to specjalna klasa w AndroidSDK reprezentująca zdefiniowane/dostepne w projekcie zasoby (resources) takie jak ikony, napisy, grafiki, ale także layout’y.
- Innymi słowy metoda `setContentView(R.layout.activity_main)` wczytuje activity_main.xml w /res/layout i na jego podstawie ustawia zawartość ekranu aktywności
- Dobrze, wróćmy do naszego pliku xml’owego
- Generalnie mamy w AndroidSDK pewien zestaw predefiniowanych layout’ów. m.in.:
    - LinearLayout – w którym wszystkie elementy są umieszczane liniowo w pionie lub poziomie obok siebie lub jeden pod drugim (więcej informacji)
    - RelativeLayout – w którym elementy lokowane są “względem siebie – tzn. coś lokujemy na prawo od czegoś, coś innego pod czymś etc (Więcej informacji)
    - TableLayout – w kórym elementy są lokowane w wierszach i kolumnach (Więcej informacji)
    - ConstraintLayout – gdzie elementy są wzajemnie „pospinane” constraitami
- Ponieważ, w naszym przypadku, będziemy układać poszczególne elementy UI po prostu kolejno jeden pod drugim, skorzystamy z LinearLayout’u
- Usuwamy zatem w pliku xml’owym to co zaznaczyłem poniżej (czyli całość wygenerowaną przez wizard na etapie tworzenia projektu definicję ConstraintLayout’u):

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/re
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

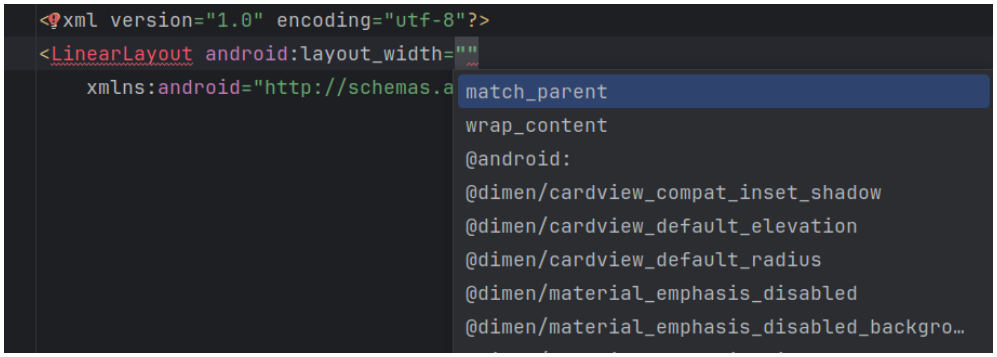
- Teraz, otwieramy tag xml'owy (<), zaczynamy pisać Line Studio powinno nam podpowiedzieć jak poniżej:



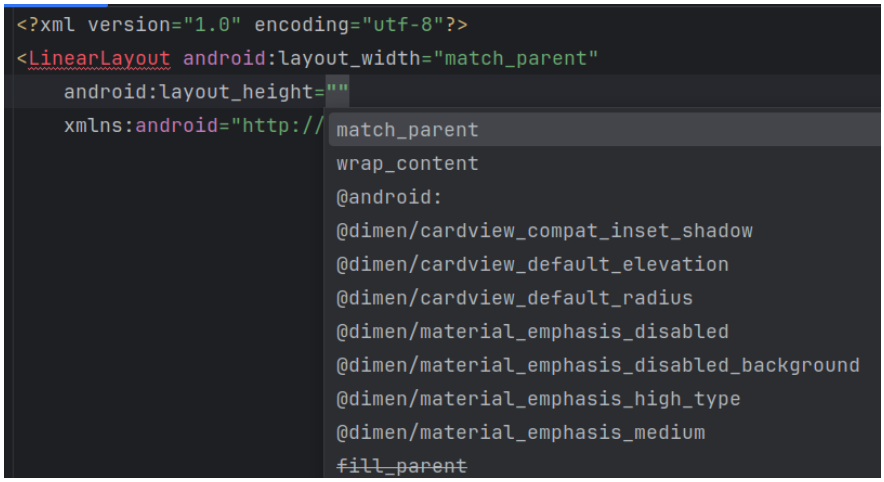
- Packamy enter, i dalej, zaczynamy pisać width… studio nam uzupełni:



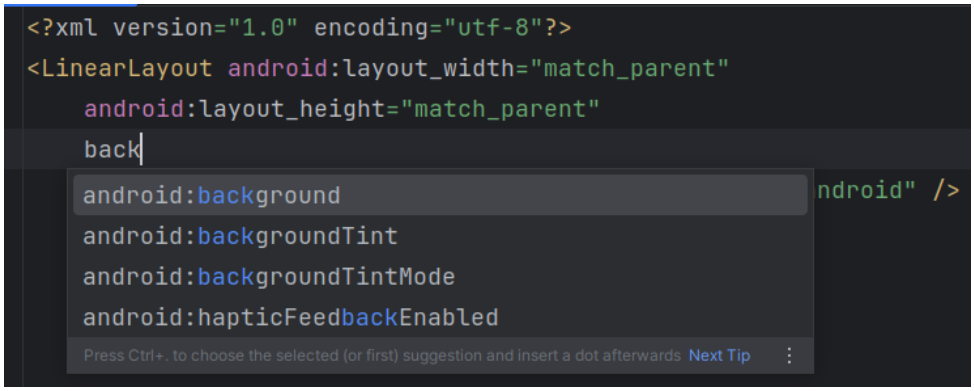
- Wchodzimy w cudzysłowy, dajemy ctrl + spacja, intelisense powinien pokazać nam dostępne opcje

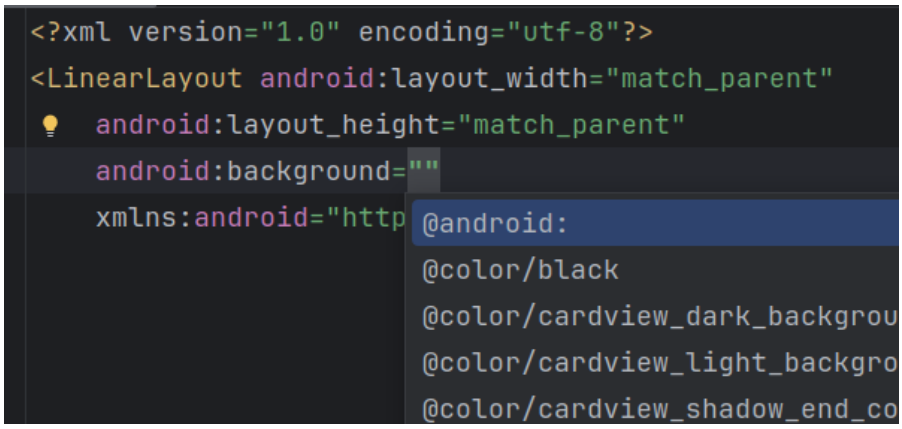


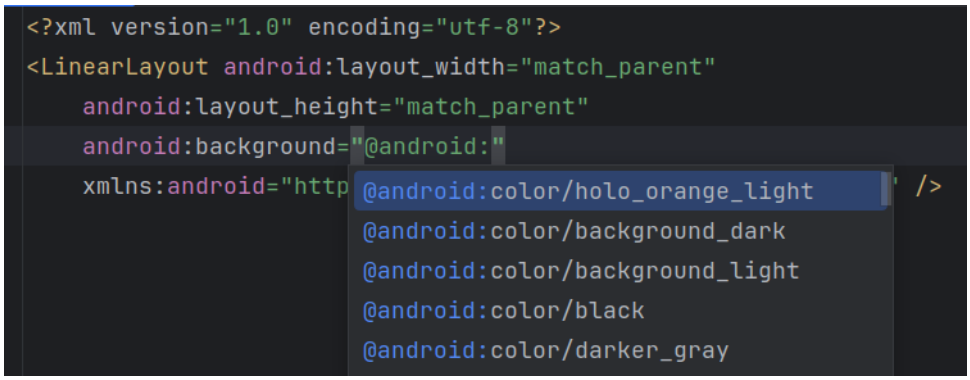
- i wybieramy match_parent, definiując w ten sposób szerokość naszego layoutu na taką samą jak szerokość elementu nadrzędnego. A że nasz layout nie jest zagnieżdżony w żadnym innym elemencie (jest layout'em pierwszego poziomu), to de facto ustawiamy jego szerokość na taką jaką będzie miało ekran urządzenia na którym będzie on rozmieszczany / na którym uruchamiana będzie nasza aplikacja
- Analogicznie, zaczynamy pisać height, Intelisense powinien podpowiedzieć:
- 

- packamy enter, przechodzimy do środka cudzysłowu, packamy ctrl+spacja, i ponownie wybieramy match_parent
- 

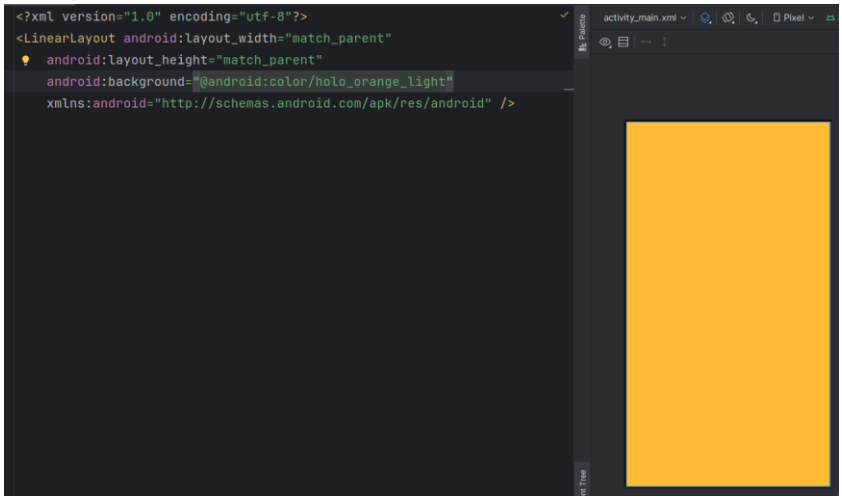
- W ten sposób, nasz layout jest tak szeroki i tak wysoki jaka będzie szerokość i wysokość ekranu na jakim będzie wyświetlany.
- Żeby ewentualnie łatwiej było śledzić gdzie/jak pozycjonowany jest nasz layout, ustawmy mu jakiś kolor tła. Czyli zaczynamy pisać back… Studio powinno pokazać nam dostępne opcje

