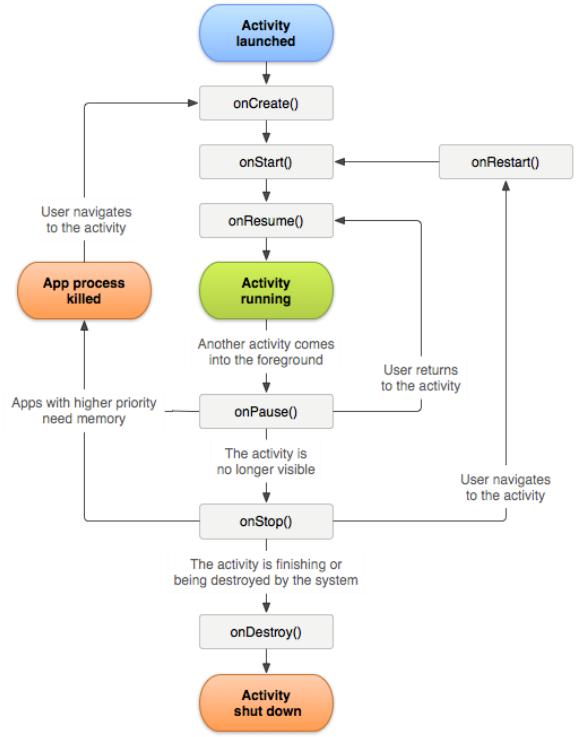
Cykl życia Activity



1. Uruchom aplikację Android Studio.
2. Utwórz nowy projekt:
   2.1. SKD: 22 (android 5.1 Lollipop)
   2.2. Typ: Empty Activity
3. Plik MainActivity.java (położony w folderze: java/{package}/ zawiera kod tworzonego programu (w modelu MVC odpowiada za C-Kontroler)
4. W pliku znajduje się domyślnie utworzona metoda onCreate uruchamiana przy inicjalizacji aplikacji.
5. Dodaj na końcu danej metody jedną (lub obie) z poniższych komend:
   5.1.
   ```
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);

       Log.d("Lab2", "Metoda OnCreate");
   }
   ```
   Funkcja Log.d() tworzy logi aplikacji w oknie debug AndroidStudio. Przyjmuje 2 parametry:
   - TAG – pozwalający na identyfikację logu spośród innych (dowolny ciąg znaków)
   - Wiadomość

   5.2.
   ```
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);

       Toast.makeText(this, "Metoda onCreate", Toast.LENGTH_SHORT).show();
   }
   ```
   Metoda Toast pozwala wyświetlać krótkie wiadomości tekstowe na ekranie (systemu Android) w postaci półprzezroczystego boxu z tekstem. W danym przypadku Tworzymy (niejawnie) obiekt klasy Toast uruchamiając na nim metodę makeText przyjmującą 3 parametry:
   - Aktualną activity (Context)
   - Tekst do wyświetlenia
   - Długość wyświetlania komunikatu (Toast. LENGTH_SHORT, lub Toast. LENGTH_LONG)

   Następnie należy na utworzonym obiekcie uruchomić metodę show() by wyświetlić komunikat.
6. Dodaj brakujące biblioteki (IDE podkreśla na czerwono metody i klasy których nie rozpoznaje)
   6.1. Zaznacz podkreślenie i naciśnij ALT+enter
   6.2. Wybierz z menu „import class”
7. Utwórz (przeciąż) metodę onResume w tej samej klasie MainActivity i dodaj tam wyświetlenie odpowiedniego komunikatu (tak samo, jak w metodzie onCreate)

```
@Override
protected void onResume(){
    super.onResume();

    Log.d("lab2", "metoda onResume");
    //lub
    Toast.makeText(this, "Metoda onResume", Toast.LENGTH_SHORT).show();
}
```

Przeciążając istniejącą metodę musimy użyć słowa kluczowego @Override, następnie podajemy modyfikatory i nazwę metody (np. onResume). Metody te nie przyjmują argumentów.
W ciele metody musimy odwołać się do metody rodzicielskiej poprzez odwołanie super.nazwaMetody()

