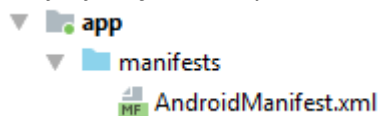


## TEMAT: Struktura projektu. Kontrolki i widoki. Interfejs programistyczny aplikacji.

### WPROWADZENIE

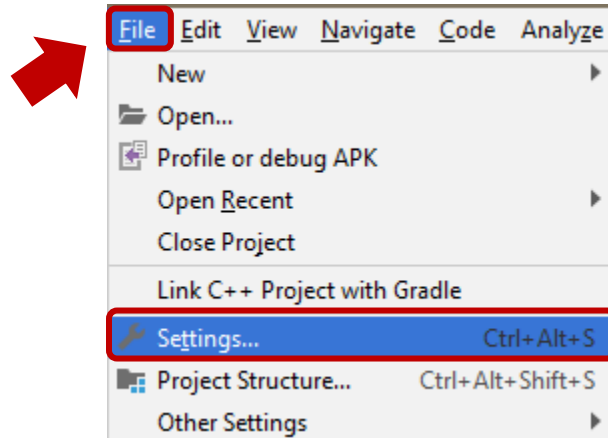
1. Bardzo istotne jest zrozumienie koncepcji działania aplikacji dedykowanych na platformę Android. Aplikacje takie składają się z wielu pojedynczych komponentów, z których każdy może potencjalnie stanowić punkt startowy lub samodzielną jednostkę, możliwą do uruchomienia przez inną aplikację (np. YouTube uruchamiany z poziomu aplikacji Messenger lub mapa z przeglądarki).
2. Podstawową taką jednostką, elementem budulcowym aplikacji jest **Activity**. Można je utożsamiać z pojedynczym ekranem widocznym w danym momencie na smartfonie. Programistycznie jest to dowolna klasa dziedzicząca po klasie Activity i nadpisująca jej metody oraz implementująca swoje własne.
3. Oprócz Activity istnieją jeszcze inne elementy budulcowe aplikacji, takie jak:
  - a. **Service**, czyli serwis służący do wykonywania czasochłonnych operacji w tle (nie posiada więc interfejsu graficznego),
  - b. **Broadcast Receiver**, czyli komponent odpowiedzialny za rozgłaszanie powiadomień dotyczących całego systemu, np. informacji o niskim poziomie baterii, czy zakończeniu pobierania plików lub wygaszeniu ekranu (działanie widoczne w postaci notyfikacji),
  - c. **Content Provider**, który odpowiada za współdzielone dane w aplikacji.
4. Trzy z wymienionych elementów budulcowych aplikacji (Activity, Service i Broadcast Receiver) są aktywowane poprzez asynchroniczne wywołanie zwane **Intent**. W ten sposób możemy stworzyć bardziej złożoną aplikację, zbudowaną z wielu powiązanych ze sobą komponentów. Należy pamiętać, że wywołania *Intent* mogą być kierowane także do elementów budulcowych innych aplikacji (w przypadku posiadania odpowiednich uprawnień). Wywołanie *Intent* można rozumieć jako żądanie wykonania pewnego określonego zadania kierowane do innego komponentu.
5. Konfiguracja całej aplikacji dokonywana jest za pośrednictwem pliku **AndroidManifest.xml**. To tu oznaczana jest klasa startowa aplikacji oraz określone są wymagane uprawnienia. W pliku tym znajduje się lista wszystkich komponentów budulcowych aplikacji.



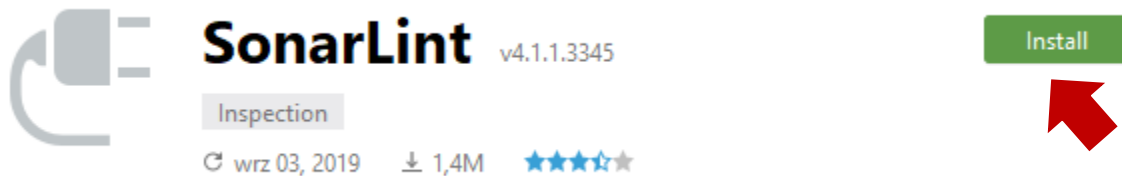
6. Do implementacji kodu używa się dedykowanego środowiska - **Android Studio**.
7. Ważne jest, aby dbać o aktualność wykorzystywanego oprogramowania (wersji Android Studio, obsługiwanego przez aplikację API itp.).

