Elementy widoku

1. Uruchom aplikację Android Studio.
2. Utwórz nowy projekt:
   2.1. SKD: 22 (android 5.1 Lollipop)
   2.2. Typ: Blank Activity (Basic Activity)
   2.3. Nie twórz automatycznie fragmentu
3. Otwórz plik res/layout/activity_main.xml w trybie tekstowym (kod XML)
4. Zmień położenie przycisku o id „fab”
5. Zmień ikonę przycisku „fab” (skorzystaj z autouzupełniania)
6. Otwórz plik res/layout/content_main.xml
7. Dodaj przycisk (Button) na środku ekranu
8. Dodaj parametr id dla obiektu RelativeLayout (ConstraintLayout) (np. „cont”)
9. Otwórz plik res/menu/main_menu.xml
10. Utwórz kilka elementów menu (obiekt item – analogicznie do istniejącego)
11. Uruchom aplikację i sprawdź poprawność wprowadzonych zmian.

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android=
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="pl.antons.layout.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main" />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="16dp"
        android:src="@android:drawable/ic_dialog_email" />

</android.support.design.widget.CoordinatorLayout>
```

W pliku znajdują się 3 kontrolki:

- Kontener AppBarLayout zawierający kontrolkę Toolbar
- Kontrolkę include
- Kontrolkę FloatingActionButton

Toolbar jest paskiem umieszczonym na górze ekranu, zawiera też menu (ukryte na razie)
FloatingActionButton jest przyciskiem, który wyświetlany jest na dole ekranu, nad innymi elementami (zawsze na wierzchu).
Kontrolka include powoduje załadowanie widoku z innego pliku, w tym przypadku content_main.xml. Zawartość tego zostaje wyświetlony jako element widoku.

MainActivity.java | content_main.xml | activity_main.xml | menu_main.xml

```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="pl.antons.lab3.MainActivity">
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="@string/action_settings"
        app:showAsAction="never" />
</menu>
```

Opis wszystkich argumentów obiektu item znajdziesz w dokumentacji: http://developer.android.com/guide/topics/resources/menu-resource.html

Pamiętaj, że wszystkie id powinny być różne.

Dzięki prefiksowi „@+” w przypadku braku danego zasobu zostaje on utworzony.

Obsługa interakcji

1. Otwórz plik MainActivity.java
2. Zmień akcję wywoływaną po przyciśnięciu przycisku „fab” na wywołanie Toasta:

```
Toast.makeText(getApplicationContext(),
"Kliknięto przycisk FAB",
Toast.LENGTH_SHORT).show();
```

W pliku MainActivity.java w odróżnieniu od EmptyActivity istnieją 3 metody.

Metoda onCreateOptionsMenu zawiera tworzenie elementów menu (w tym przypadku ładowanie obiektów item z pliku menu main.xml)

Metoda onOptionsItemSelected wywoływana jest w momencie wybrania elementu z menu

3. Utwórz publiczną funkcję „kliknij” która nic nie zwraca (void), a jako parametr przyjmuje obiekt typu View
4. W ciele funkcji kliknij dodaj Toasta z wiadomością „Kliknięto przycisk Button”
5. W funkcji onOptionsItemSelected zamień instrukcję warunkową if na switch, która będzie reagowała na utworzone elementy menu
6. Każdy case powinien zawierać Toasta z wiadomością odnośnie który przycisk został wywołany (np. jego id)
7. Otwórz plik content_main.xml i dla utworzonego przycisku ustaw właściwość onClick jako funkcję „kliknij”
8. Uruchom aplikację i sprawdź co się dzieje po naciśnięciu na poszczególne przyciski

W metodzie onCreate pojawiło się utworzenie triggera reagującego na kliknięcie przycisku „fab” (inną możliwością – łatwiejszą jest skorzystanie z właściwości onClick)

Zmienna item.getItemId() przyjmuje wartości, które zostały określone w pliku menu_main.xml jako zmienne id, np. R.id.action_settings dla android:id="@+id/action_settings"

Designer w widoku properties (prawy, dolny róg) dla właściwości onClick tworzy listę wyboru publicznych funkcji, które przyjmują jako argument obiekt typu View

Intencje

1. Utwórz nową aktywność typu LoginActivity (prawy przycisk myszy na drzewie aplikacji „app” new-> Activity->LoginActivity)
2. W pliku MainActivity.java dla dowolnej akcji wywoływanej przyciskiem utwórz obiekt intencji oraz uruchom nową activity (Login)

```
Intent intencja = new Intent(this, LoginActivity.class);
startActivity(intencja);
```

3. Utwórz zmienną typu static final int która będzie identyfikować akcję (jako zmienną globalną klasy, nie metody).
4. Dla innego przycisku stwórz obiekt intencji przyjmującej tylko jeden parametr: MediaStore.ACTION_IMAGE_CAPTURE
5. Uruchom nowe Activity za pomocą funkcji: startActivityForResult która jako parametry przyjmuje intencję z pkt 4 oraz zmienną static final int z pkt 3.
6. Uruchom aplikację i sprawdź co się dzieje po kliknięciu na przyciski

Abstrakcyjny mechanizm odpowiedzialny przede wszystkim za obsługę rozkazów wydawanych przez użytkownika (np. uruchamianie aktywności, uruchamianie usługi (z ang. service) czy nadanie komunikatu (broadcast))

Obiekt instancji przyjmuje jako parametr obiekt typu Context (this lub getApplicationContext()) oraz klasę, w postaci jej nazwy z sufiksem .class

MediaStore.ACTION_IMAGE_CAPTURE określa intencję związaną z akcją utworzenia zdjęcia. W tym przypadku określamy co chcemy osiągnąć (zrobić zdjęcie), a nie z jakiej aplikacji skorzystać.