Zadanie 1 – cykl życia

1. Utwórz metody, które wyświetlą komunikat (w wybranej wcześniej formie) w momencie gdy:
   - Activity zostanie przykryte
   - Activity zostanie wstrzymane
   - Activity zostanie zamknięte
   - Activity zostanie przywrócone po wstrzymaniu
2. Uruchomić program (wcześniej oczywiście należy odpalić emulator Androida w wersji co najmniej 5.1)
3. Należy sprawić, by zostały wyświetlone wszystkie sześć utworzonych komunikatów (poprzez oddziaływanie użytkownika na emulator)

Metody tworzymy tak jak utworzona została metoda onResume

Nazwy potrzebnych metod można znaleźć w diagramie na początku instrukcji

Należy zrozumieć kiedy, jakie metody będą wywoływane

LogCat/Konsola AndroidStudio

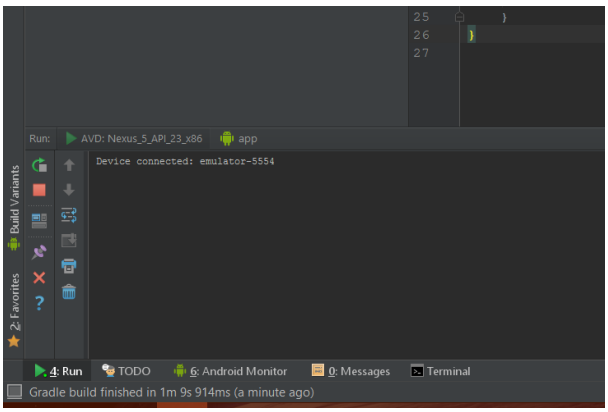
Konsola w Android Studio została podzielona na kilka zakładek.

W zakładce **Run** – znajdziemy informację o aktualnie przebiegającej komunikacji pomiędzy AndroidStudio a emulatorem (np. uruchomienie, podłączenie, wgrywanie, instalowanie, uruchamianie).

Warto monitorować daną zakładkę, gdyż często tylko tam można znaleźć informację, że AndroidStudio pracuje

W zakładce **Android Monitor** znajdujemy zapis logów (w czym te pochodzące z funkcji Log.d()). Zakładka ta pozwala także na filtrowanie logów po ich tagach i typie (w wyszukiwarkę tej zakładki wystarczy wpisać pierwsze litery Tagów komunikatu „Lab2”, a pozostałe rekordy pozostaną ukryte).

Pasek statusu (na samym dole) również wskazuje, że AndroidStudio wykonuje jakieś operacje i należy uzbroić się w cierpliwość (niestety nie zawsze pokazuje wszystkie działania).



Zasoby

Zasoby pozwalają na rozdzielenie warstwy prezentacji(wyglądu), od warstwy aplikacji (kodu), odpowiadają więc za V –view w modelu MVC.

Zasoby opisujemy plikami XML, są umieszczone w odpowiednich folderach odpowiadających ich typowi w głównym folderze „res”

1. Przejdź do pliku Activity_main.xml

W zadaniu Lab 1-Hello World tworzone były

(res/layout) – automatycznie uruchomi się widok designera
2. Przełącz się do widoku XML danego zasobu (zakładka Text na dolnym pasku designera)
3. Odnajdź kontrolkę TextView. Właściwość android:text zawiera bezpośredni ciąg znaków do wyświetlenia
4. Otwórz zasoby typu String (res/values/strings.xml)
5. Dodaj zasób tekstowy o nazwie „moj_tekst" i wartości „Witaj świecie!"
6. Wróć do zasobu Layoutu i modyfikuj tekst w kontrolce TextView, aby wyświetlić utworzony w pkt.5 zasób.
   - Aby odwołać się do zasobów należy najpierw podać jego typ poprzedzony znakiem @, a następnie jego nazwę
7. Uruchom Translation Editor na pliku zasobów typu String
8. Kliknij ikonkę Globu w lewym górnym rogu Edytora i wybierz język (np. Polski)
9. Automatycznie zostanie utworzona wybrana wersja językowa aplikacji, za pomocą edytora można zmieniać zasoby typu tekstowego w różnych wersjach lokalizacyjnych
   - Wersje językowe można poznać, gdyż pojawiają się dodatkowe pliki strings.xml z oznaczeniem którego języka dotyczą.
   - System automatycznie uruchamia tą wersję językową, jaka jest ustawiona w systemie, w przypadku gdy aplikacja nie posiada takiej wersji, uruchamiana jest wersja domyślna