## Konfiguracja: przydatne informacje

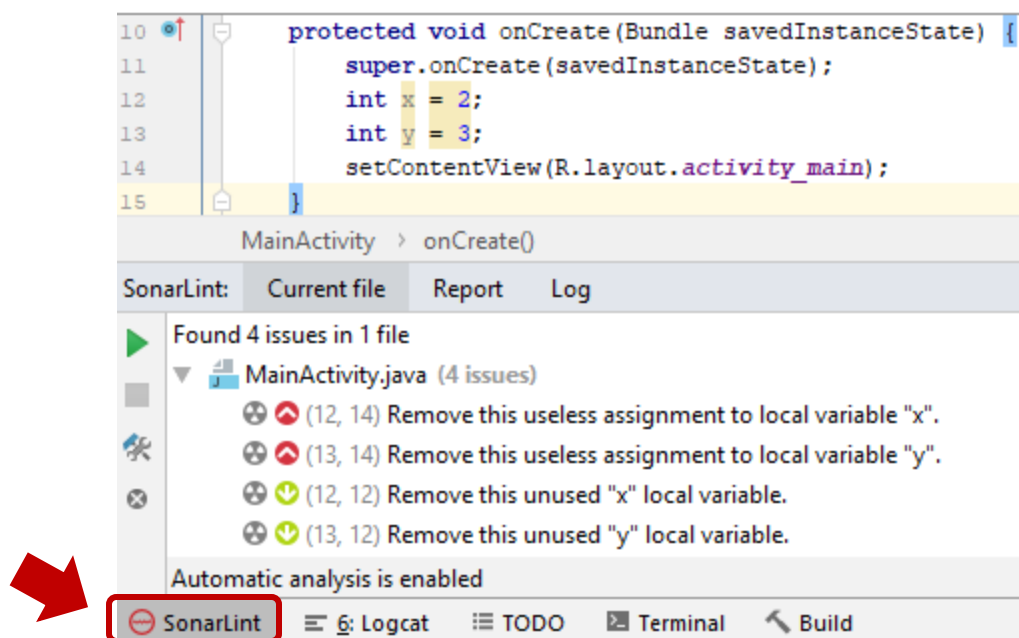
1. Pobieranie starszych wersji środowiska SDK
  - a. Komponenty starszych wersji Androida (np. do uruchomienia aplikacji na emulatorze ze starszą wersją systemu) są dostępne w *Tools/Android/SDK Manager*.
2. Instalujemy dodatkowe, przydatne pluginy:



- SonarLint – wykrywa błędy, które zazwyczaj są powodem zablokowania akceptacji kodu w trakcie code review. Zaznacza kod, który nie jest zgodny z dobrymi praktykami programistycznymi i powinien zostać poprawiony (więcej o dobrych praktykach w książce: „*Clean Code*” Robert C. Martin).
- Po instalacji pluginów zrestartuj Android Studio.



- Jak działa SonarLint? Przykład wykrytych błędów w kodzie:



3. Przykładowe skróty:

- 2 x Shift – wyszukiwanie dowolnego elementu kodu źródłowego.
- Ctrl + Shift + F – wyszukiwanie plików, ale też zawartości plików.
- Ctrl + Spacja – uzupełnianie kodu (wyświetlenie sugestii, której zawartość jest związana z tym, co aktualnie wpisujemy)
- Alt + Enter – podpowiedzi kontekstowe

Pozostałe: <https://developer.android.com/studio/intro/keyboard-shortcuts>

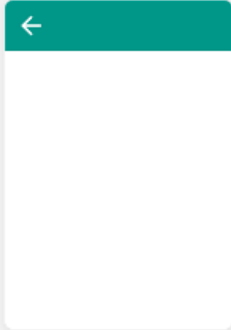
## TREŚĆ ZADANIA

- Cel:** pierwsze zadanie będzie służyło zapoznaniu się z podstawowymi założeniami budowania aplikacji na platformę Android. Obejmować będzie analizę struktury projektu, zapoznanie się ze środowiskiem Android Studio, implementację prostej aplikacji Quiz, tworzenie oraz powiązanie różnych komponentów, a także definiowanie własnych widoków, zawierających różne kontrolki.

