Проектування мобільних застосунків

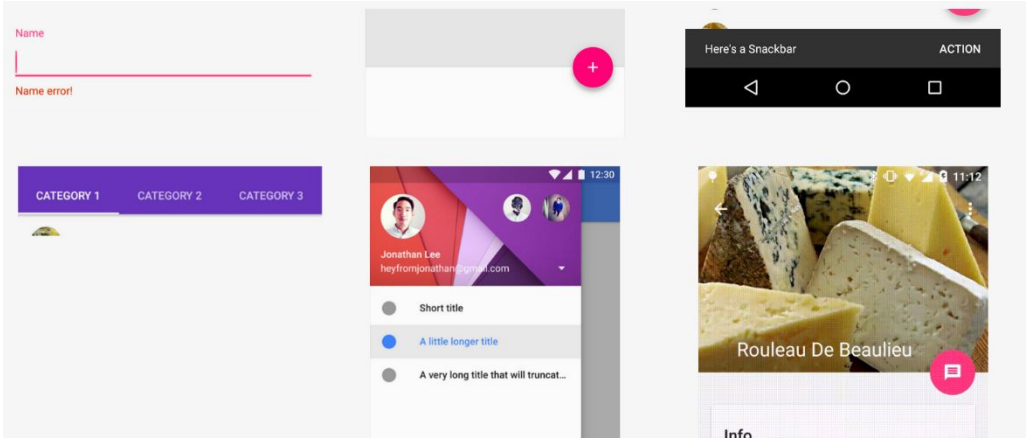
Лекція №6. Стандарти графічного інтерфейсу

Заняття проводить

Лимаренко Вячеслав Володимирович
slaw_lww@ukr.net
+38094-977-08-08

Material design (матеріальний дизайн)

Material design це вичерпна крос-платформна концепція візуального оформлення застосунків, яка зачіпає правила відображення анімації, загальний стиль програми та правила взаємодії з керуючими елементами.



Основні принципи

- плаский
- елемент – площина – матеріал
- реакція лише на верхню площину
- елемент може змінювати форму та повертатись
- :) але не може змінюватись у 3D (згини, заминки тощо)
- максимум однотонності
- рух в одному напрямку

Для використання потрібна ***Android Design Support Library***.

Матеріальний дизайн

Ви можете завантажити весь код прикладу на ваш комп'ютер…

https://github.com/googlecodelabs/android-design-library/archive/master.zip

…або клонувати GitHub репозиторій з командного рядка.

```
$ git clone
https://github.com-/googlecodelabs/android-design-library.git
```

Наступні інструкції опишуть як відкрити приклад в Android Studio:

1. Імпортуйте android-design-library пакет.

2. Виберіть кореневу директорію android_studio_folder.pngandroid-design-library (Quickstart> Import Project ...> android-design-library). Подальшу роботу ви будете вести в модулі Base-1.

3. Натисніть на кнопку  Gradle sync.

4. Увімкніть режим налагодження по USB на вашому Android пристрої. Підключіть ваш Android пристрій і натисніть кнопку  Run. Ви побачите домашній екран застосунку через кілька секунд.

Теми, Кольори і Шрифти

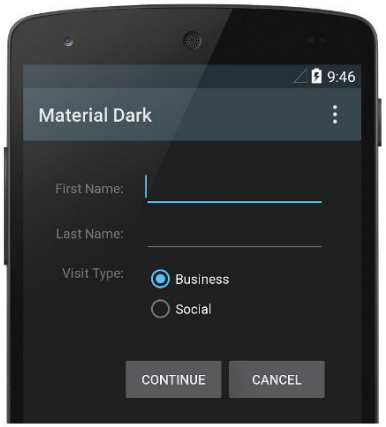


Figure 1. Dark material theme

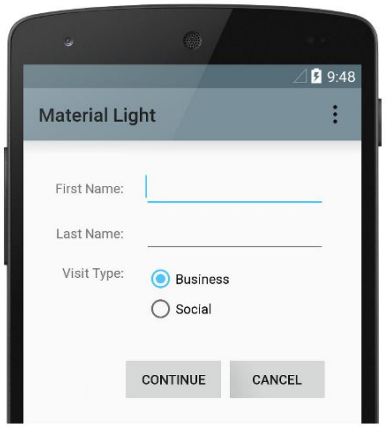


Figure 2. Light material theme

Налаштування кольорів проводиться через атрибути теми, які автоматично використовуються всіма компонентами застосунку, наприклад *colorPrimaryDark* для *Status Bar* і *colorPrimary* для *App Bar*