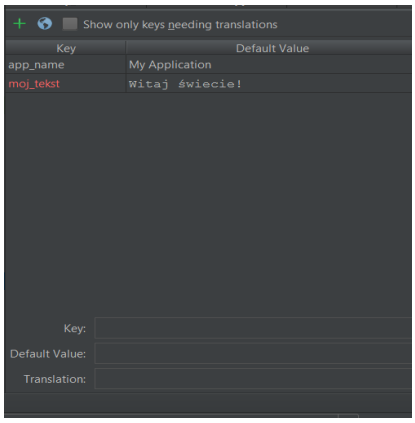
zasoby widoku (layout) w zależności od orientacji ekranu. Zasób ten opisywał kontrolki widoczne na ekranie, ich położenie i właściwości.

Zasoby typu String (res/values/strings.xml) zawierają zmienne typu tekstowego.

```
<resources>
    <string name="app_name">My Application</string>
    <string name="moj_tekst">Witaj świecie!</string>
</resources>
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/moj_tekst" />
```

Android Studio udostępnia edytor do bardziej intuicyjnego dodawania i zmian zasobów typu tekstowego (opcja „Open editor" w prawym górnym rogu pliku XML)



Zadanie 2 – Lokalizacja oprogramowania

1. Wstaw co najmniej 5 nowych kontrolek typu TextView w zasobie Layoutu
2. Utwórz co najmniej 5 nowych zasobów typu String
3. Przypisz zasoby typu String do atrybutu text utworzonych kontrolek
4. Przetłumacz aplikację na co najmniej 3 języki (np. googleTranslate)

Język systemu Android można zmienić w:
ustawienia (settings)->
osobiste (personal)->
język i wprowadzanie tekstu (language&input)->
język (language)

Obrazy w tle

1. Odnajdź folder „drawable" w explorerze Windows

Kliknięcie prawym przyciskiem myszy folder zasobów rozwija menu podręczne, gdzie jedną z opcji jest „Show in Explorer"

2. Wgraj (lub utwórz) do tego folderu jakiś obrazek (bez spacji w nazwie)
3. Android studio Automatycznie powinien zaktualizować zawartość folderu drawable i wyświetlić nowy zasób w postaci dodanego obrazka
4. Przejdź do zasobu Layout (activity_main.xml)
5. W dowolnie wybranej kontrolce (np. RelativeLayout) dodaj parametr „android:background” i nadaj mu wartość zasobu „@drawable/nazwaObrazka” (bez rozszerzenia)

Do wyświetlania obrazów w aplikacji można użyć też kontrolki ImageView. W tym przypadku należy ustawić parametr android:src celem załadowania określonego obrazu.

Zadanie 3 – kolorowe tła

1. Utwórz kilka zasobów typu Color (res/values/colors.xml)
2. Dodaj utworzone zasoby Color do parametru background kontrolek w Twojej aplikacji (tak samo jak w przypadku obrazu, jednak nalży pamiętać że zmienia się typ zasobu)

Zasób color jest zmienną zawierającą hexadecymalną wartość określonego koloru. Wartość koloru: #RRGGBB gdzie R odpowiada za składową czerwoną <00, FF>, G składową zieloną <00,FF>, a B składową niebieską <00,FF>

Zasoby dimension

W folderze /res/values/dimens występują pliki xml, w których dla różnych wariantów layoutu (wielkość, rozdzielczość, orientacja, tryb, wersji i inne) można ustawić zmienne „wielkości” np. marginesy, wielkość czcionek, czy innych elementów.
Tak jak w przypadku innych zasobów należy je przypisywać do odpowiednich atrybutów poprzez @{typ}/{nazwa}

Wartości wymiarów można podawać w następujących jednostkach:

- *px – pixele*
- *in – cale*
- *mm – milimetry*
- *pt – punkty (1/72cala)*
- *dp (Density-independent pixel)– skala relatywna, gdzie* `px = dp * (dpi / 160)`