**Uwaga:** poniższa instrukcja nie jest poradnikiem krok po kroku instruującym jak wykonać zadanie. Instrukcja zawiera luki, które należy uzupełnić samodzielnie na podstawie zdobytej wiedzy.

- W celu wykonania zadania utwórz nowy projekt w Android Studio zgodnie z instrukcją.

## Configure your project



Empty Activity

Creates a new empty activity

Name

Quiz

Package name

pl.edu.pb.wi

Save location

C:\Users\Kasia\AndroidStudioProjects\Quiz

Language

Java

Minimum API level

API 19: Android 4.4 (KitKat)

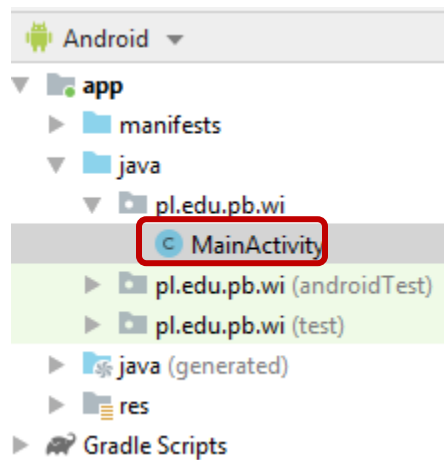
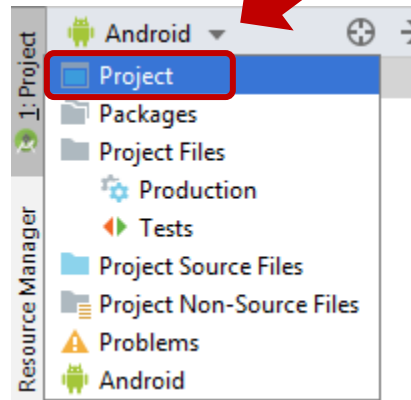
*i* Your app will run on approximately 95,3% of devices.  
[Help me choose](#)

☐ This project will support instant apps

☒ Use androidx.\* artifacts

- a. Wybierz:
- Typ aktywności: **empty activity**
  - Język: **Java**
  - Minimalna obsługiwana wersja SDK: **API 19: Android 4.4 (KitKat)** – sprawdź, jak zmienia się szacunkowa liczba urządzeń, które będą mogły obsługiwać Twoją aplikację w zależności od tego, którą wersję API wybierzesz. Poniżej wybranej wersji, system powinien odmówić zainstalowania aplikacji.
- Wskazówka: Wybierając minimalne obsługiwane API należy uwzględnić, jak wiele urządzeń będzie mogło korzystać z naszej aplikacji. Warto też mieć na uwadze, czy zadziała ona poprawnie na naszym smartfonie. Oczywiście nowsze wersje oferują więcej ciekawych funkcji.*
3. Zapoznaj się ze strukturą katalogów/plików w projekcie, z podstawowymi komponentami środowiska programistycznego, z oknami narzędziowymi:

a. Okno projektu



Po wybraniu opcji *Android* układ folderów i plików jest dostosowany do wygodnego tworzenia aplikacji – rzeczywista struktura katalogów jest ukryta. Jeśli jednak chcemy wyświetlić rzeczywistą strukturę katalogów i plików, możemy wybrać z listy rozwijanej widoków opcję *Project*.