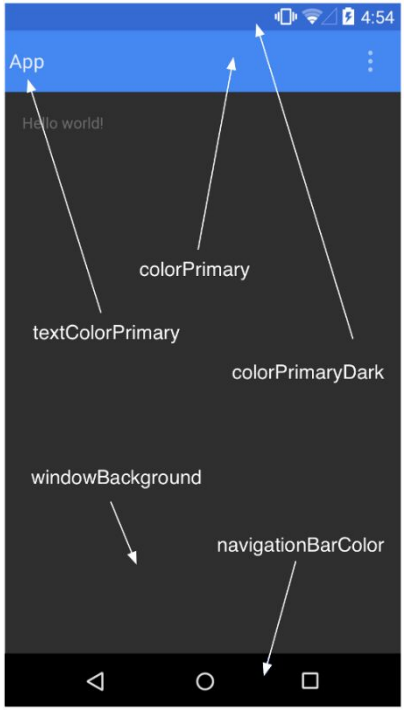
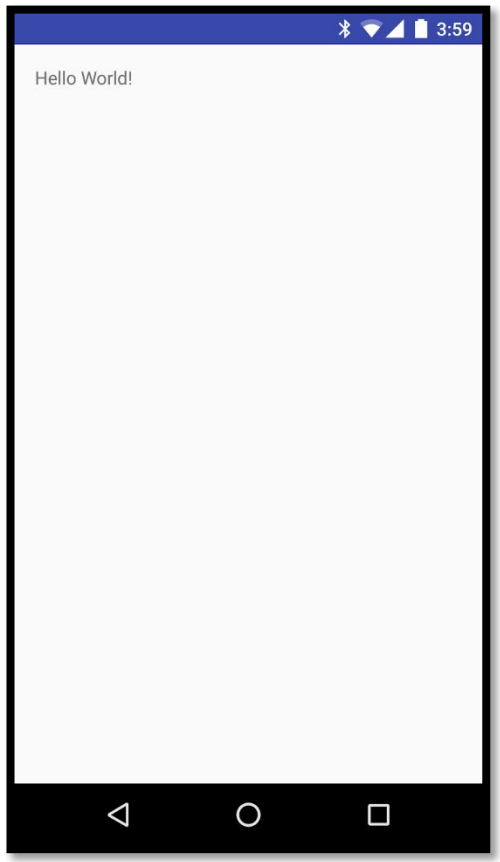


Figure 3. Customizing the material theme.

Додамо світлу тему в наш застосунок і налаштуємо частину кольорів в *res/values/styles.xml*

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme.Base" parent="Theme.AppCompat.Light.NoActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">#3F51B5</item>
        <!-- Light Indigo -->
        <item name="colorPrimaryDark">#3949AB</item>
        <!-- Dark Indigo -->
        <item name="colorAccent">#00B0FF</item>
        <!-- Blue -->
    </style>
    <style name="AppTheme" parent="AppTheme.Base"></style>
</resources>
```

Застосунок має виглядати наступним чином:



Шари і Анімація

Використовувані принципи

Material Design:

- тактильні поверхні
- виділені елементи
- Meaningful Motion

Додамо toolbar

Тепер ви готові писати поверх стартового проекту (Base-1).

1. Додамо *toolbar* і *tabs* в *activity_main.xml* і *MainActivity.java*
2. Видалимо *TextView* в *activity_main.xml*

activity_main.xml

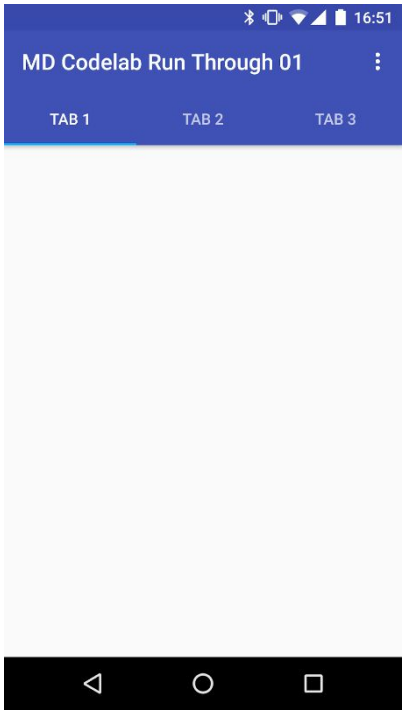
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:layout_scrollFlags="scroll|enterAlways"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />

</RelativeLayout>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {

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

        // Adding Toolbar to Main screen
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
```