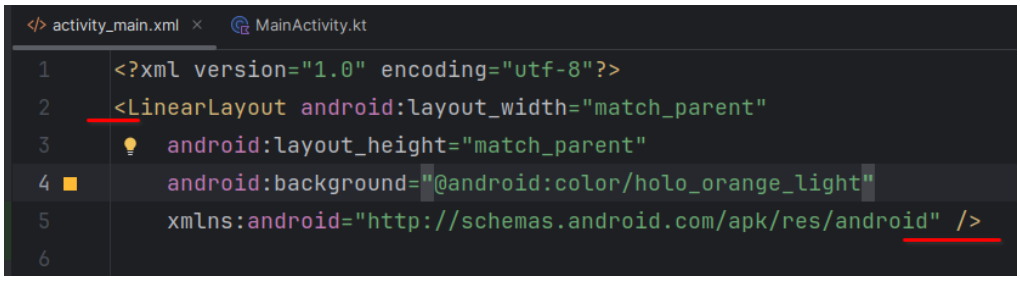
- 

- Wybieramy pierwszą, i dalej jako wartość wybieramy android
- 

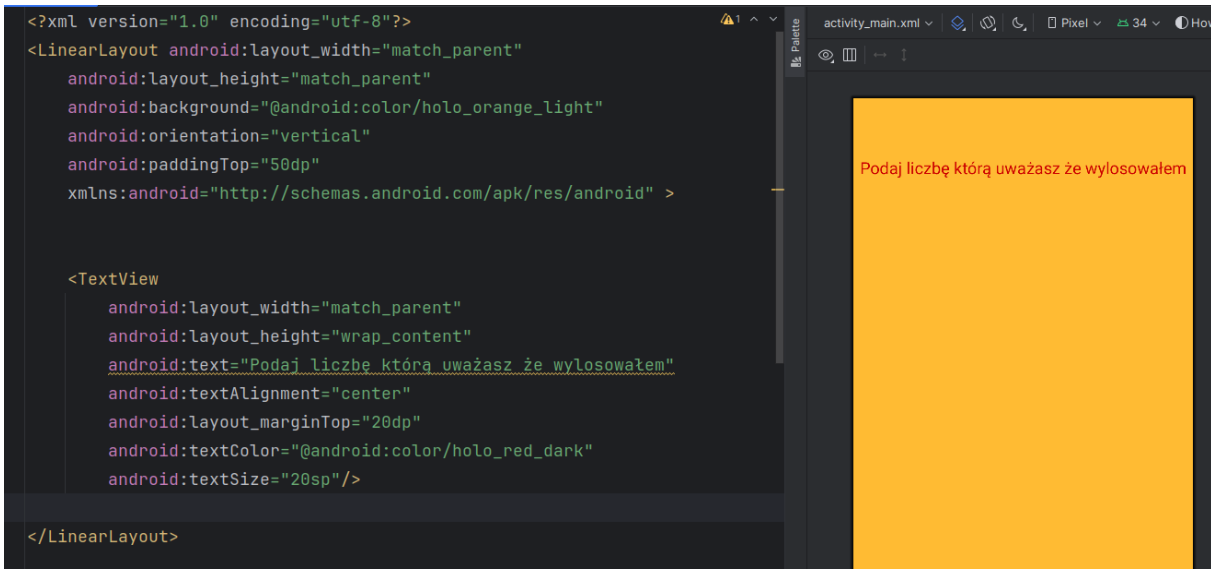
- i następnie, np. color/holo_orange_light (lub jakikolwiek inny, wg Państwa uznania :)
- 

- Po tej zmianie, na podglądzie/w widoku designera od razu będziemy mogli zweryfikować usytuowanie naszego layout'u

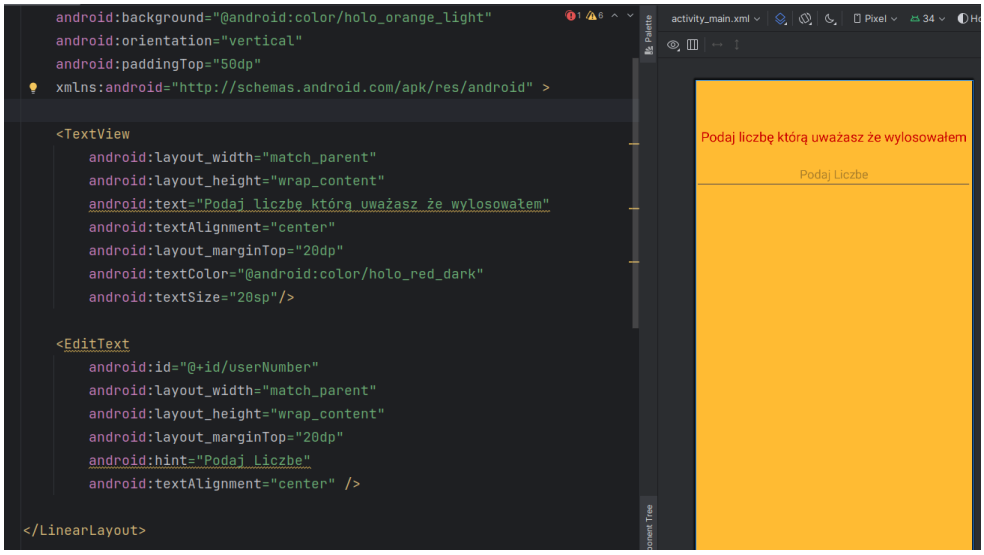
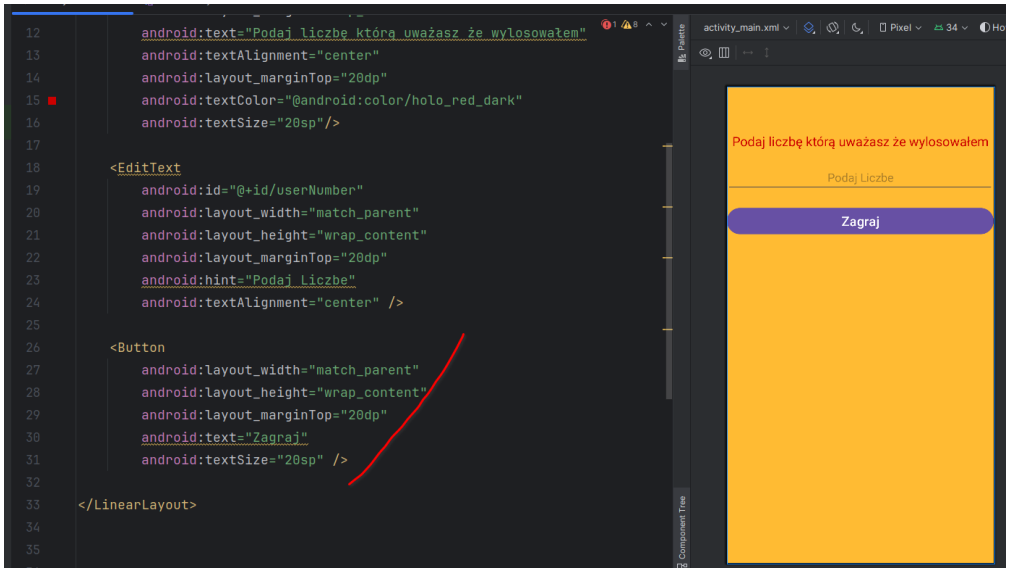
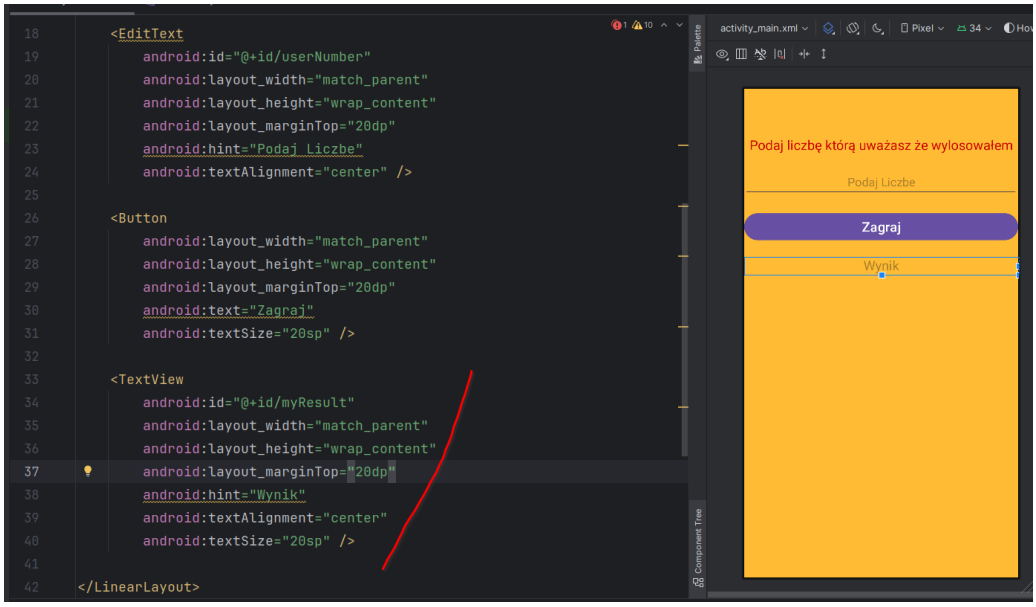
- 

- Z atrybutów obowiązkowych musimy jeszcze określić czy elementy mają być umieszczane obok siebie (orientacja pozioma), czy jeden pod drugim (orientacja pionowa). Dodajemy zatem orientation jako vertical
- I wreszcie, możemy odsunąć trochę umiejscowienie pierwszego elementu od górnej krawędzi ekranu ustawiając paddingTop np na 50dp.
- Finalnie, na tym etapie, definicja layoutu wygląda u mnie następująco:
- ```
  <?xml version="1.0" encoding="utf-8"?>
  <LinearLayout android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:background="@android:color/holo_orange_light"
      android:orientation="vertical"
      android:paddingTop="50dp"
      xmlns:android="http://schemas.android.com/apk/res/android" />
  ```
- To idźmy dalej
- Aktualnie, poza atrybutami opisującymi sam layout, nie bardzo mamy możliwość wstawienia do niego elementów interfejsu bowiem jest on w tej chwili otwierany, definiujemy mu jakieś właściwości i od razu go zamykamy
- 

- Żeby stworzyć sobie miejsce na dodanie do layoutu innych elementów musimy rozbić tą definicję na tag otwierający (zawierający definicję właściwości samego layout'u) i osobny tag zamykający.
- A zatem usuwamy znajdujący się tuż przed zamknięciem taga ukośnik (który oznacza że tutaj kończy się zakres tego layoutu, ,a w nowej linii zaczynamy pisać </
  ```

- Studio powinno domyślić się że chodzi o zakończenie rozpoczętej wcześniej definicji layoutu, uzupełni nam ten tag , i powinniśmy finalnie mieć sytuację, jak poniżej:

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

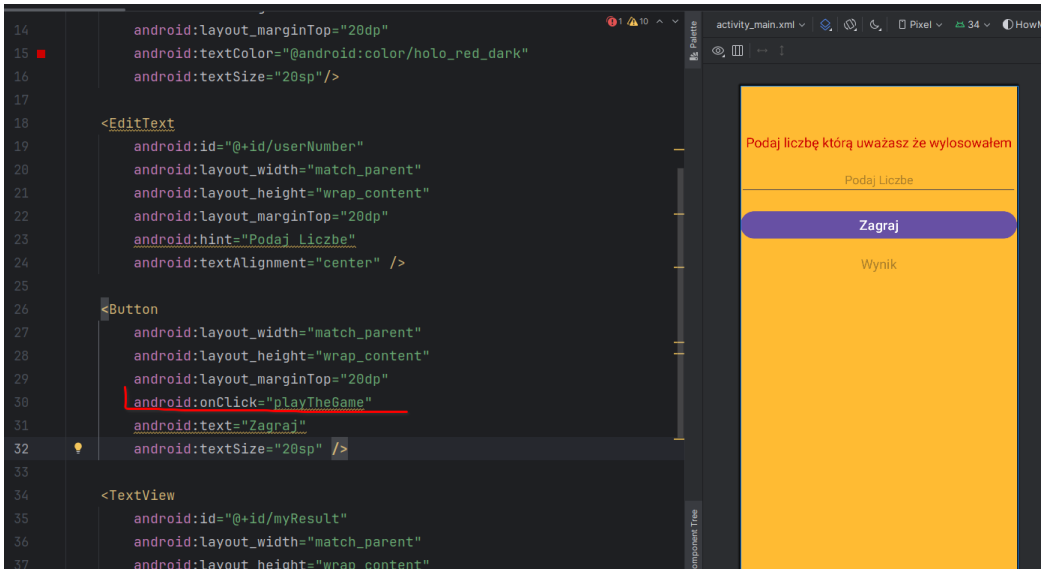
</LinearLayout>
```

- I teraz, pomiędzy tagiem początkowym a kończącym, możemy dodać kolejne elementy.
- Zacznijmy od dodania napisu “Podaj liczbę”, a zatem używamy elementu TextView np. jak poniżej:




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/holo_orange_light"
    android:orientation="vertical"
    android:paddingTop="50dp"
    xmlns:android="http://schemas.android.com/apk/res/android" >


    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Podaj liczbę którą uważasz że wylosowałem"
        android:textAlignment="center"
        android:layout_marginTop="20dp"
        android:textColor="@android:color/holo_red_dark"
        android:textSize="20sp"/>

</LinearLayout>
```

- W dalszej kolejności potrzebujemy miejsca na wpisanie przez użytkownika liczby którą uważa, że została przez aplikację wylosowana.
- Używamy do tego elementu EditText, np. tak jak poniżej:

- 

- Dodajmy teraz przycisk który uruchamiał będzie grę.
- Czyli używamy elementu Button, np. jak poniżej:
- 

- I wreszcie, dodajmy jeszcze jedno TextView w którym będziemy wyświetlać informację o wyniku gry (czy użytkownik odgadł czy też nie jaka liczba została wylosowana).
- Czyli np. coś takiego jak poniżej:

- 


```
    <EditText
        android:id="@+id/userNumber"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:hint="Podaj Liczbe"
        android:textAlignment="center" />

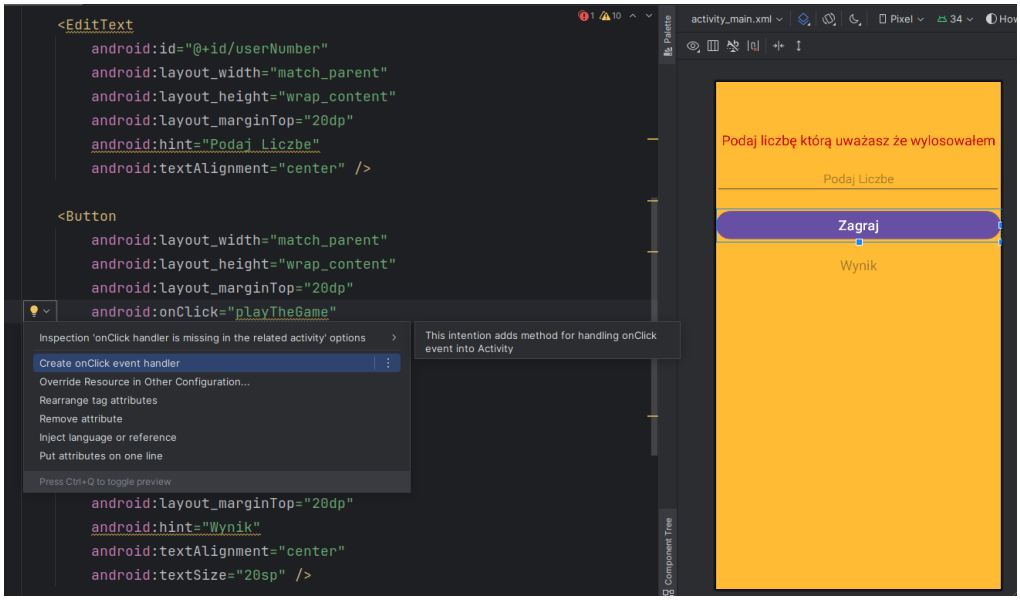
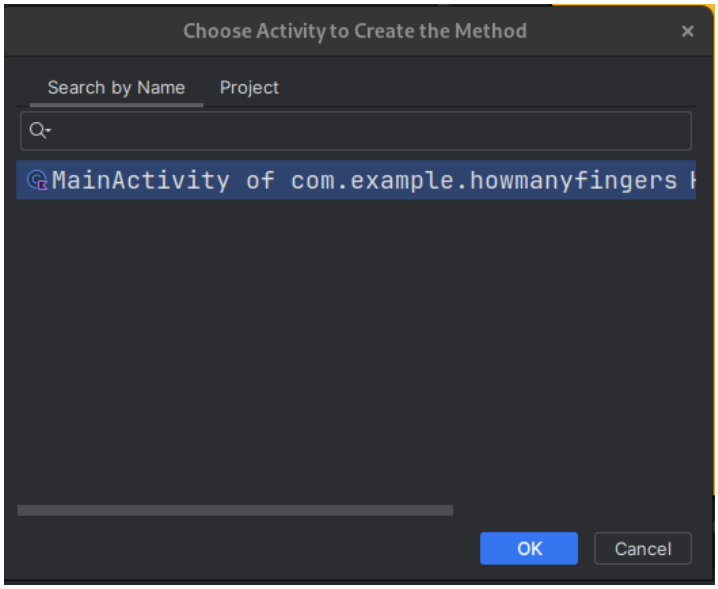
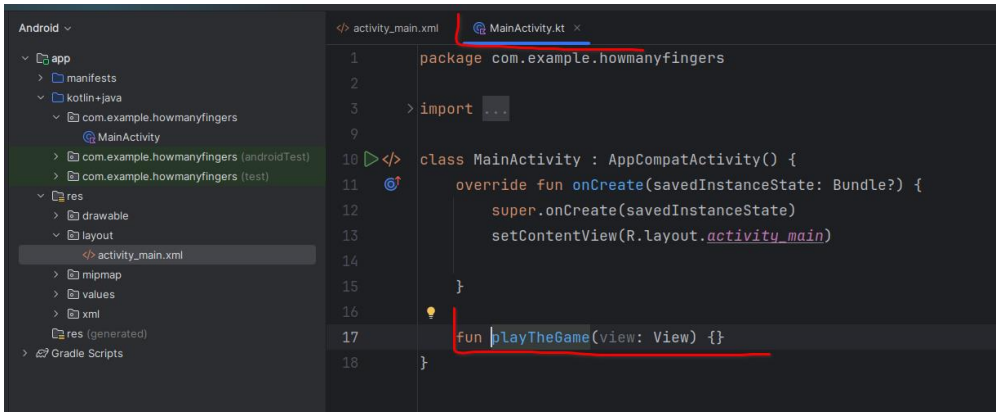
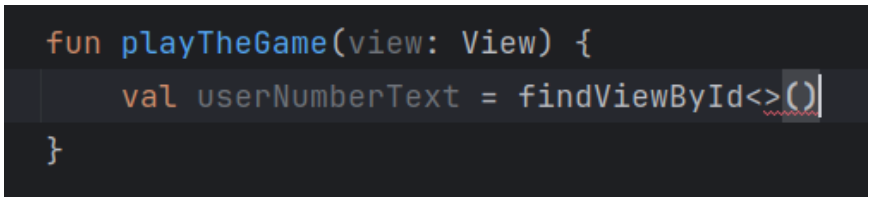
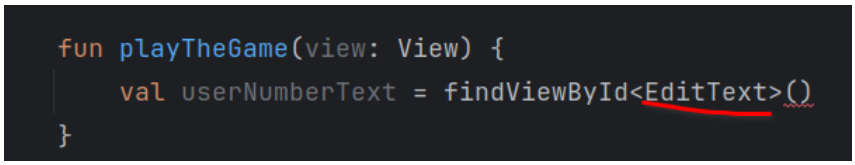
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="Zagraj"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/myResult"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:hint="Wynik"
        android:textAlignment="center"
        android:textSize="20sp" />

</LinearLayout>
```

- No to teraz możemy przystąpić do implementacji samej “rozgrywki”.
- W tym celu, dodajemy do definicji przycisku atrybut onClick definiując nazwę funkcji która ma się wykonać w momencie tapnięcia przez użytkownika na ten przycisk.
- Przyjmijmy że funkcja ta będzie nazywała się playTheGame, więć definicja atryburtu onClick przycisku wyglądać będzie jak poniżej:
- 


```
        android:layout_marginTop="20dp"
        android:textColor="@android:color/holo_red_dark"
        android:textSize="20sp"/>

    <EditText
        android:id="@+id/userNumber"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:hint="Podaj Liczbe"
        android:textAlignment="center" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:onClick="playTheGame"
        android:text="Zagraj"
        android:textSize="20sp" />

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

- No to teraz musimy zaimplementować naszą funkcję playTheGame.
- Na początek posłużmy się samym studio do wygenerowania szkieletu tej funkcji.
- Czyli np. będąc kursorem na nazwie playTheGame packamy alt+Enter i wybieramy z menu kontekstowego opcję Create event handler jak poniżej:

- 

- Studio zapyta nas w jakiej klasie ta funkcja powinna się znaleźć.
- Chcemy ją dodać do klasy która obsługuje logikę naszego interfejsu, czyli do klasy MainActivity
- W naszym przypadku, ponieważ w projekcie mamy tylko właśnie tą klasę, packamy na kolejnym okienku ok / enter
- 

- I aktualnie w pliku MainActivity.kt powinien pojawić się szkielet funkcji playTheGame (której ciało zaraz sobie doimplementujemy, a która będzie wywoływana przy każdym kliknięciu w nasz przycisk):

- 

- No to zacznijmy implementować ciało naszej funkcji.
- Zacznijmy od tego, że stworzymy sobie zmienną która reprezentować będzie pole tekstowe do którego użytkownik będzie wpisywał swój typ.
- Nazwijmy tę zmienną np. userNumberEText, a do sięgnięcia (i przypisania do niej) odpowiedniego elementu interfejsu użytkownika użyjemy funkcji findViewById czyli
- 

- Funkcja findViewById, oczekuje, że (w ostrych nawiasach) podamy typ elementu do którego “siegamy”. W naszym przypadku jest to EditText, zatem dodajemy:
- 

- I wreszcie, w nawiasach okrągłych musimy podać id elementu do którego chcemy się odwołać.
- Skąd “wziąć” to id? Ano musimy je sobie zdefiniować / utworzyć.
- Żeby to zrobić:
    - Wędrujemy do pliku xml z layout’em tam gdzie mamy zdefiniowany element do którego chcemy się w kodzie odwołać (w naszym przypadku pole do wprowadzania przez użytkownika liczby którą uważa że aplikacja wylosowała) i dodajemy coś takiego:

```
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Podaj liczbę którą uważasz że wylosowałem"
        android:textAlignment="center"
        android:layout_marginTop="20dp"
        android:textColor="@android:color/holo_red_dark"
        android:textSize="20sp"/>

    <EditText
        android:id="@+id/userNumber"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:hint="Podaj Liczbe"
        android:textAlignment="center" />
```

- userNumber - to nadawana przez nas nazwa po której będziemy się do tego pola odwoływać w kodzie, a dzięki dyrektywie @+id SDK/Android wygeneruje sobie na tej podstawie swoje wewnętrzne id/Id w swoim formacie.

- No to teraz wracamy do naszej funkcji playTheGame i jako id elementu do ktorego chcemy sie odwolac w funkcji findViewbyId podajemy R.id.userNumber czyli finalnie definicja tej zmiennej userNumberEText wyglada u mnie następująco:

```
import androidx.core.view.WindowInsetsCompat

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

    }

    fun playTheGame(view: View) {
        val userNumberEText = findViewById<EditText>(R.id.userNumber)
    }
}
```

- To idąc dalej potrzebujemy zczytac podaną przez użytkownika wartość i przekonwertować ją do liczby (bo za sekundę bedziemy ja porównywać z wylosowaną przez aplikację wartością liczbową, a wartość którą zczytamy z pola EditText dostaniemy jako String. W moim przypadku zrobiłem to jak poniżej:

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

    }

    fun playTheGame(view: View) {
        val userNumberEText = findViewById<EditText>(R.id.userNumber)
        val userNumberValue =  Integer.parseInt(userNumberEText.getText().toString())

    }
```

- Nie jest to może zbyt piękne, i potencjalnie bedzie się “wywalało” (bo co jak użytkownik zamiast wpisać do pola grzecznie np 5 czy 10 wpisze “lato”... Ale “uszczelnienie” tego pozostawiam juz Panstwu :)
- No to teraz spróbujmy wygenerowac jakąś wartość losową. Na poczatek, żeby łatwiej było “trafić” (co przyda się przy debugowaniu” :) losujmy może z jakiegoś mniejszego zakresu typu 1-5.
- U siebie zrobiłem to następująco:

```
        }

        fun playTheGame(view: View) {
            val userNumberEText = findViewById<EditText>(R.id.userNumber)
            val userNumberValue =  Integer.parseInt(userNumberEText.getText().toString())

            val randomNumber = (1 ≤ .. ≤ 5).random()

        }
    }
```

- No to teraz, mając obie wartości, możemy je porównać i sprawdzić czy użytkownik się wstrzelił:

```
    fun playTheGame(view: View) {
        //Siegnijmy do pola tekstowego w ktorym uzytkownik bedzie wpisywal swoje Typy
        val userNumberEText = findViewById<EditText>(R.id.userNumber)

        //I zczytajmy stamtad podana przez uzytkownika wartosc,
        // konwertujac jaa od razu do wartosci liczbowej
        val userNumberValue =  Integer.parseInt(userNumberEText.getText().toString())

        //Wygenerujmy w imieniu aplikacji jakas liczbe
        val randomNumberValue = (1 ≤ .. ≤ 5).random()


        //I sprawdzmy czy uzytkownik odgadl
        if (userNumberValue == randomNumberValue){

        } else {

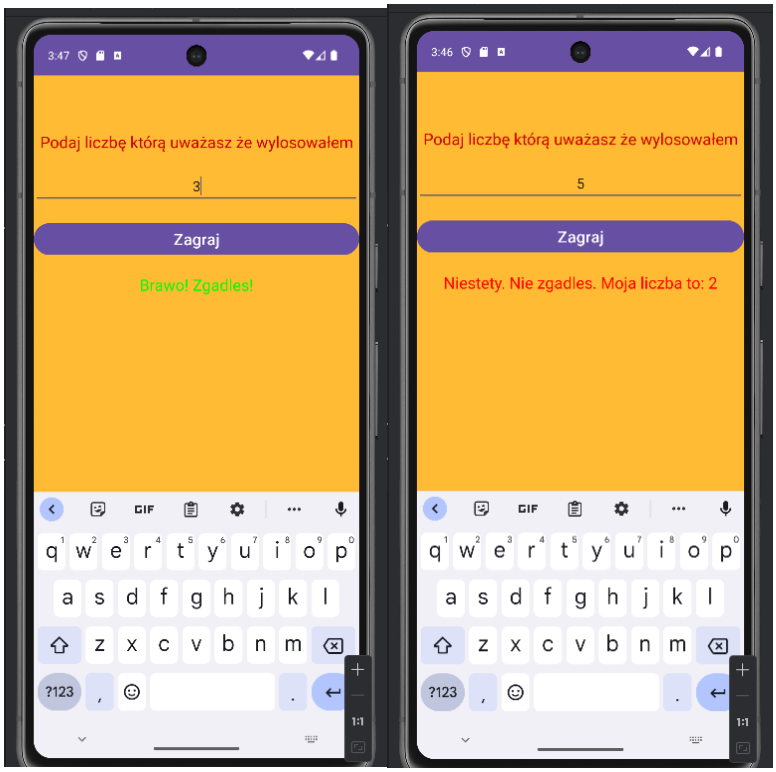
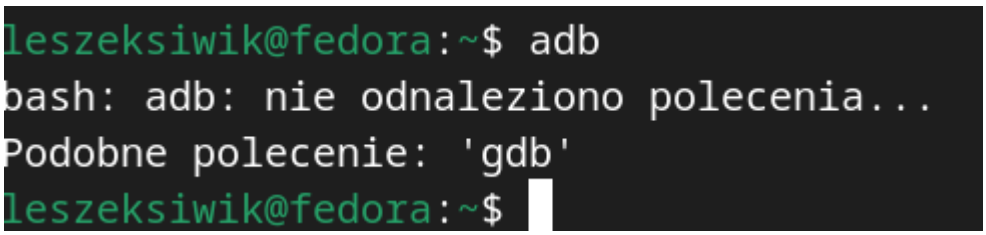
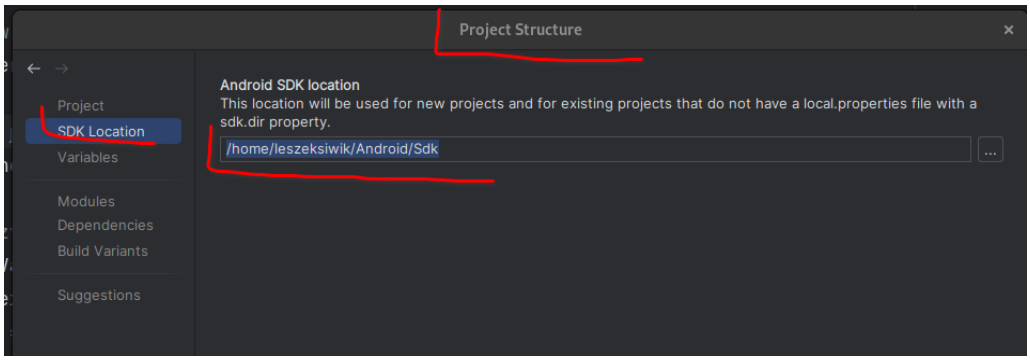
        }


    }
```

- Żeby odpowiednio obsłużyć oba przypadki, chcielibyśmy wyświetlić użytkownikowi informacje o tym czy odgadł czy nie.
- Żeby to zrobić, będziemy potrzebowali odwołać się z naszego kodu do elementu interfejsu użytkownika gdzie mamy wypisać odpowiednią informację.
- A zatem, podobnie jak wcześniej, tak i teraz posłużymy się funkcją findViewById - ale wiemy już, że żeby to zrobić musimy wygenerować / dodać atrybut id do pola do którego chcemy sie odwołać
- No więc wędrujemy do naszego layout'u i dla elementu gdzie mamy wyświetlać wynik dodajemy atrybut id, np. jak ponizej:

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:onClick="playTheGame"
    android:text="Zagraj"
    android:textSize="20sp" />

<TextView
    android:id="@+id/resTV"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:hint="Wynik"
    android:textAlignment="center"
    android:textSize="20sp" />

/LinearLayout>
```

- I mając to możemy dodać w kodzie wyświetlanie odpowiedniej informacji:

```
//Wygenerujmy w imieniu aplikacji jakas liczbe
val randomNumberValue = (1 ≤ .. ≤ 5).random()

//Siegnijmy do pola w ktorym mamy wypisac informacje czy uzytkownik odgadl czy nie
val resTV = findViewById<TextView>(R.id.resTV)

//Sprawdzmy czy uzytkownik odgadl i wyswietlmy odpowiednia informacje
if (userNumberValue == randomNumberValue) {
    resTV.setTextColor(Color.GREEN)
    resTV.text = "Brawo! Zgadles!"
} else {
    resTV.setTextColor(Color.RED)
    resTV.text = "Niestety. Nie zgadles. Moja liczba to: " + randomNumberValue.toString()
}
```

- No i jako pierwsze przybliżenie, na razie tyle.
- Spróbujmy uruchomić naszą aplikację, sprawdzić że / czy działa i naprawić ewentualne błędy.

- 

- Moglibyśmy tą naszą aplikację jeszcze długo rozwijać, ale ponieważ nie to jest zasadniczym celem naszego ćwiczenia, uznajmy że (na razie) to tyle i przejdźmy do “clue” - czyli próby “zhackowania” tej naszej aplikacji
- Spróbujemy zatem zroobić tak, te, że użytkownik będzie wygrywał niezależnie od tego jaką liczbę wpisze (moze nie jest to zbyt spektakularne / nie wydaje sie bardzo niebezpieczne, ale co, gdyby na podobnej zasadzie aplikacja “wpuszczała” do środka niezależnie od tego jakie hasło zostałoby podane?
- A tak czy inaczej chodzi o przećwiczenie pewnego schematu, a w takim przypadku, im na początek prościej tym lepiej 😳
- Zacznijmy od upewnienia się, że mamy “pod ręką” narzędzie adb, które pozwoli m.in. na wyciągnięcie z urządzenia, zainstalowanej tam aplikacji
- Tak więc otwórzmy terminal systemowy, wpiszmy adb i pacnijmy enter.
- Jeśli dostaniemy coś takiego jak poniżej:
- 

- To najprawdopodobniej musimy dodać do ścieżki systemowej katalog w którym ‘rezyduje” adb
- Żeby dodać coś do ścieżki / zmiennej systemowej na Linuxach/Macach robimy to np. w następujący sposób:

- ***export PATH=$PATH:YourAndroidSDKPath/platform-tools/***
- Lokalizację / ścieżkę do SDK możemy znaleźć np. w AndroidStudio: File - Project Structure -> SDK Location
- 

- Tak więc w moim przypadku ustawienie ścieżki będzie wyglądało następująco:
- ***export PATH=$PATH:/home/leszeksiwik/Android/Sdk/platform-tools/***
- I po tym kroku, adb powinno nam się już “zgłosić”:
- ```
leszeksiwik@fedora:~$ adb
bash: adb: nie odnaleziono polecenia...
leszeksiwik@fedora:~$ export PATH=$PATH:/home/leszeksiwik/Android/Sdk/platform-tools/
leszeksiwik@fedora:~$ adb
Android Debug Bridge version 1.0.41
Version 35.0.1-11580240
Installed as /home/leszeksiwik/Android/Sdk/platform-tools/adb
Running on Linux 6.8.9-200.fc39.x86_64 (x86_64)

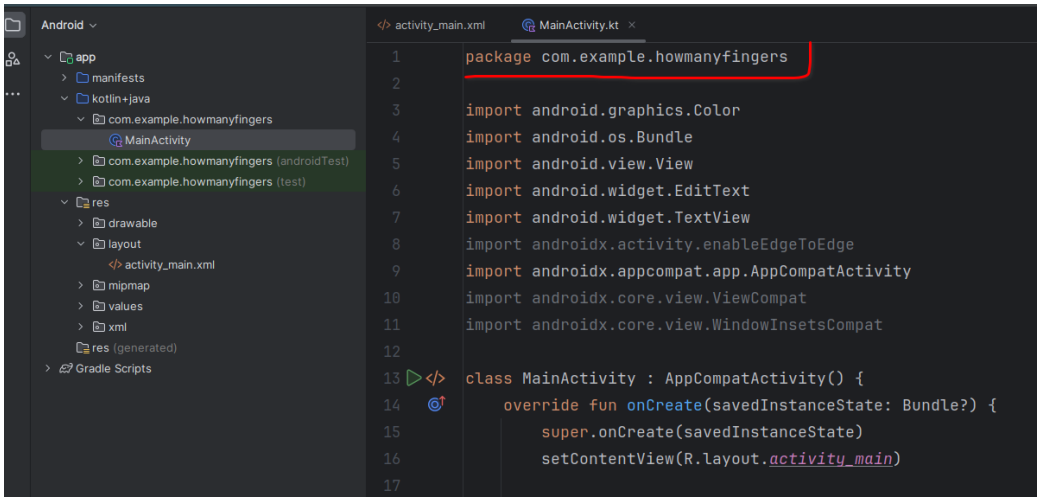
global options:
 -a   listen on all network interfaces, not just localhost
 -d   use USB device (error if multiple devices connected)
 -e   use TCP/IP device (error if multiple TCP/IP devices available)
 -s SERIAL   use device with given serial (overrides $ANDROID_SERIAL)
 -t ID   use device with given transport id
 -H   name of adb server host [default=localhost]
 -P   port of adb server [default=5037]
 -L SOCKET   listen on given socket for adb server [default=tcp:localhost:5037]
 --one-device SERIAL|USB only allowed with 'start-server' or 'server nodaemon', server will only con
nect to one USB device, specified by a serial number or USB device address.
 --exit-on-write-error   exit if stdout is closed
```
- To wykorzystamy teraz adb żeby zobaczyć co / jakie pakiety / aplikacje mamy zainstalowane na naszym urządzeniu.
- Czyli:
    - adb shell pm list packages
- i powinniśmy dostać listę wszyskich pakietów (czyli de facto aplikacji) zainstalowanych na urządzeniu:
```

```
leszeksiwik@fedora:~$ adb shell pm list packages
package:com.android.systemui.auto_generated_rro_vendor__
package:com.google.android.providers.media.module
package:com.google.android.overlay.permissioncontroller
package:com.google.android.overlay.googlewebview
package:com.android.calllogbackup
package:com.android.carrierconfig.auto_generated_rro_vendor__
package:com.android.systemui.accessibility.accessibilitymenu
package:com.android.internal.emulation.pixel_3_xl
package:com.android.providers.contacts
package:com.android.internal.emulation.pixel_4a
package:com.android.dreams.basic
package:com.android.companiondevicemanager
package:com.android.cts.priv.ctsshim
package:com.google.android.calendar
package:com.google.android.networkstack.tethering.emulator
package:com.google.android.contacts
package:com.android.mms.service
package:com.google.android.cellbroadcastreceiver
package:com.android.providers.downloads
package:com.android.bluetoothmidiservice
package:com.android.credentialmanager
package:com.google.android.printservice.recommendation
package:com.google.android.captiveportallogin
package:com.android.storagemanager.auto_generated_rro_product__
```

- To spróbujmy zobaczyć, czy mamy tam zainstalowaną grę/pakiet howmanyfingers.
- W tym celu, możemy użyć np. narzędzia grep żeby przefiltrować sobie listę tych wszystkich dostępnych na urządzeniu pakietów po tym czego szukamy:

```
package:com.google.android.youtube
package:com.android.simappdialog.auto_generated_rro_product__
package:com.android.externalstorage
package:com.android.server.telecom
leszeksiwik@fedora:~$
leszeksiwik@fedora:~$ adb shell pm list packages |grep howmanyfingers
package:com.example.howmanyfingers
leszeksiwik@fedora:~$
```

- No i coś mamy, a cały pakiet nazywa się com.example.howmanyfingers.
- Swoją drogą jest to dokładnie ta nazwa pakietu którą ustawiliśmy tworząc aplikację:



- Ale to tylko na marginesie, żeby wiedzieć skąd sie to bierze 😳
- Mając nazwę pakietu możemy sprawdzić dokładną ścieżkę pod jaką pakiet ten jest zainstalwany na urządzeniu, tak więc:

```
leszeksiwik@fedora:~$
leszeksiwik@fedora:~$ adb shell pm list packages |grep howmanyfingers
package:com.example.howmanyfingers
leszeksiwik@fedora:~$ adb shell pm path com.example.howmanyfingers
package:/data/app/~~S03_apZ6a8JKi7yCxMYurg==/com.example.howmanyfingers-mlQDBR-vGFV1fRXo_VTGRQ==/base
.apk
leszeksiwik@fedora:~$
```

- A mając pełną scieżkę, możemy tą aplikację / ten pakiet sciągnąć z urządzenia.
- Żeby łatwiej się nam dalej pracowało, stwórzmy sobie może jakis (pod)katalog w naszym katalogu domowym żebyśmy później tej aplikacji nie szukali po całym systemie :), i tam sobie tę naszą aplikacje ściągniemy.
- Czyli odpowiednio:
  - upewnijmy sie czy/że jesteśmy w katalogu domowym
    - pwd

```
leszeksiwik@fedora:/$ pwd
/
leszeksiwik@fedora:/$
```

  - a jeśli nie, np. jak ja powyżej:
  - przejdźmy do swojego home'a - czyli np. po prostu cd

```
leszeksiwik@fedora:/$ pwd
/
leszeksiwik@fedora:/$ cd
leszeksiwik@fedora:~$
```

  - i po tym powinniśmy być już w swoim homie

```
leszeksiwik@fedora:/$ cd
leszeksiwik@fedora:~$ pwd
/home/leszeksiwik
leszeksiwik@fedora:~$
```

- Stwórzmy sobie teraz katalog pod dalsze działania. Niech się nazywa howmanyfingershacking :) czyli:
  - `leszeksiwik@fedora:~$ mkdir howmanyfingershacking`
- Przejdźmy tam sobie:
  - ```
    leszeksiwik@fedora:~$ cd howmanyfingershacking/
    leszeksiwik@fedora:~/howmanyfingershacking$
    ```
- I możemy działać dalej :)

- No to ściągnijmy z urządzenia ten pakiet / aplikację howmany fingers, czyli:
  - ```
    leszeksiwik@fedora:~/howmanyfingershacking$ adb pull /data/app/~~S03_apZ6a8JKi7yCxMYurg==/com.example
    .howmanyfingers-mlQDBR-vGFV1fRXo_VTGRQ==/base.apk
    /data/app/~~S03_apZ6a8JKi7yCxMYurg==/com.example...., 0 skipped. 79.7 MB/s (12995464 bytes in 0.156s)
    leszeksiwik@fedora:~/howmanyfingershacking$
    ```
- gdzie scieżka którą podałem jako argument wywołanią, to jest dokładnie to co zwróciło nam wywołanie
  - adb shell pm path com.example.howmanyfingers
- czyli:
  - ```
    leszeksiwik@fedora:~/howmanyfingershacking$ adb shell pm path com.example.howmanyfingers
    package:/data/app/~~S03_apZ6a8JKi7yCxMYurg==/com.example.howmanyfingers-mlQDBR-vGFV1fRXo_VTGRQ==/base.apk
    ```
- Jeśli wszystko przebieglo ok, w naszym katalogu który wcześniej sobie założyliśmy, powinien pojawić się plik base.apk z aplikacją którą chcemy (niecnie :) zmodyfikować:
  - ```
    leszeksiwik@fedora:~/howmanyfingershacking$ adb pull /data/app/~~S03_apZ6a8JKi7yCxMYurg==/com.example.h
    1fRXo_VTGRQ==/base.apk
    /data/app/~~S03_apZ6a8JKi7yCxMYurg==/com.example...., 0 skipped. 79.7 MB/s (12995464 bytes in 0.156s)
    leszeksiwik@fedora:~/howmanyfingershacking$ ls -al
    razem 12692
    drwxr-xr-x. 1 leszeksiwik leszeksiwik       16 05-23 12:59 .
    drwx------. 1 leszeksiwik leszeksiwik      680 05-23 12:55 ..
    -rw-r--r--. 1 leszeksiwik leszeksiwik 12995464 05-23 12:59 base.apk
    ```
- No to spróbujmy dobrać się do tej aplikacji i ją zdekompilować.
- Posłużymy się w tym celu narzędziem apktool.
- Wędrujemy zatem na stronę: https://apktool.org/docs/install/ i zgodnie z instrukcją, dla systemów linux'owych:

Linux

1. Download the Linux wrapper script. (Right click, Save Link As `apktool`)
2. Download the latest version of Apktool.
3. Rename the downloaded jar to `apktool.jar`.
4. Move both `apktool.jar` and `apktool` to `/usr/local/bin`. (root needed)
5. Make sure both files are executable. (`chmod +x`)
6. Try running `apktool` via CLI.

-
- Ściągamy zatem (do katalogu który sobie wcześniej przygotowaliśmy):
    - wrapper linuxowy
    - oraz najnowszą wersję samego tool'a (aktualnie jest to wersja 2.9.3
- Po tych dwóch krokach, zawartość naszego "hackerskiego" katalogu powinna być następująca (lub zbliżona):
    -
```
leszeksiwik@fedora:~/howmanyfingershacking$ ls -al
razem 35408
drwxr-xr-x. 1 leszeksiwik leszeksiwik       72 05-23 13:10 .
drwx------. 1 leszeksiwik leszeksiwik      680 05-23 12:55 ..
-rw-r--r--. 1 leszeksiwik leszeksiwik 23254968 05-23 13:10 apktool_2.9.3.jar
-rw-r--r--. 1 leszeksiwik leszeksiwik     1156 05-23 13:08 apktool.bat
-rw-r--r--. 1 leszeksiwik leszeksiwik 12995464 05-23 12:59 base.apk
```
- To idąc dalej, zgodnie z instrukcją instalacji apktool'a, "przemianowujemy" apktool_2.9.3.jar na apktool.jar czyli:
    -
```
leszeksiwik@fedora:~/howmanyfingershacking$ mv apktool_2.9.3.jar apktool.jar
```
- A z kolei apktool.bat przemianowujemy na apktool (jeśli nie zrobiliśmy tego na etapie ściągania tego skryptu), czyli:
-
```
leszeksiwik@fedora:~/howmanyfingershacking$ mv apktool.bat apktool
leszeksiwik@fedora:~/howmanyfingershacking$
```
- I, wreszcie, dodajemy tym ściagniętym plikom prawa wykonywania, czyli np:
    -
```
drwxr-xr-x. 1 leszeksiwik leszeksiwik       52 05-23 13:31 .
drwx------. 1 leszeksiwik leszeksiwik      680 05-23 12:55 ..
-rw-r--r--. 1 leszeksiwik leszeksiwik     1156 05-23 13:08 apktool
-rw-r--r--. 1 leszeksiwik leszeksiwik 23254968 05-23 13:10 apktool.jar
-rw-r--r--. 1 leszeksiwik leszeksiwik 12995464 05-23 12:59 base.apk
leszeksiwik@fedora:~/howmanyfingershacking$ chmod +x apktool*
```
- Więc finalnie, w na tym etapie w naszym "hackerskim" katalogu powinniśmy mieć sytuację następującą:

```
leszeksiwik@fedora:~/howmanyfingershacking$ ls -al
razem 35408
drwxr-xr-x. 1 leszeksiwik leszeksiwik       52 05-23 13:31 .
drwx------. 1 leszeksiwik leszeksiwik      680 05-23 12:55 ..
-rwxr-xr-x. 1 leszeksiwik leszeksiwik     1156 05-23 13:08 apktool
-rwxr-xr-x. 1 leszeksiwik leszeksiwik 23254968 05-23 13:10 apktool.jar
-rw-r--r--. 1 leszeksiwik leszeksiwik 12995464 05-23 12:59 base.apk
```

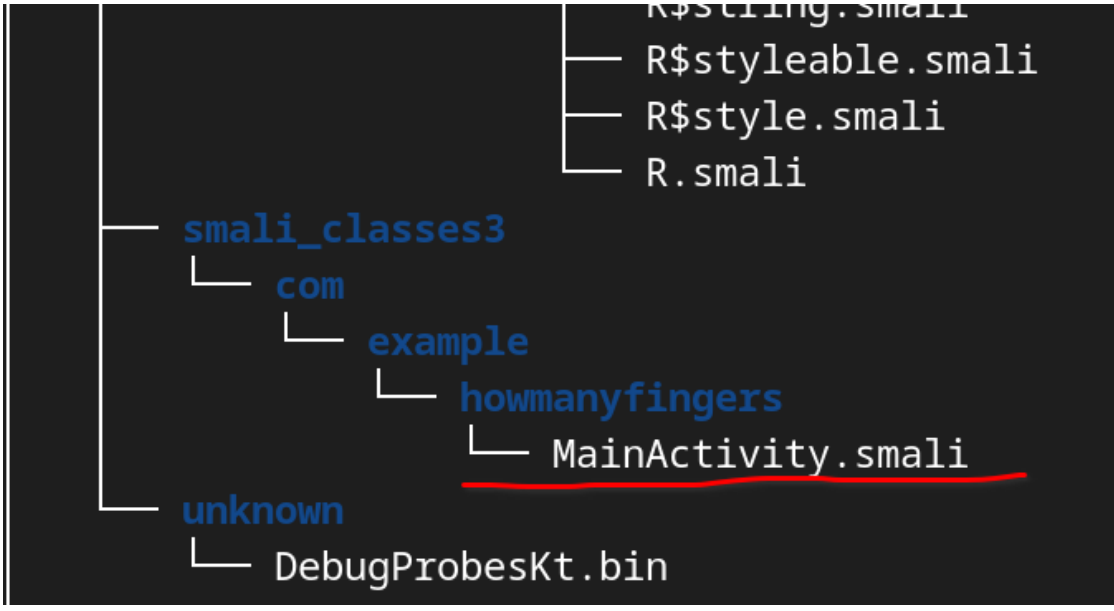
- No to możemy przystąpić do dekompilacji naszej aplikacji, czyli:

```
leszeksiwik@fedora:~/howmanyfingershacking$ ./apktool d base.apk
```

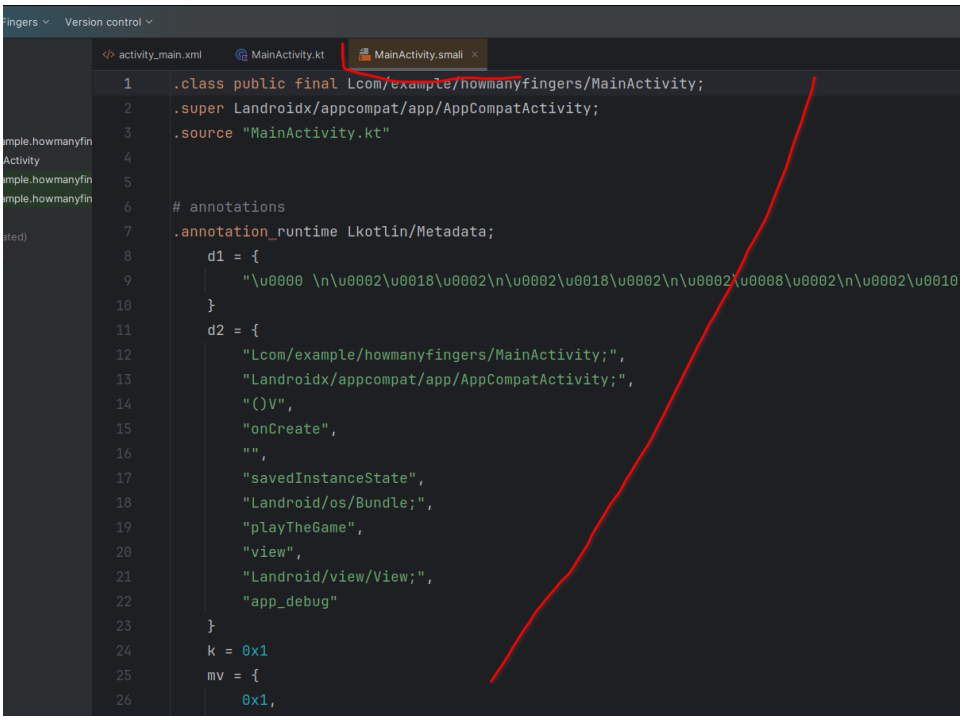
- Po tym kroku (i serii komunikatów na ekranie) w katalogu powinnien pojawić nam się katalog base, które zawiera naszą zdekompilowaną aplikację:

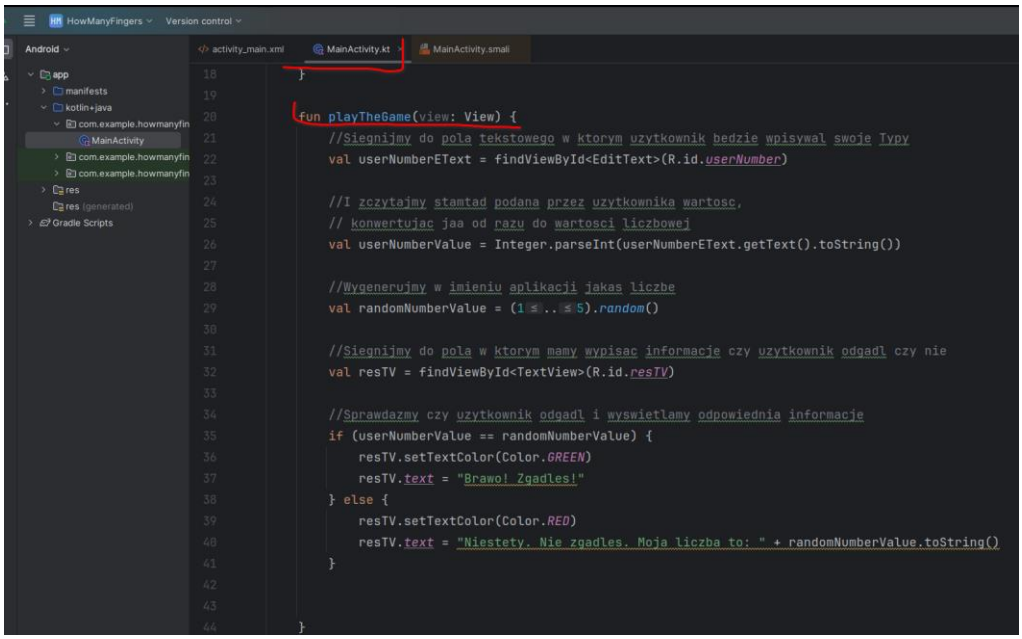
```
leszeksiwik@fedora:~/howmanyfingershacking$ ls -al
razem 35412
drwxr-xr-x. 1 leszeksiwik leszeksiwik       66 05-23 13:42 .
drwx------. 1 leszeksiwik leszeksiwik      680 05-23 12:55 ..
-rwxr-xr-x. 1 leszeksiwik leszeksiwik     1156 05-23 13:08 apktool
-rwxr-xr-x. 1 leszeksiwik leszeksiwik 23254968 05-23 13:10 apktool.jar
drwxr-xr-x. 1 leszeksiwik leszeksiwik      190 05-23 13:42 base
-rw-r--r--. 1 leszeksiwik leszeksiwik 12995464 05-23 12:59 base.apk
```

- Powstały w wyniku dekompilacji quasi-asemblerowy kod głównej aktywności naszej aplikacji umieszczony jest w podkatalogu smali_classes3/com/example/howmanyfingers/MainActivity.smali powstałego katalogu base

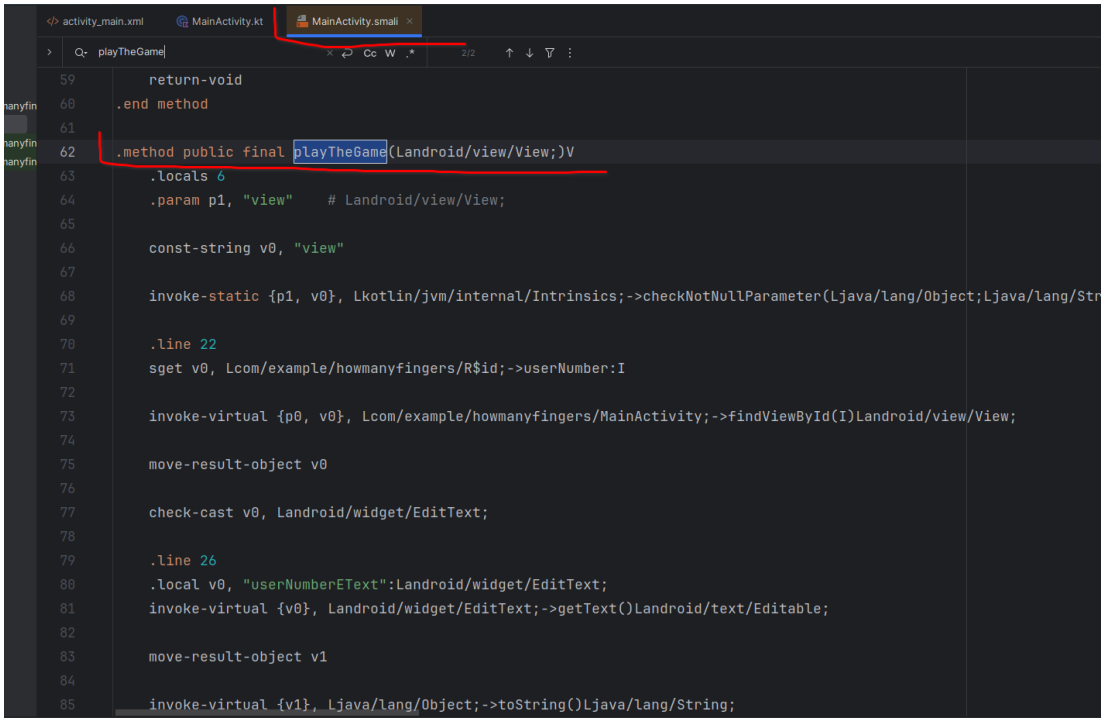


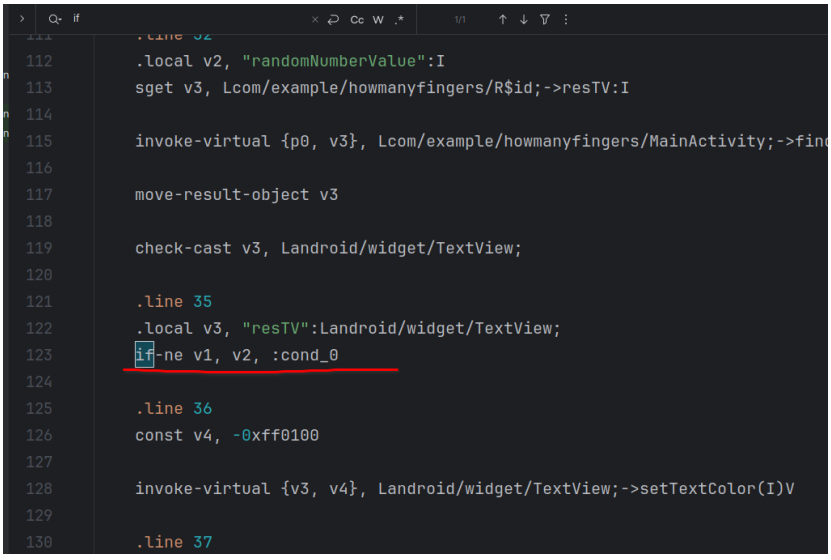
- Otwórzmy sobie ten plik w jakimś edytorze, np w AndroidStudio, czyli File->Open… i wędrujemy do naszego zdekompilowanego pliku.
- Po otwarciu powinniśmy zobaczyć coś takiego:

- 

- No i teraz już nic prostszego jak zmodyfikowanie/zainfekowanie tego kodu :)
- Tak jak sobie powiedzieliśmy wcześniej, to co chcemy uzyskać w wyniku naszej “niecnej” modyfikacji, to takie działanie aplikacji, zebyśmy “wygrywali” niezależnie od tego jaką liczbę podamy. No więc ewidentnie musimy “popsuć” działanie metody playTheGame która, dla przypomnienia, z perspektywy dewelopera/”normalnego kodu” wygląda następująco:
- 

- No to poszukajmy metody playTheGame w zdekompiklowanym pliku .smali
- W efekcie powinniśmy znaleźć miejsce gdzie zaczyna się zdekompilowany kod tej metody:

- 

- To, czym bylibyśmy szczególnie zainteresowani, to moment sprawdzania czy liczba wylosowana przez aplikację i liczba podana przez użytkownika są soobie równe czy nie. No to poszukajmy w tym zdekompilwanym kodzie instrukcji if.
- Jak zaczniemy szukac, okaże się, że jest taka jedna:
- 

- Która oznacza ni mniej ni więcej tylko tyle, że jeśli v1 nie równa się v2 (if-ne czyli if-not equal) wtedy przejdź do etykiety cond_0
- Jak popatrzymy co się dzieje pod etykietą :cond_0, to w największym skrócie mamy tam fragment kodu wykonywany w sytuacji kiedy użytkownik “nie odgadł” czyli zmiana koloru czcionki, wyświetlenie komunikatu że użytkownik nie odgadł (i dodatkowo jaką liczbę wylosowała sama aplikacja):

```
    :cond_0
    const/high16 v4, -0x10000

    invoke-virtual {v3, v4}, Landroid/widget/TextView;->setTextColor(I)V

    .line 40
    new-instance v4, Ljava/lang/StringBuilder;

    invoke-direct {v4}, Ljava/lang/StringBuilder;-><init>()V

    const-string v5, "Niestety. Nie zgadles. Moja liczba to: "

    invoke-virtual {v4, v5}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/Str

    move-result-object v4

    invoke-virtual {v4, v2}, Ljava/lang/StringBuilder;->append(I)Ljava/lang/StringBuilder;

    move-result-object v4

    invoke-virtual {v4}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

    move-result-object v4

    check-cast v4, Ljava/lang/CharSequence;

    invoke-virtual {v3, v4}, Landroid/widget/TextView;->setText(Ljava/lang/CharSequence;)V