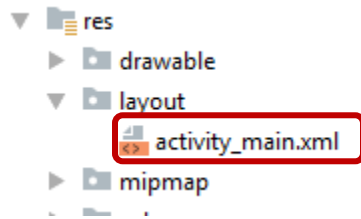
- b. Edytor – można w nim przeglądać i implementować kod, ale też wizualną stronę aplikacji na zasadzie drag & drop.
- c. SDK Manager – odpowiada za pobieranie starszych wersji API, aktualizacji narzędzi czy też nowych wersji systemu Android.



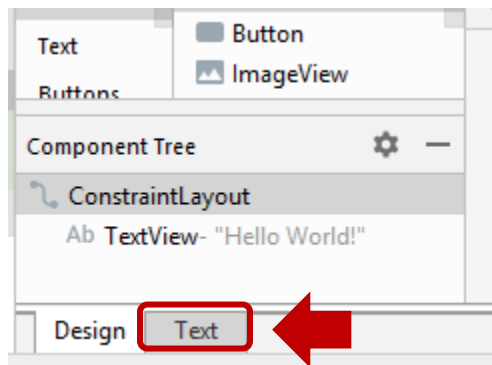
SDK Platforms				
SDK Tools				
SDK Update Sites				
Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.				
	Name	API Level	Revision	Status
<input checked="" type="checkbox"/>	Android 10.0 (Q)	29	3	Installed
<input type="checkbox"/>	Android 9.0 (Pie)	28	6	Not installed
<input type="checkbox"/>	Android 8.1 (Oreo)	27	3	Not installed
<input type="checkbox"/>	Android 8.0 (Oreo)	26	2	Not installed

4. W aplikacji domyślnie utworzony został interfejs użytkownika przypisany do głównej aktywności. Wprowadź w nim następujące zmiany:

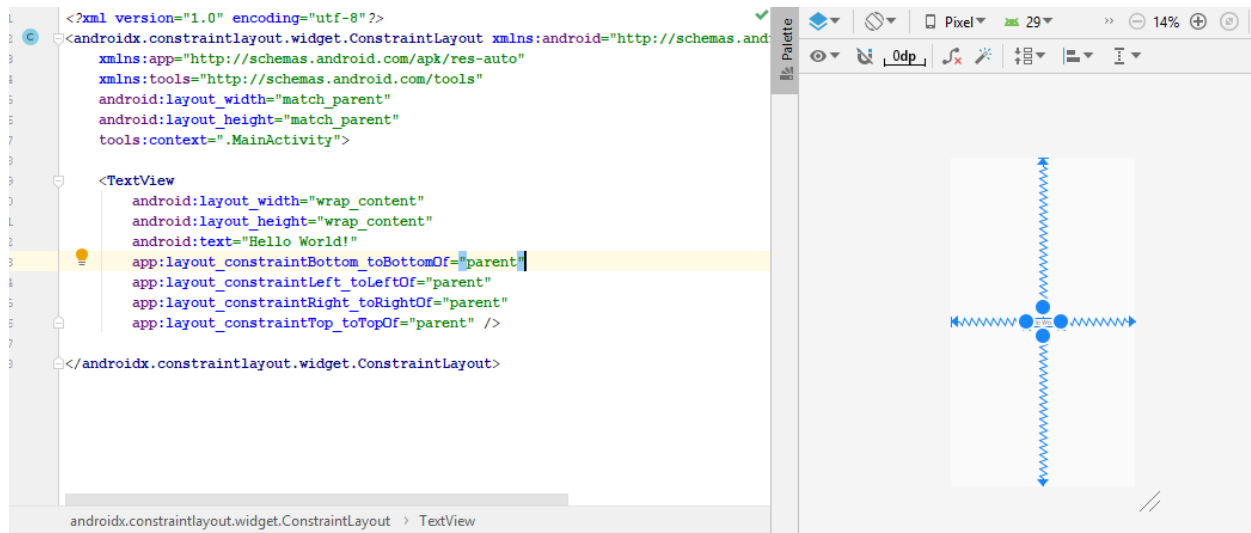
- a. Odnajdź i otwórz plik `activity_main.xml` w katalogu `app/res/layout`:



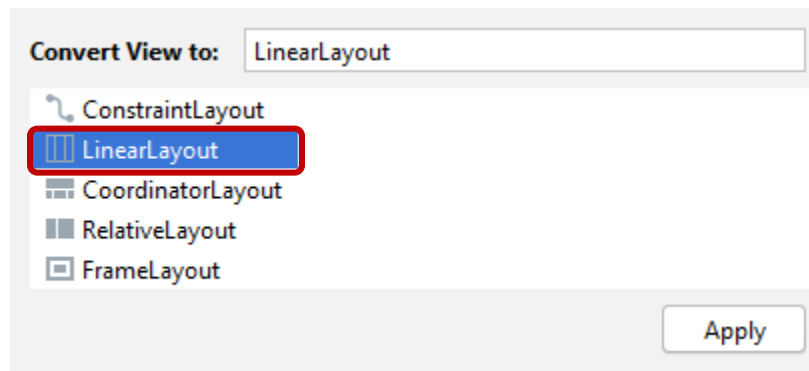
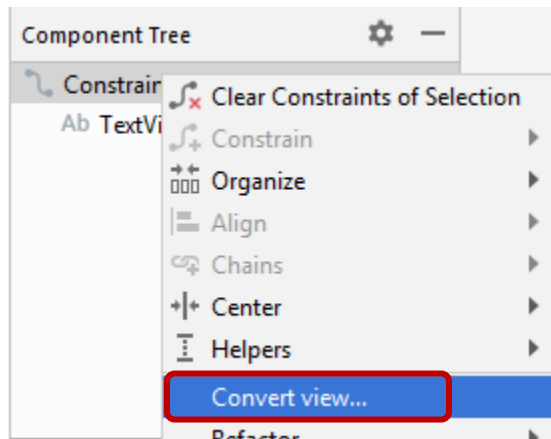
- b. Przejdź do zakładki *Text*. Jest to widok kodu, który definiuje elementy wyświetlające się w wybranym ekranie (aktywności) aplikacji. Wspomniane elementy składowe, z których budowany jest interfejs użytkownika, nazywane są też widgetami. Każdy z nich może być konfigurowany za pomocą odpowiednich atrybutów. Zapoznaj się z domyślnie wygenerowanym kodem. Spróbuj dokonać jakichś zmian w widoku w zakładce *Design* i sprawdź, jakie to ma skutki w kodzie xml w zakładce *Text*.



- c. Zmiany wprowadzane w kodzie xml domyślnie będą widoczne w oknie podglądu.



- d. W zakładce *Design* w drzewie komponentów zmień typ układu interfejsu na *LinearLayout*. Kliknij prawym przyciskiem myszy na nazwę aktualnego układu – *ConstraintLayout* i wybierz opcję *Convert view*.



Układ stanowi kontener dla innych elementów (widgetów). Może narzucać sposób ich rozmieszczenia na ekranie. Układ *LinearLayout* służy do rozmieszczenia widgetów w jednym wierszu lub kolumnie. Elementy wyświetlają się w takiej kolejności, w jakiej zostały zdefiniowane wewnątrz układu.

- e. Zmień orientację układu na pionową i ustaw wyśrodkowanie:

```
android:orientation="vertical"
android:gravity="center">
```

- f. Wewnątrz układu dodaj następujące elementy:

- *TextView*
- Poziomy układ *LinearLayout*
- Trzy przyciski *Button*

- g. Dwa przyciski powinny znajdować się wewnątrz układu poziomego (dzięki temu będą wyświetlane obok siebie).

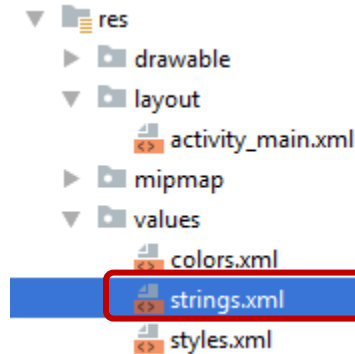
- h. Predefiniowane wartości atrybutów *layout\_width* i *layout\_height*:

- *wrap\_content* – widok będzie miał rozmiary dopasowane do rozmiarów jego zawartości,

- *match\_parent* – widok będzie miał rozmiary obiektu nadrzędnego.