```
Intent intencja = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(intencja, REQUEST_IMAGE_CAPTURE);
```

W przykładzie zmienna REQUEST_IMAGE_CAPTURE jest zdefiniowana na początku klasy jako:

```
static final int REQUEST_IMAGE_CAPTURE = 1;
```

W pkt 5. Tworzymy intencję która uruchamia aparat dzięki któremu można zrobić zdjęcie.

Obsługa danych powracających z intencji (nie zadziała, gdy aparat zwraca błąd)

1. W pliku MainActivity.java przeciąż metodę onActivityResult, która przyjmuje takie parametry jak:
   a. int requestCode,

Aby wykorzystać zrobione zdjęcie z poprzedniej części w naszej aplikacji należy je przechwycić z utworzonej intencji

requestCode zwraca kod wywołania intencji, w

   b. int resultCode,
   c. Intent data
2. W ciele metody onActivityResult utwórz obiekt klasy Bundle który przechwyci zdjęcie za pomocą metody getExtras()
3. Tworzymy obiekt typu Bitmap, który konwertuje otrzymane dane do postaci obrazka (bitmapy) za pomocą rzutowania oraz metody get(„data”) wywoływanej na utworzonym obiekcie Bundle
4. Aby zmodyfikować istniejący obiekt z poziomu kodu należy go zmapować np. dla Layoutu RelativeLayout któremu nadaliśmy id w pkt 8 pierwszej części „Elementy widoku” należy utworzyć zmienną danego typu – RelativeLayout i zmapować na dany typ wynik funkcji findViewById, którego argumentem jest id danego obiektu (id przechowywane są w obiekcie R.id. – autouzupełnianie pomaga w znalezieniu odpowiedniej nazwy obiektu)
5. Na zmapowanym obiekcie można użyć metody setBackground, która zmienia wartość parametru „android:background”
6. Metoda setBackground przyjmuje jako wartość obiekt typu Drawable, dlatego należy go utworzyć z naszej bitmapy (zmienna imageBitmap):

```
new BitmapDrawable(getResources(), imageBitmap)
```

7. Uruchom aplikację i sprawdź co się stało po zrobieniu zdjęcia (wywołanego przyciskiem z części „Intencje”)

tym przypadku będzie to wartość zmiennej static final int (REQUEST_IMAGE_CAPTURE)

resultCode zwraca kod poprawności wykonania akcji przez intencję (np. RESULT_OK)

data zwraca dane z aktywności (w danym przypadku zdjęcie)

```
Bundle extras = data.getExtras();
Bitmap imageBitmap = (Bitmap) extras.get("data");
RelativeLayout lay = (RelativeLayout)findViewById(R.id.cont);
lay.setBackground(new BitmapDrawable(getResources(), imageBitmap));
```

Przykładowy kod dla pkt 3-6. Pamiętaj o własnych nazwach zmiennych czy id obiektu.

Niestety android zmusza programistów do częstego rzutowania obiektów czy konwersji ich typów.

Zadanie 1 – Logowanie

1. Przeanalizuj kod pliku LoginActivity.java (kod nowo utworzonej aktywności w pkt 1 działu „intencje”)
2. Dodaj login i hasło swojego użytkownika (zmienna tablicowa DUMMY_CREDENTIALS)
3. Zmień walidację hasła (np. hasło musi mieć więcej niż 2 znaki)
4. Przy rejestracji dodaj Toasta „Próba utworzenia konta: {mail}”
5. Zablokuj przejście dalej (powrót z activity) przy rejestracji.

Wykonywanie sprawdzenia logowania czy rejestracji jest wykonywane w tzw. AsyncTask działającym w tle. Utworzenie w nim Toasta spowoduje błąd aplikacji i jej zamknięcie.

Jak wyświetlić Toasta by działało?

http://stackoverflow.com/questions/16830255/how-to-display-toast-in-asynctask-in-android

Zadanie 2 – Zmiana właściwości kontrolek

1. Dla jednego z utworzonych przycisków (lub nowego) ustal akcję, która po jego kliknięciu:
   - Wylosowania liczby losowej
   - W zależności od wylosowanej liczby zmień ikonkę wyświetlaną na danym przycisku
   - Ikonki powinny być zarówno z zasobów systemu (android.R.drawable) jak i z zasobów projektu (R.drawable) dodanych przez programistę (np. obrazek z internetu)

Zmiany dokonujemy w MainActivity.java

Liczbę losową można wygenerować z pomocą obiektu klasy Random (metoda nextInt)

Modyfikację można przeprowadzić za pomocą metody setImageDrawable która przyjmuje w parametrze obiekt typu Drawable

Aby pobrać obiekt Drawable z zasobu (mając jego ID) można skorzystać z funkcji getResources().getDrawable(id_zasobu, getTheme())