    .line 44
    :goto_0
    return-void
.end method
```

- Po czym nie ma tam już żadnych instrukcji sterujących tylko kończymy metodę
- No dobrze, a przyjrzyjmy się jeszcze raz tej instrukcji odpowiedzialnej za porównanie tych dwóch liczb: wylosowanej przez aplikację i podanej przez uzytkownika:

```
110
111     .line 32
112     .local v2, "randomNumberValue":I
113     sget v3, Lcom/example/howmanyfingers/R$id;->resTV:I
114
115     invoke-virtual {p0, v3}, Lcom/example/howmanyfingers/MainActivity;->findViewById(I)Landroid/view/View;
116
117     move-result-object v3
118
119     check-cast v3, Landroid/widget/TextView;
120
121     .line 35
122     .local v3, "resTV":Landroid/widget/TextView;
123     if-ne v1, v2, :cond_0
124
125     .line 36
126     const v4, -0xff0100
127
128     invoke-virtual {v3, v4}, Landroid/widget/TextView;->setTextColor(I)V
129
130     .line 37
131     const-string v4, "Brawo! Zgadles!"
132
133     check-cast v4, Ljava/lang/CharSequence;
134
135     invoke-virtual {v3, v4}, Landroid/widget/TextView;->setText(Ljava/lang/CharSequence;)V
136
137     goto :goto_0
138
```

- I przeanalizujmy jeszcze raz, jak to działa. Wiemy już, że jesli v1 i v2 nie są sobie równe to skaczemy do etykiedy cond_0.
- A co jeśli ten warunek nie jest spełniony? Czyli jesli v1 i v2 są sobie równe?
- No to wtedy nie skaczemy do :cond_0 tylko wykonujemy kolejne linijki kodu gdzie ewidentnie mamy obsługę sytuacji kiedy użytkownik odgadł wylosowaną liczbę, czyli ustawienie komunikatu dla uzytkownika na “Brawo zgadłeś” etc. Po czym mamy wywołaną instrukcję skoku (goto - linijka 137 na poprzednim zrzucie) do etykiety :goto_0
- A jak zerkniemy na to co dzieje się pod etykietą goto_0j:

```
158     move-result-object v4
159
160     invoke-virtual {v4}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;
161
162     move-result-object v4
163
164     check-cast v4, Ljava/lang/CharSequence;
165
166     invoke-virtual {v3, v4}, Landroid/widget/TextView;->setText(Ljava/lang/CharSequence;)V
167
168     .line 44
169     :goto_0
170     return-void
171 .end method
172
```

- Ano nic. W sensie ustawiamy zwracany z metody tym na voida i kończymy metodę
- Innymi słowy, gdybyśmy np. wyeliminowali z kodu ten warunek sprawdzania/porównywania liczb:

```
    invoke-virtual {p0, v3}, Lcom/example/howmanyfingers/MainActivity;->findViewById(I)Landroid

    move-result-object v3

    check-cast v3, Landroid/widget/TextView;

    .line 35
    .local v3, "resTV":Landroid/widget/TextView;
    if-ne v1, v2, :cond_0

    .line 36
    const v4, -0xff0100

    invoke-virtual {v3, v4}, Landroid/widget/TextView;->setTextColor(I)V

    .line 37
    const-string v4, "Brawo! Zgadles!"

    check-cast v4, Ljava/lang/CharSequence;

    invoke-virtual {v3, v4}, Landroid/widget/TextView;->setText(Ljava/lang/CharSequence;)V

    goto :goto_0

    .line 39
    :cond_0
    const/high16 v4, -0x10000
```

- to aplikacja nie przeskoczy do etykiety :cond_0 i nie będzie obsługiwać sytuacji w ktrej użytkownik nie odgadł wylosowanej liczby, tylko pójdzie dalej, wyświetli komunikat o wygranej i skoczy to etykiety goto_0 czyli zakończy wykonywanie metody playTheGame
- No to spróbujmy “wyłączyć” to sprawdzanie, czyli w najprostszy sposób zakomentujmy tą linijkę z instrukcją if-ne. Znak komentarza w kodzie smali to # czyli robimy:

```
    invoke-virtual {p0, v3}, Lcom/example/howmanyfingers/MainActivity;->findViewById(I)Landro

    move-result-object v3

    check-cast v3, Landroid/widget/TextView;

    .line 35
    .local v3, "resTV":Landroid/widget/TextView;
#    if-ne v1, v2, :cond_0

    .line 36
    const v4, -0xff0100

    invoke-virtual {v3, v4}, Landroid/widget/TextView;->setTextColor(I)V

    .line 37
    const-string v4, "Brawo! Zgadles!"

    check-cast v4, Ljava/lang/CharSequence;

    invoke-virtual {v3, v4}, Landroid/widget/TextView;->setText(Ljava/lang/CharSequence;)V
```

- No to spróbujmy zbudować z powrotem “paczkę” (czyli plik .apk) z naszą (zhackowaną) aplikacją i zobaczymy co nam to wszystko da.
- Żeby zbudować ponownie paczkę potrzebujemy ponownie użyć narzędzia apktool tylko z trochę innymi opcjami, a mianowicie: ./apktool b -f -d base

```
leszeksiwik@fedora:~/howmanyfingershacking$ java -jar apktool.jar b -f -d base
I: Using Apktool 2.9.3
I: Smaling smali folder into classes.dex...
I: Smaling smali_classes3 folder into classes3.dex...
I: Smaling smali_classes2 folder into classes2.dex...
I: Building resources...
I: Using aapt2 - setting 'debuggable' attribute to 'true' in AndroidManifest.xml
I: Copying libs... (/kotlin)
I: Copying libs... (/META-INF/services)
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk into: base/dist/base.apk
leszeksiwik@fedora:~/howmanyfingershacking$
```

- Swoją drogą na ostatnim zrzucie pokazałem jak użyć samego narzędzia/biblioteki apktool bez pośrednictwa wrapera którego działanie sprowadza się do wywołania javy z opcją -jar żeby wykonać/uruchomić program spakowany jako archiwum javy
- Tak czy inaczej jeśli wszystko przebiegło ok, dostajemy komunikat, że przebudowana wersja aplikacji (czyli nowa wersja pliku base.apk) umieszczona została w katalogu base/dist
- No to zaglądnijmy tam:

```
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk into: base/dist/base.apk
leszeksiwik@fedora:~/howmanyfingershacking$ cd base/dist/
leszeksiwik@fedora:~/howmanyfingershacking/base/dist$ ls -al
razem 10960
drwxr-xr-x. 1 leszeksiwik leszeksiwik       16 05-23 14:46 .
drwxr-xr-x. 1 leszeksiwik leszeksiwik      208 05-23 14:46 ..
-rw-r--r--. 1 leszeksiwik leszeksiwik 11222403 05-23 14:46 base.apk
leszeksiwik@fedora:~/howmanyfingershacking/base/dist$
```

- No i faktycznie mamy tam nową paczkę. Więc nie pozostaje nic innego jak podmienić tą paczką poprawną wersję aplikacji, którą użytkownik ciągle ma na swoim urządzeniu
- Niestety musimy przed tym wykonać jeszcze dwa kroki, a mianowicie musimy:
    - dostosować nasze archiwum/naszą paczkę do formatu wymaganego przez Androida - zrobimy to korzystając z narzędzia zipalign
    - podpisać naszą paczkę zaplikacją (podpiszemy oczywiście “anonimowo” - bo w końcu jesteśmy niecnymi hackerami :)
- Zacznijmy od dopasowania się do formatu zgodnego z androidem, czyli:

```
leszeksiwik@fedora:~/howmanyfingershacking$ zipalign -v 4 base/dist/base.apk base/dist/base-aligned.apk
Verifying alignment of base/dist/base-aligned.apk (4)...
      44 resources.arsc (OK)
 1010917 AndroidManifest.xml (OK - compressed)
 1012646 META-INF/services/kotlinx.coroutines.CoroutineExceptionHandler (OK - compressed)
 1012809 META-INF/services/kotlinx.coroutines.internal.MainDispatcherFactory (OK - compressed)
 1012916 classes.dex (OK)
10607982 classes3.dex (OK - compressed)
10609511 classes2.dex (OK - compressed)
10724239 res/anim/abc_fade_in.xml (OK - compressed)
10724527 res/anim/abc_fade_out.xml (OK - compressed)
10724832 res/anim/abc_grow_fade_in_from_bottom.xml (OK - compressed)
10725280 res/anim/abc_popup_enter.xml (OK - compressed)
10725610 res/anim/abc_popup_exit.xml (OK - compressed)
10725959 res/anim/abc_shrink_fade_out_from_bottom.xml (OK - compressed)
10726411 res/anim/abc_slide_in_bottom.xml (OK - compressed)
10726714 res/anim/abc_slide_in_top.xml (OK - compressed)
```

- Po wywołaniu dostaniemy dość długą liste komunikatów, ale ważne abyśmy finalnie dostali informację, że wszystko poszło ok, czyli:

```
11112736 res/mipmap-xxhdpi/ic_launcher_round.webp (OK)
11118716 res/mipmap-xxxhdpi/ic_launcher.webp (OK)
11122632 res/mipmap-xxxhdpi/ic_launcher_round.webp (OK)
11130473 res/mipmap-anydpi/ic_launcher.xml (OK - compressed)
11130800 res/mipmap-anydpi/ic_launcher_round.xml (OK - compressed)
11131112 res/xml/backup_rules.xml (OK - compressed)
11131267 res/xml/data_extraction_rules.xml (OK - compressed)
11131453 kotlin/kotlin.kotlin_builtins (OK - compressed)
11136795 kotlin/annotation/annotation.kotlin_builtins (OK - compressed)
11137448 kotlin/collections/collections.kotlin_builtins (OK - compressed)
11139059 kotlin/coroutines/coroutines.kotlin_builtins (OK - compressed)
11139302 kotlin/internal/internal.kotlin_builtins (OK - compressed)
11139778 kotlin/ranges/ranges.kotlin_builtins (OK - compressed)
11141154 kotlin/reflect/reflect.kotlin_builtins (OK - compressed)
11142483 DebugProbesKt.bin (OK - compressed)
Verification succesful
leszeksiwik@fedora:~/howmanyfingershacking$
```

- Jeśli narzędzie zipalign się nie zgłasza, trzeba dodać katalog build-tools z AndroidSDK do systemowego PATH'a
- Pełna ścieżka do katalogu w którym znajduje się zipalign to w moim przypadku:

```
[siwik@d17-429-pc-teacher 34.0.0]$ pwd
/mnt/workspaces/shared/root_android-sdk/build-tools/34.0.0
```

- czyli dodanie go PATH'a to:

```
[siwik@d17-429-pc-teacher 34.0.0]$ export PATH=$PATH:/mnt/workspaces/shared/root_android-sdk/build-tools/34.0.0
```

- i po tym kroku zipalign powinien się już zgłaszać:

```
[siwik@d17-429-pc-teacher 34.0.0]$ zipalign
Zip alignment utility
Copyright (C) 2009 The Android Open Source Project

Usage: zipalign [-f] [-p] [-v] [-z] <align> infile.zip outfile.zip
       zipalign -c [-p] [-v] <align> infile.zip

  <align>: alignment in bytes, e.g. '4' provides 32-bit alignment
  -c: check alignment only (does not modify file)
  -f: overwrite existing outfile.zip
  -p: page-align uncompressed .so files
```

- Po tym kroku w katalogu z naszą zmodyfikowaną paczką powinien pojawić się plik base-aligned.apk (bo taką nazwę podałem przy wywołaniu narzędzia zipalign) który jest paczką zawierającą nasz zmodyfikowany / złośliwy kod, dostosowaną do formatu androidowego
- Musimy jeszcze tę paczkę podpisać, w tym celu użyjemy z kolei narzędzia apksigner.
- Jeśli przy próbie wywołania polecenia apksigner dostaniemy komunikat typu:

```
leszeksiwik@fedora:~$ apksigner
bash: apksigner: nie odnaleziono polecenia...
```

- To najprawdopodbniej (podobnie jak wcześniej dla polecenia/narzędzia adb) odpowiedni katalog nie został dodany do scieżki systemowej.
- apksigner zlokalizowany jest w katalogu w którym mamy nasze AndroidSDK i dalej w podkatalogu build-tools/numer wersji sdk której używamy (w moim przypadku 34.0.0) czyli finalnie w moim przypadku apksigner zlokalizowany jest w:

```
leszeksiwik@fedora:~/Android/Sdk/build-tools/34.0.0$ pwd
/home/leszeksiwik/Android/Sdk/build-tools/34.0.0
leszeksiwik@fedora:~/Android/Sdk/build-tools/34.0.0$ ls -al
razem 23104
drwxr-xr-x. 1 leszeksiwik leszeksiwik     586 05-14 12:53 .
drwxr-xr-x. 1 leszeksiwik leszeksiwik      12 05-14 12:53 ..
-rwxr-xr-x. 1 leszeksiwik leszeksiwik 1525352 05-14 12:53 aapt
-rwxr-xr-x. 1 leszeksiwik leszeksiwik 6076216 05-14 12:53 aapt2
-rw-r--r--. 1 leszeksiwik leszeksiwik     343 05-14 12:53 aarch64-linux-android-ld
-rwxr-xr-x. 1 leszeksiwik leszeksiwik 4746640 05-14 12:53 aidl
-rwxr-xr-x. 1 leszeksiwik leszeksiwik    2959 05-14 12:53 apksigner
-rw-r--r--. 1 leszeksiwik leszeksiwik     343 05-14 12:53 arm-linux-androideabi-ld
-rwxr-xr-x. 1 leszeksiwik leszeksiwik   38808 05-14 12:53 bcc_compat
-rw-r--r--. 1 leszeksiwik leszeksiwik   15149 05-14 12:53 core-lambda-stubs.jar
-rwxr-xr-x. 1 leszeksiwik leszeksiwik    2678 05-14 12:53 d8
-rwxr-xr-x. 1 leszeksiwik leszeksiwik 7297120 05-14 12:53 dexdump
-rw-r--r--. 1 leszeksiwik leszeksiwik     343 05-14 12:53 i686-linux-android-ld
drwxr-xr-x. 1 leszeksiwik leszeksiwik      38 05-14 12:53 lib
drwxr-xr-x. 1 leszeksiwik leszeksiwik     210 05-14 12:53 lib64
-rwxr-xr-x. 1 leszeksiwik leszeksiwik     647 05-14 12:53 lld
drwxr-xr-x. 1 leszeksiwik leszeksiwik       6 05-14 12:53 lld-bin
-rwxr-xr-x. 1 leszeksiwik leszeksiwik 1057936 05-14 12:53 llvm-rs-cc
-rw-r--r--. 1 leszeksiwik leszeksiwik     343 05-14 12:53 mipsel-linux-android-ld
-rw-r--r--. 1 leszeksiwik leszeksiwik 1069352 05-14 12:53 NOTICE.txt
```

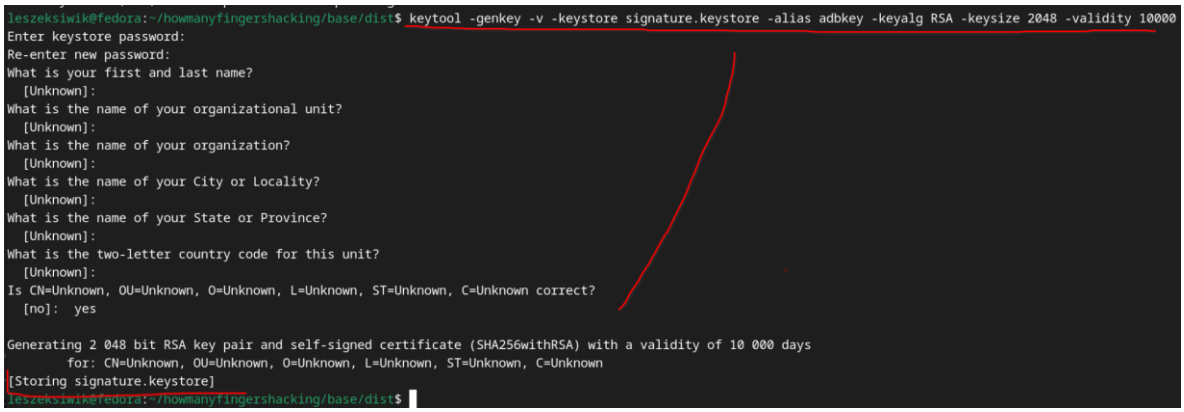
- więc jeśli dodamy sobie ścieżkę tego katalogu do systemowego PATH'a

```
leszeksiwik@fedora:~/Android/Sdk/build-tools/34.0.0$ export PATH=$PATH:/home/leszeksiwik/Android/Sdk/build-tools/34.0.0
```

- wówczas apksigner powinien już być widoczny / dostępny z dowolnej lokalizacji:

```
leszeksiwik@fedora:~/howmanyfingershacking/base/dist$
leszeksiwik@fedora:~/howmanyfingershacking/base/dist$ apksigner
USAGE: apksigner <command> [options]
       apksigner --version
       apksigner --help

EXAMPLE:
       apksigner sign --ks release.jks app.apk
       apksigner verify --verbose app.apk

apksigner is a tool for signing Android APK files and for checking whether
signatures of APK files will verify on Android devices.


        COMMANDS
rotate                Add a new signing certificate to the SigningCertificateLineage

sign                  Sign the provided APK
```

- No to (będąc w naszym "hackerskim" katalogu) spróbujmy użyć apksignera do podpisania naszej aplikacji:

- Zgodnie z informacjami podanymi tutaj: https://developer.android.com/tools/apksigner?hl=pl wywołanie apksignera to
    - apksigner sign –ks signature.keystore file.apk
- apksigner nam sie już zgłasza z dowolnej lokalizacji file.apk to będzie nasze base-aligned.apk, ostatnia zagadka co to jest signature.keystore
- Otóż jest to dedykowany plik do przechowywania kluczy (prywatnych) do szyfrowania / podpisywania etc. No i nie ma wyjścia musimy sobie taki wygenerować.
- Polecenie do tego:
- 

- I jeśli to się powiodło powinniśmy móc podpisać naszeą aplikację, no więc spróbujmy:

```
leszeksiwik@fedora:~/howmanyfingershacking$ apksigner sign --ks signature.keystore base/dist/base-aligned.apk
Keystore password for signer #1:
leszeksiwik@fedora:~/howmanyfingershacking$
```

- Jeśli wszystko poszło ok, obok podpisywanego pliku base-aligned.apk, powiniem pojawić się “odcisk” podpisu (plik .idsig), a sam podpisywany plik powininen był być modyfikowany w tym samym momencie kiedy tworzony był ten odcisk:

```
leszeksiwik@fedora:~/howmanyfingershacking$ apksigner sign --ks signature.keystore ba
Keystore password for signer #1:
leszeksiwik@fedora:~/howmanyfingershacking$ ls -al base/dist/
razem 22092
drwxr-xr-x. 1 leszeksiwik leszeksiwik       92 05-23 15:42 .
drwxr-xr-x. 1 leszeksiwik leszeksiwik      208 05-23 15:33 ..
-rw-r--r--. 1 leszeksiwik leszeksiwik 11298566 05-23 15:42 base-aligned.apk
-rw-r--r--. 1 leszeksiwik leszeksiwik    95772 05-23 15:42 base-aligned.apk.idsig
-rw-r--r--. 1 leszeksiwik leszeksiwik 11222403 05-23 15:33 base.apk
leszeksiwik@fedora:~/howmanyfingershacking$
```

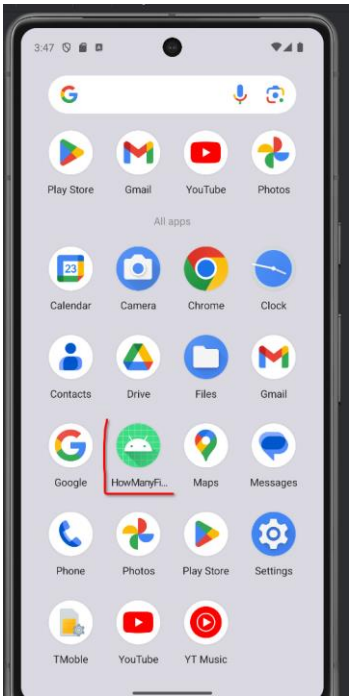
- No to finalnie, usuńmy z urządzenia starą (tę poprawnie działającą) wersję aplikacji, i zainstalujmy w to miejsce naszą “złośliwą” wersję. czyli:
- Odinstalowujemy:

```
leszeksiwik@fedora:~/howmanyfingershacking/base/dist$ adb uninstall com.example.howmanyfingers
Success
```

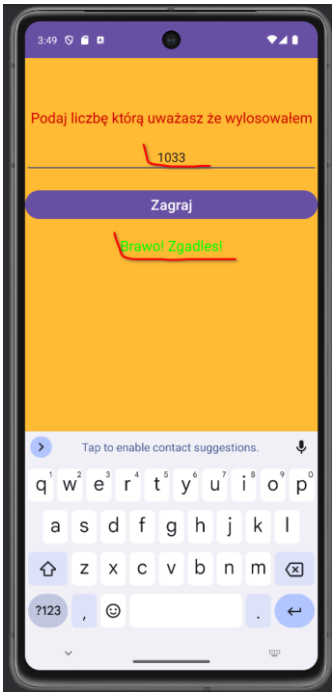
- I “podrzucamy” użytkownikowi naszą niecną wersję gry:

```
leszeksiwik@fedora:~/howmanyfingershacking$ ls -al base/dist/
razem 22092
drwxr-xr-x. 1 leszeksiwik leszeksiwik       92 05-23 15:42 .
drwxr-xr-x. 1 leszeksiwik leszeksiwik      208 05-23 15:33 ..
-rw-r--r--. 1 leszeksiwik leszeksiwik 11298566 05-23 15:42 base-aligned.apk
-rw-r--r--. 1 leszeksiwik leszeksiwik    95772 05-23 15:42 base-aligned.apk.idsig
-rw-r--r--. 1 leszeksiwik leszeksiwik 11222403 05-23 15:33 base.apk
leszeksiwik@fedora:~/howmanyfingershacking$ adb install -t base/dist/base-aligned.apk
Performing Incremental Install
Serving...
All files should be loaded. Notifying the device.
Success
Install command complete in 294 ms
leszeksiwik@fedora:~/howmanyfingershacking$
```

- I jeśl użytkownik uruchomi teraz swoją ulubioną grę:



- To powinno okazać się, że wygrywa niezaleznie od tego co poda jako swój typ.
- Na zrzucie poniżej przykład dla sytuacji w której podałem jako użytkownik wartość 1033 która na pewno nie zgadza się z tym co losujemy, bo aplikacja losuje swoje liczby z zakresu od 1..5 :)

- 

- Uff. Napracowalismy się, ale wygląda że osiągnelismy zamierzony efekt :)
- To, żeby nabrać wprawy:
    - Rozwińmy naszą aplikację dodając do niej licznik wygranych / punktację
    - Użytkownik powinien zyskiwać punkt za każdym razem kiedy uda mu się odgadnąć jaka liczba została wylosowana.
    - Przetestujmy / upewnijmy się że naliczanie punktacji działa poprawnie
    - A następnie “zhackujmy” aplikację w taki sposób, że po każdym odgadnięciu będzie się doliczało użytkownikowi nie jeden a 10 punktów :)
    - Może być w tym celu przyjrzenie się / zrozumienie jak działają / jak są zapisywane w kodzie .samli operacje inkrementowania / dodawnia
    - Objaśnienie większości operacj które możemy spotkać w plikach .smali można znaleźć np. tutaj: http://pallergabor.uw.hu/androidblog/dalvik_opcodes.html
    - Miłej zabawy :)