We wszystkich dodanych elementach ustaw szerokość i wysokość na wartość *wrap\_content*. Dodatkowo w widoku *TextView* określ odstęp (*padding*) na 30dp (*dp* to jednostka miary niezależna od gęstości pikseli na ekranie urządzenia).

- Dodaj napisy do poszczególnych elementów. Najpierw utwórz stałe, w których zdefiniujesz napisy. W tym celu otwórz plik *strings.xml*:



- Nie usuwaj istniejącej zawartości tego pliku, tylko dodaj następujące definicje:

```
<string name="button_true">Prawda</string>
<string name="button_false">Fałsz</string>
<string name="button_next">Następne</string>
```

- Odwołaj się do stworzonych napisów w pliku układu *activity\_main.xml*:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_true"/>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_false"/>
<Button
    android:id="@+id/next_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_next"/>
```

- Dodaj identyfikatory poszczególnym elementom widoku:

```
<TextView
    android:id="@+id/question_text_view"

<Button
    android:id="@+id/true_button"

<Button
    android:id="@+id/false_button"
```



```
<Button
    android:id="@+id/next_button"
```

Znak plusa przy wartościach atrybutu *android:id* związany jest z faktem, że identyfikatory są tworzone. W przypadku innych atrybutów, np. *android:text*, jedynie odwołujemy się do istniejących elementów (w podanym przykładzie do zdefiniowanych napisów).

5. Utwórz pomocniczą klasę *Question*, która będzie zawiera pojedyncze pytanie quizu. Klasa powinna zawierać następujące pola:

```
public class Question {
    private int questionId;
    private boolean trueAnswer;

    public Question(int questionId, boolean trueAnswer) {
        this.questionId = questionId;
        this.trueAnswer = trueAnswer;
    }
}
```

6. Otwórz klasę *MainActivity* – kolejne zmiany będą dotyczyły podłączenia komponentów interfejsu użytkownika do aktywności, stworzenia obiektów nasłuchujących oraz zastosowania wzorca MVC (model-view-controller).
  - a. Zwróć uwagę na następującą linijkę kodu, która od początku znajduje się w pliku aktywności:

```
setContentView(R.layout.activity_main);
```

Metoda *setContentView* dołącza widok (układ interfejsu użytkownika zdefiniowany w pliku *activity\_main.xml*) do aktywności. Poszczególne elementy układu są tworzone (rozwijane) zgodnie z definicjami.

- b. W kodzie aktywności możemy odwoływać się do zdefiniowanych elementów widoku poprzez ich identyfikatory. Stosujemy do tego metodę *findViewById*. Dodaj odwołania do przycisków oraz napisu:

```

private Button trueButton;
private Button falseButton;
private Button nextButton;
private TextView questionTextView;

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

    trueButton = findViewById(R.id.true_button);
    falseButton = findViewById(R.id.false_button);
    nextButton = findViewById(R.id.next_button);
    questionTextView = findViewById(R.id.question_text_view);
}

```

- c. Odnajdź i wyświetl plik R, w którym zapisywane są wszystkie zasoby aplikacji. Wyszukaj w nim utworzone komponenty (zasoby) – przyciski, pole tekstowe, napisy.
- d. Stwórz listę pytań (minimum 5) oraz zmienną przechowującą indeks bieżącego pytania. Tematyka quizu może być dowolna (poniżej podane są jedynie przykładowe pytania).

Dodaj definicje pytań w pliku *strings.xml*:

```

<string name="q_activity">Activity jest jednym z komponentów aplikacji Android</string>
<string name="q_version">Android 8 jest najnowszą wersją systemu Android</string>
<string name="q_listener">Aplikacje dla systemu Android są sterowane zdarzeniowo</string>
<string name="q_resources">Zasoby aplikacji dla systemu Android definiuje się w plikach .xml</string>
<string name="q_find_resources">Do zasobów w aplikacji dla systemu Android odnosimy się po nazwach</string>

```

Utwórz listę pytań w klasie *MainActivity*:

```

private Question[] questions = new Question[] {
    new Question(R.string.q_activity, answer: true),
    new Question(R.string.q_find_resources, answer: false),
    new Question(R.string.q_listener, answer: true),
    new Question(R.string.q_resources, answer: true),
    new Question(R.string.q_version, answer: false)
};

```

Utwórz zmienną – indeks bieżącego pytania:

```

private int currentIndex = 0;

```

- e. Dodaj metodę sprawdzającą, czy odpowiedź na pytanie jest prawidłowa. Metoda w ostatniej linii zawiera utworzenie komunikatu typu *toast*. W komunikacie podawana będzie informacja o poprawności odpowiedzi użytkownika.

```
private void checkAnswerCorrectness(boolean userAnswer) {
    boolean correctAnswer = questions[currentIndex].isTrueAnswer();
    int resultMessageId = 0;
    if (userAnswer == correctAnswer) {
        resultMessageId = R.string.correct_answer;
    } else {
        resultMessageId = R.string.incorrect_answer;
    }
    Toast.makeText(context: this, resultMessageId, Toast.LENGTH_SHORT).show();
}
```

- f. Dodaj w metodzie *onCreate* obiekty nasłuchujące poprzez zastosowanie anonimowych klas wewnętrznych. Obiekty nasłuchujące w Androidzie implementują predefiniowane interfejsy, obsługujące różne zdarzenia, na które aplikacja może zareagować. W przypadku aplikacji z tego zadania będziemy nasłuchiwać zdarzenia kliknięcia przycisku przez użytkownika. Istnieje odpowiadający za to interfejs: *View.OnClickListener*.

```
trueButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        checkAnswerCorrectness( userAnswer: true);
    }
});
```

Analogiczną metodę utwórz dla przycisku *falseButton*.

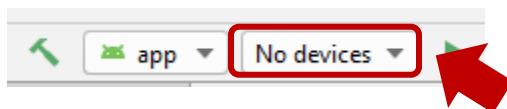
- g. Dodaj metodę ustawiającą w polu tekstowym aktualne pytanie:

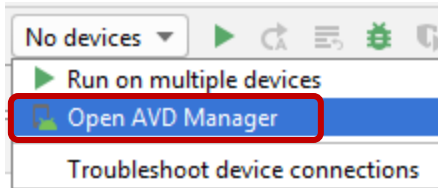
```
private void setNextQuestion() {
    questionTextView.setText(questions[currentIndex].getQuestionId());
}
```

- h. Dodaj wywołania powyższej metody: domyślne ustawienie na pierwsze pytanie i zmianę po kliknięciu przycisku *Następne*:

```
nextButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        currentIndex = (currentIndex + 1)%questions.length;
        setNextQuestion();
    }
});
setNextQuestion();
```

7. Utwórz emulator (wirtualne urządzenie), aby uruchomić aplikację:





Kliknij *Create virtual device* i wybierz predefiniowane urządzenie lub stwórz własne.

Pobierz wybraną wersję systemu Android, która będzie na emulatorze:

Select a system image			
Recommended x86 Images Other Images			
Release Name	API Level	ABI	Target
<a href="#">Q Download</a>	29	x86	Android 10.0 (Google Play)
<b>Pie</b>	28	x86	Android 9.0 (Google Play)
<a href="#">Oreo Download</a>	27	x86	Android 8.1 (Google Play)
<a href="#">Oreo Download</a>	26	x86	Android 8.0 (Google Play)
<a href="#">Nougat Download</a>	25	x86	Android 7.1.1 (Google Play)
<a href="#">Nougat Download</a>	24	x86	Android 7.0 (Google Play)

- Jeśli posiadasz telefon z systemem Android, uruchom na nim swoją aplikację (za pierwszym razem konieczne będzie włączenie opcji programistycznych w ustawieniach smartfona).
- Dodaj do aplikacji zliczanie poprawnych odpowiedzi – wyświetl rezultat po udzieleniu przez użytkownika odpowiedzi na ostatnie pytanie.
- Ostatecznie aplikacja powinna wyglądać w następujący sposób:

