Android Studio

Położenie: (nie dotyczy)

Informacje

Android Studio to nowsze (w stosunku do słabo rozwijanego *Eclipse*) środowisko programistyczne firmy Google wydane na licencji *„Apache 2.0”* (bezpłatne z możliwością komercyjnego zastosowania, ale daje firmie *Google* możliwość zmian licencji w każdym momencie bez uzasadnienia). Istnieje zarówno wersja na *Linux*, jak i *Windows*. Niestety, *Android Studio* wymaga 32-bitowych bibliotek i narzędzi. W jądrze Linuksa musi więc być aktywna opcja:
Executable file formats / Emulations --->
[*] IA32 Emulation
Android Studio wymaga także środowiska JDK. Może to być:

- **oracle-jdk-bin** (dobrze działa z wersją 1.8, w wersji 11 są problemy z ustawieniem w *eselect*);
- **icedtea** (zastępuje JDK 1.8);
- **openjdk** (zastępuje JDK 11);

Jeśli *Android Studio* nie wykryje ścieżki do JDK, można ręcznie ją ustawić w następującej lokalizacji: *File / Project Structure / SDK Location / JDK Location*: */opt/android-studio/jre* (lub */usr/lib/jvm/oracle-jdk-bin-1.8*).

Po instalacji

- Zainstalować lub uaktualnić dodatkowe pakiety → *File / Settings / Apperance & Behavior / System Settings / Android SDK / SDK Platforms...* lub *Tools / SDK Manager*;
- Utworzyć virtualny telefon w emulatorze → *Tools / AVD Manager* *(Uwaga: Jeśli projekt nie jest poprawnie otwarty, pozycja nie pojawi się w sekcji Tools)*
- Uaktywnić opcję automatycznych aktualizacji edytora → *File / Settings / Editor / General / Auto Import / Add unambiguous imports on the fly*

Struktura aplikacji

- *nazwaProjektu/src/main/res/layout/**activity_main.xml*** (lub ***content_main.xml***) – definicje wyglądu aplikacji; plik taki posiada na dole dwie zakładki: *Design* (tryb WYSIWYG) oraz *Text* (tryb tekstowy);
- *nazwaProjektu/src/main/java/com/example/nazwaProjektu/MainActivity/**MainActivity.java*** – co aplikacja ma robić;
- *C:\Users\robert\AndroidStudioProjects\ZamianaTekstu3\app\build\outputs\apk\apk-debug.apk* – zbudowana aplikacja;
- aplikacja może składać się z wielu plików *activity* (wielu screenów); nowe activity tworzymy poprzez wciśnięcie prawego przycisku myszy na *nazwaAplikacji/app/New/Activity*.

Emulator

Jeśli emulator nie uruchamia się, należy zmniejszyć mu ilość pamięci RAM do 512MB lub przełączyć w tryb „Store a *snapshot for faster startup*”. Należy także sprawdzić, czy nie uruchomił się poza ekranem (sprawdzić czy jest uruchomiony na pasku zadań).

Do działania emulatora w Linux, wymaga także modułu KVM (maszyna wirtualna, która koliduje z *VirtualBox*):
[*] Virtualization --->
<M> Kernel-based Virtual Machine (KVM) support
<M> KVM for Intel processors support
<M> Host kernel accelerator for virtio net
Uwaga: Wcześniej należy sprawdzić, czy procesor wspiera wirtualizację (wsparcie musi być włączone w BIOS):
 # lscpu
oraz
 # egrep -c '(vmx|svm)' /proc/cpuinfo
Jeśli wynik będzie wynosił „0” - nie ma wsparcia wirtualizacji. Jeśli powyżej „0” - jest wsparcie.

Aby maszyna działała, należy na ten czas wyładować moduły *VirtualBox* (jeśli takie są załadowane), choć nie zawsze jest to konieczne:
modprobe -r vboxnetadp vboxnetflt
Należy także sprawdzić, czy zwykły użytkownik ma dostęp do urządzenia */dev/kvm*:
ls -al /dev/kvm
Moduły odpowiedzialne za działanie *kvm* to:
modprobe kvm
modprobe kvm_intel
Jeśli zwykły użytkownik nie ma dostępu do urządzenia */dev/kvm*, należy ustawić w */etc/udev/rules.d/80_kvm.rules*:
KERNEL=="kvm", GROUP="kvm", MODE="0666" (dostępne dla wszystkich użytkowników)
KERNEL=="kvm", GROUP="kvm", MODE="0660" (dostępne tylko dla będących w grupie *kvm*)
Uwaga: Zagadką jest, że emulator został uruchomiony po skompilowaniu jądra bez modułu KVM (brak urządzenia */dev/kvm*). Prawdopodobnie użyty jest pakiet *qemu* (bez akceleracji, więc działa bardzo powoli).

Podłączanie telefonu
Aby włączyć na telefonie tryb programisty (*developer options*) należy kliknąć 7 razy w:
Ustawienia / Informacje o urządzeniu / Numer wersji (kompilacji)
Pojawi się wtedy następująca sekcja, którą należy zaznaczyć:
Ustawienia / Opcje programisty / Debugowanie USB
W *Android Studio* należy najpierw włączyć opcję *Tools/Android/Enable ADB Integration* (lub wyłączyć i włączyć ponownie), uruchomić *Run/Debug* '*app*', a potem *Run/Run* '*app*'.
W telefonie należy wybrać połączenie USB typu PTP (anulować typ MTP jeśli zostanie zaproponowany).

Słownik
ADB - *Android Debug Bridge*, pozwala zarządzać emulatorem.

Google Play
Koszt umieszczenia swojej aplikacji w sklepie *Play* wynosi 25 dolarów (oraz 30% ceny sprzedawanej aplikacji). Do tego należy doliczyć koszty utrzymania konta obsługującego płatności.
Konta *Google Play* zakłada się na stronie: *https://play.google.com/apps/publish/signup/*
Następnie należy podać numer karty kredytowej, nazwę banku, kod bezpieczeństwa. Po założeniu konta, przesyłamy plik *.apk i określamy, na jakie urządzenia i wersje Androida jest przeznaczony. Musimy także ustalić sprawy podatkowe.

Programowanie
Struktura kodu
Nadrzędnym składnikiem kodu jest **klasa**, wewnątrz której znajdują się **metody** (odpowiednik funkcji z innych języków programowania). Jeśli metoda nie ma zwracać wyniku, jest typu ***void***. Jeśli ma zwracać jakiś wynik, musimy w miejscu słowa „*void*” wpisać typ zwracanej zmiennej i zastosować na koniec kodu ***return***. Istotę metody można ująć w zasadzie „*Code reuse*”, która brzmi: „*Write once, run it many times*”.
Klasy i metody mają modyfikatory:
public - wszystkie klasy mają dostęp do pól danych i metod *public*;
private - dostęp do metod i pól danych posiadają jedynie inne metody tej samej klasy;
protected - używana jedynie przez metody swojej klasy oraz metody wszystkich jej klas pochodnych;
static – nierozrywalnie połączona z klasą, w której znajduje się; wywołanie tak zdefiniowanej metody musi być poprzedzone zawsze nazwą klasy, w której znajduje się, np. *nazwaKlasy.nazwaMetody();*
@override – zastąpienie (nadpisanie) poprzedniego działania (poprzedniej *Activity*) obecnym. Także nadpisanie domyślnych (wbudowanych w Androida) metod i zachowań.
Przykład deklaracji metody:

dostęp	*deklaracja*	*typ wyniku*		*typ*	*parametr*	
public	**static**	**int**	**nazwaMetody**	**(int**	**number)**	**{…}**

Metoda z parametrem „*View*” oznacza, że będzie uruchomiona po kliknięciu w widoczny element. Aby ją wywołać (ang. „*method call*”) w innym miejscu kodu, należy wprowadzić:
nazwaMetody(null);

Wartość null

Wartość *null* może występować tylko w polach typu *string* (nigdy nie może być typu *integer*). Wartości *null* nigdy nie parsujemy, w szczególności nie będzie działać „*Integer.parseInt()*”. Aby sprawdzić, czy pole ma wartość null powinniśmy stosować ==, a nie „*equals*”. Metody sprawdzania null:

```
int wartoscPola = 0;
if (!TextUtils.isEmpty(nazwaSprawdzanegoPola.getText())) {
   wartoscPola = Integer.parseInt(nazwaSprawdzanegoPola.getText());
}
```

Inny sposób:

```
public boolean czyPoleJestPuste (EditText editText) {
   boolean result = editText.getText().toString().length() <= 0;
   if (result)
      Toast.makeText(context, "Wprowadzać jakąś liczbę!", Toast.LENGTH_SHORT).show();
   return result;
}
```

Jeszcze inny sposób:

```
public static void nazwaMetody() {
    try{
        String nazwaPola=null;
        System.out.println (str.length());
    }catch(NullPointerException e){
        System.out.println("Wystapil wyjatek");
    }
}
```

Zamiana Integer na String

```
int liczbaJakoInteger = -10;
String liczbaJakoString = String.valueOf(liczbaJakoInteger);
// lub
int liczbaJakoInteger = -10;
String liczbaJakoString = Integer.toString(liczbaJakoInteger);
```

Wyświetlanie liczb w polu tekstowym

```
mojTekst.setText("Oto liczba: " + String.valueOf(liczbaInteger)); // lub
mojTekst.setText("Oto liczba: " + Integer.toString(liczbaInteger));
```

Zamiana String na Integer

```
String liczbaJakoString = "1234";
int liczbaJakoInteger = Integer.parseInt(liczbaJakoString);
// lub
String liczbaJakoString = "1234";
int liczba = Integer.valueOf(liczbaJakoString);
```

Zmiana koloru tekstu

```
mojTekst.setTextColor(Color.parseColor("#ffff00"));
```

SharedPreferences

Prosty rodzaj bazy danych w formacie XML (klucz = wartość).

```
// Sprawdzanie, czy dana wartość istnieje
```

Nie trzeba sprawdzać. Korzystając z *getString*, podajemy drugi **zastępczy** parametr (jakąś wartość), na wypadek, jeśli klucza w ogóle by nie było.Zwracana jest wtedy właśnie ta wartość.
Można także wykorzystać:
contains(String key);

```
// ZAPISYWANIE KLUCZY w SharedPreferences
public void save(Context context, String text) {
   SharedPreferences ustawienia;
```

```
    Editor edytor;
    ustawienia = context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
    edytor = ustawienia.edit();
    edytor.putString(PREFS_KEY, text);
    edytor.commit();
}

// Odczytywanie KLUCZY z SharedPreferences
public String getValue(Context context) {
    SharedPreferences ustawienia;
    String text;
     ustawienia = context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE); //1
    text = ustawienia.getString(PREFS_KEY, null); //2
    return text;
}

// Usuwanie pojedynczego klucza
public void removeValue(Context context) {
    SharedPreferences ustawienia;
    Editor edytor;
    ustawienia = context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
    edytor = ustawienia.edit();
    edytor.remove(PREFS_KEY);
    edytor.commit();
}

// Czyszczenie całego SharedPreferences
public void clearSharedPreference(Context context) {
    SharedPreferences ustawienia;
    Editor edytor;
    ustawienia = context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
    edytor = ustawienia.edit();
    ustawienia.clear();
    ustawienia.commit();
}

// Sprawdzanie, czy aplikacja jest uruchomiona pierwszy raz
SharedPreferences preferences = getSharedPreferences("myprefs", MODE_PRIVATE);
if (preferences.getBoolean("firstLogin", true)) {
        SharedPreferences.Editor editor = preferences.edit();
        editor.putString("firstLogin", false);
        editor.commit();
}
```

Gdzie jest AndroidManifest.xml?

Plik ten przetrzymuje ogólne definicje na temat *Activity* wchodzących w skład aplikacji.
Ścieżka: *NazwaProjektu / app / src / main / AndroidManifest.xml*
Można do niego dodać na przykład możliwość zapisywania na zewnętrznych nośnikach (np. zapisywania *SharedPreferences*):

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

lub pozwolenie na robienie fotek:

```
<uses-permission android:name="android.permission.CAMERA"/>
```

lub blokowanie orientacji ekranu:

```
<activity
        android:name=".MainActivity"
        android:screenOrientation="portrait">
```

lub blokowanie zmiany konfiguracji przy zmianie trybu na landscape i odwrotnie:

```
<activity
        android:name=".MainActivity"
```

android:configChanges="keyboardHidden|orientation|screenSize">

Tworzenie ikon programu

http://romannurik.github.io/AndroidAssetStudio/
Launcher Icons / Image / Download .ZIP
Rozpakowaną zawartość skopiować do: *nazwaProjektu / src / main / res / **
Istnieje także pakiet domyślnych ikon Androida do pobrania:
http://commondatastorage.googleapis.com/androiddevelopers/design/
Android_Design_Icons_20131106.zip

Debugowanie

Aby debugowanie zatrzymało się na konkretnej linii, należy ją zaznaczyć tzw. Breakpoint.
Aby zacząć debugować już uruchomioną aplikację, należy wydać polecenie *Run / Attach debugger to Android process*.

Komentowanie

W pliku XML (np. w *activity_main.xml*):
<!-- To jest komentarz... -->
W pliku Java (np. w *MainActivity.java*):
// To jest komentarz...

Błędy

Błąd przy starcie

Jeśli podczas pierwszego uruchomienia pojawi się błąd:

```
Internal error. Please report to https://code.google.com/p/android/issues
java.lang.RuntimeException: java.lang.IllegalArgumentException: Argument for @NotNull parameter 'name' of com/android/tools/idea/welcome/Platform.<init> must not be null
    at com.intellij.idea.IdeaApplication.run(IdeaApplication.java:178)
    at com.intellij.idea.MainImpl$1$1$1.run(MainImpl.java:52)
    at java.awt.event.InvocationEvent.dispatch(InvocationEvent.java:312)
    at java.awt.EventQueue.dispatchEventImpl(EventQueue.java:745)
    at java.awt.EventQueue.access$300(EventQueue.java:103)
    at java.awt.EventQueue$3.run(EventQueue.java:706)
    at java.awt.EventQueue$3.run(EventQueue.java:704)
    at java.security.AccessController.doPrivileged(Native Method)...
```

oznacza to, że program nie wykrył SDK ani lokalnie ani zdalnie. Należy sprawdzić (w *Linux*), czy system korzysta z SDK:
 # eselect java-vm list
Jeśli to nie odniesie skutku, należy przed pierwszym uruchomieniem odciąć dostęp do Internetu, program zapyta o prawidłową ścieżkę do java.
Jeśli i to nie pomoże, należy wyedytować plik */opt/android-studio/bin/idea.properties* (w *Windows* będzie to „*C:\Program Files\Android\Android Studio\bin\idea.properties*”) i dopisać w nim linię:
disable.android.first.run=true

Rendering problems

Jeśli przy otwieraniu projektu pojawi się komunikat „*This version of the rendering library is more recent than your version of android studio please update android studio*” (brak trybu WYSIWYG) – należy w menu WYSIWYG obniżyć wersję Androida np. do „*API 22: Android 5.1.1*” oraz odznaczyć „*Automatically Pick Best*”.

Jeśli pojawi się komunikat „*The following classes could not be found: android.support.v7.internal.app.WindowDecorActionBar*" – należy zmienić *theme rendera* na „*Light*”.

Jeśli w ogóle nie pojawi się renderowane urządzenie – należy również zmienić *theme rendera* na „*Light*”.

Cannot reload AVD list

Podczas uruchomienia projektu w emulatorze, pojawia się komunikat: „*Cannot reload AVD list: cvc-enumeration-valid: Value '280dpi' is not facet-valid...*”. Należy uruchomić *Tools/SDK Manager* i usunąć zbędne pakiety (wymienione w komunikacie), m.in. *Android Wear ARM EABI v7a System*

Image, Android Wear Intel x86 Atom System Image.

Brak aplikacji w emulatorze

Jeśli polecenie „*Run app*” uruchamia emulator, ale nie uruchamia w nim aplikacji, należy włączyć opcję „T*ools/Android/Enable ADB Integration*”.

Brak akceleracji sprzętowej (Intel HAXM, CPU with EPT+UG)

W wielu sytuacjach może pomóc następująca procedura:
1. Odinstaluj HAXM: *Tools/SDK Manager/SDK Tools/Intel x86 Emulator Accelerator (HAXM installer)*.
2. Ściągnij ze strony Intela oryginalne sterowniki HAXM i przekopiuj je do:
   C:\Users\nazwaUżytkownika\AppDate\Local\Android\Sdk\extras\intel
3. Uruchom plik "*intelhaxm-android.exe*" i zainstaluj.
4. Ponownie uruchom system.
5. Jeśli nadal nie działa, utwórz wirtualny smartfon na systemie *Adroid 19* (*KitKat*).

Info: Na starszym sprzęcie niestety nie będzie ta procedura działać (np. procesory *Dual Core*). Nie będzie ona także działać na maszynach wirtualnych (*VirtualBox*) nawet po aktywacji opcji "*Enable Nested VT-x/AMD-V*". W takich sytuacjach należy do testowania użyć połączeń USB (typu PTP, gdyż MTP rozłącza) do fizycznych smartfonów z włączoną funkcją "*Opcje programisty/Debugowanie USB*".

Unresolved reference... Conflicting overloads...

Powodem może być wybór języka Kotlin zamiast Java. Jeśli zmiana nie pomoże, należy zamknąć Android Studio i ponownie otworzyć.
Inna możliwość:
1. *Usunąć {folderProjektu}/.idea*.
2. *File / Sync Project with Grade Files*.

Jeszcze inna możliwość:
1. *Build / Clean Project*.
2. *Build / Make Projects*.

lub też:
1. *File / Invalidate caches / Restart*.

W ostateczności: utworzyć nowy projekt.

Importowanie projektów

Projekt stworzony w systemie *Windows* nie będzie działał w systemie Linux (inne ścieżki dostępu). Adaptacja projektu z innego systemu nie jest sprawą prostą, jeśli w ogóle możliwą.

Reset Androida

Aby przywrócić smartfona do ustawień fabrycznych w przypadku, gdy zapomnieliśmy hasła (na przykładzie *Samsung Galaxy Tab 4*), należy wyłączyć smartfon, a następnie:

- przytrzymać jednocześnie trzy przyciski: *POWER* + *VolumeUp*+*HOME*;
- za pomocą przycisku *Volume Up/Down* wybrać opcję "*wipe data / factory reset*" i zatwierdzić przyciskiem *POWER*;
- następnie wybieramy opcję "*Yes - delete all user data*".

Ostatnia aktualizacja: 30 kwietnia 2019.