Додамо *tabs* в *toolbar*

3. У *activity_main.xml* видалимо *RelativeLayout* і замінимо на *CoordinatorLayout*.
4. У *activity_main.xml* додамо *Tablayout* всередині *AppBarLayout*.
5. У *MainActivity.java* створимо клас *tabs* і ініціюємо меню



activity_main.xml

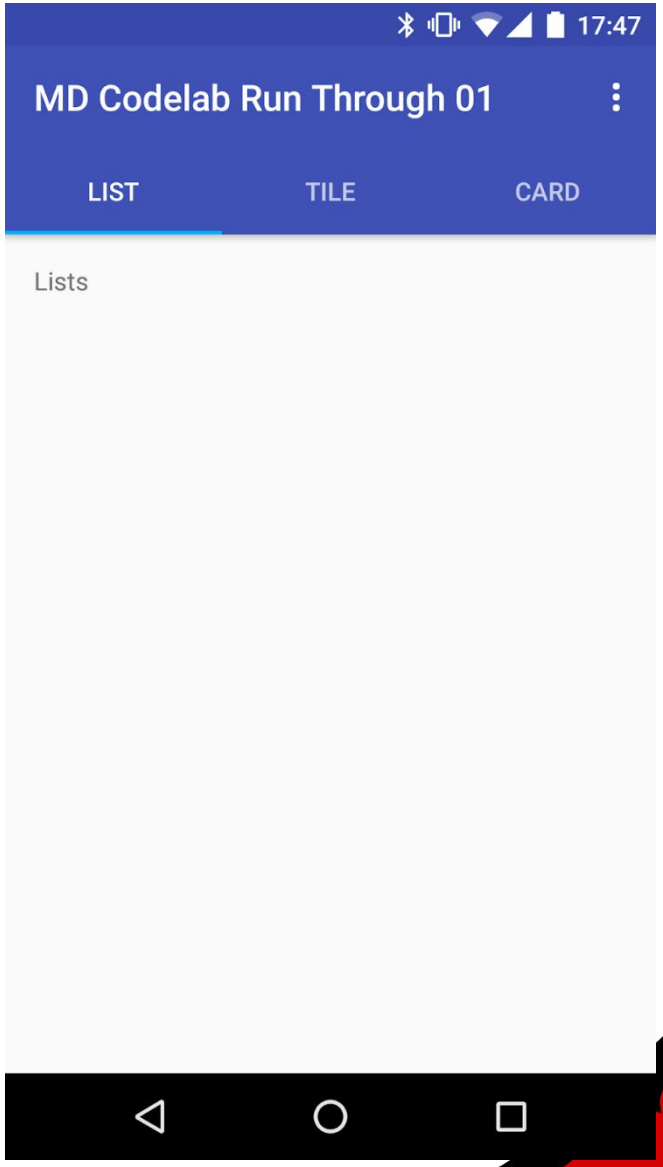
```
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/main_content"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

        <android.support.design.widget.AppBarLayout
            android:id="@+id/appbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

            <android.support.v7.widget.Toolbar
                android:id="@+id/toolbar"
                android:layout_width="match_parent"
                android:layout_height="?attr/actionBarSize"
                android:background="?attr/colorPrimary"
                app:layout_scrollFlags="scroll|enterAlways"
                app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />

            <android.support.design.widget.TabLayout
                android:id="@+id/tabs"
                android:layout_width="match_parent"
                android:layout_height="wrap_content" />

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

MainActivity.java

```
TabLayout tabs = (TabLayout) findViewById(R.id.tabs);
tabs.addTab(tabs.newTab().setText("Tab 1"));
tabs.addTab(tabs.newTab().setText("Tab 2"));
tabs.addTab(tabs.newTab().setText("Tab 3"));
```

Додамо Fragment и ViewPager

Створимо окремі контейтери в кожен з табів, щоб при навігації між ними у вас були різні рівні (шари).
6. Створимо 3 фрагмента. *ListContentFragment.java*, *TileContentFragment.java* і *CardContentFragment.java.*
7. У *MainActivity.java* створимо об'єкт *ViewPager* і *Adapter* для прокручування контенту.
8. У *activity_main.xml* додамо елемент *ViewPager* поза *AppBarLayout*.
9. Створимо файли, що описують зовнішній вигляд кожного з фрагментів: *item_list.xml*, *item_tile.xml* і *item_card.xml* всередині *res/layout/*. Зверніть увагу на індивідуальні атрибути, щоб зрозуміти які варіанти настройки доступні.



Стилізуємо кожен з екранів і додаємо *RecyclerView*

Використовувані принципи Material Design:

- тактильні поверхні
- картки

RecyclerView - це контейнер для відображення великих масивів даних, який може перегортуватися дуже ефективно (з великою швидкістю), утримуючи в пам'яті обмежену кількість елементів списку.

10. Додамо залежності в build.gradle для використання CardView і RecyclerView.

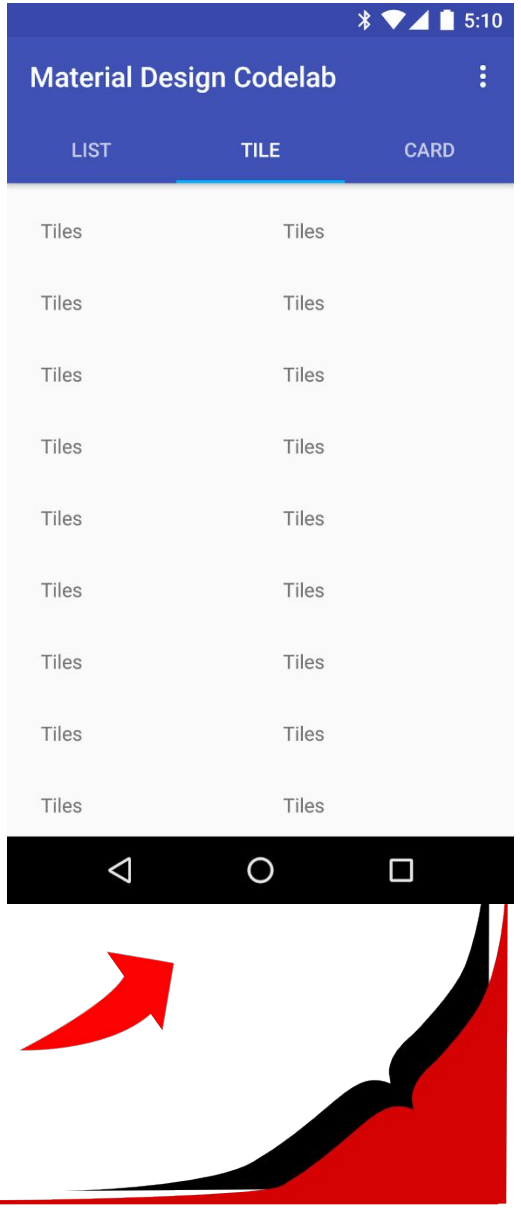
build.gradle

```
dependencies {
    compile 'com.android.support:appcompat-v7:23.0.1'
    compile 'com.android.support:design:23.0.1'
    compile 'com.android.support:cardview-v7:23.0.1'
    compile 'com.android.support:recyclerview-v7:23.0.1'
}
```

11. Створимо *recycler_view.xml* в теці *res/layout*.

recycler_view.xml

```
<android.support.v7.widget.RecyclerView xmlns:android="http://schemas.android.com/apk/re
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/my_recycler_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:clipToPadding="false"
    android:paddingBottom="@dimen/md_keylines"
    android:paddingTop="@dimen/md_keylines"
    android:scrollbars="vertical"
    app:layout_behavior="@string/appbar_scrolling_view_behavior" />
```



12. У кожному зі створених фрагментів створіть екземпляр *RecyclerView.Adapter* для настройки *RecyclerView*.

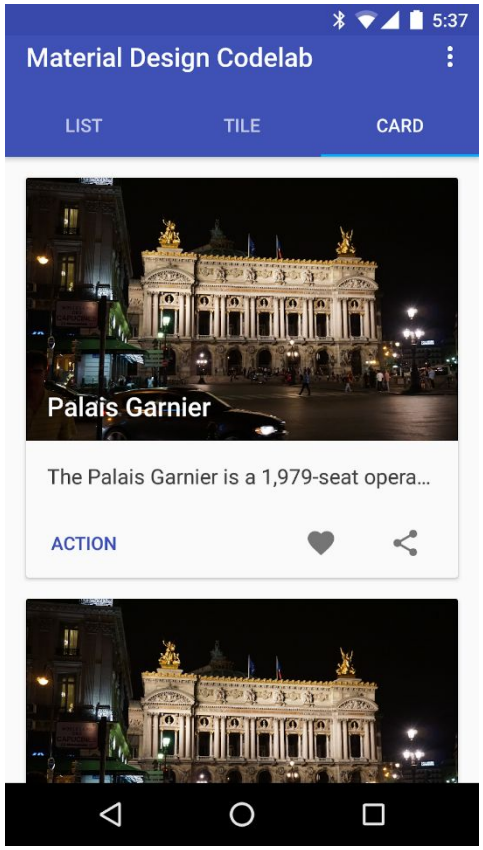
13. У кожному фрагменті налаштуйте *ViewHolder* для адаптера. Замініть *R.layout.item_tile* на *view*, який ви створите для кожного фрагмента.

14. Для кожного фрагмента поновіть *onCreateView (),* щоб використовувати *ContentAdapter*. Спеціально для *TileContentFragment* приділіть увагу налаштуванню відступів і вкажіть в *RecyclerView* використання *GridLayout* - це зробить зовнішній вигляд більш приємним на вигляд.

Щоб завершити ефект стилізуйте кожен *view* через зміну *item_list.xml*, *item_tile.xml*, щоб додати зображення.

Зробіть те ж саме для *item_card.xml*, обертаючи *RelativeLayout* в *CardView*, щоб отримати тіні і округлені кути, характерні для "карток".

(На Lollipop і новіших ОС, тіні будуть відображатися через нативну для Android властивість *elevation*. На старих пристроях *support library* симулює ефект використовуючи *drawables*).



Елементи сторінки

Використовувані принципи Material Design:

- тактильні поверхні
- Bold Elements

Додаємо NavigationDrawer

Навігаційна панель висувається зліва. Це загальний патерн, характерний для додатків Google і відповідний специфікації MD для листів.

1. Створіть файл *menu_navigation.xml*, який буде визначати елементи навігації в теці *res/menu*.

menu_navigation.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:checkableBehavior="single">
        <item
            android:icon="@drawable/ic_home_black_24dp"
            android:tint="@color/button_grey"
            android:title="One" />
        <item
            android:icon="@drawable/ic_favorite_black_24dp"
            android:tint="@color/button_grey"
            android:title="Two" />
        <item
            android:icon="@drawable/ic_bookmark_border_black_24dp"
            android:tint="@color/button_grey"
            android:title="Three" />
    </group>
</menu>
```

2. Створіть файл *navheader.xml,* що визначає зміст верхньої частини *Navigation Drawer*. Файл повинен бути розміщений в теці res/layout/.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="@dimen/navheader_height"
    android:background="?attr/colorPrimaryDark"
    android:orientation="vertical"
    android:padding="@dimen/md_keylines">
</LinearLayout>
```

3. У *activity_main.xml*: огорнемо всі компоненти в *DrawerLayout*, який дозволить витягати створюване меню через кордони екрану.

activity_main.xml

```
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/drawer"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true">
```

Додамо *NavigationView* ззовні від *CoordinatorLayout*.

activity_main.xml

```
<android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/navheader"
        app:menu="@menu/menu_navigation" />
```

11. Створимо *recycler_view.xml* в теці *res/layout*.

4. У *MainActivity.java*: додамо наступне поле на початку файлу *MainActivity*:

```
private DrawerLayout mDrawerLayout;
```

У метод *onCreate* всередині *MainActivity.java* додамо наступні рядки для налаштування бажаного меню.

```
// Create Navigation drawer and inflate layout
NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer);

// Adding menu icon to Toolbar
ActionBar supportActionBar = getSupportActionBar();
if (supportActionBar != null) {
    supportActionBar.setHomeAsUpIndicator(R.drawable.ic_menu_white_24dp);
    supportActionBar.setDisplayHomeAsUpEnabled(true);
}

// Set behavior of Navigation drawer
navigationView.setNavigationItemSelectedListener(
    new NavigationView.OnNavigationItemSelectedListener() {
        // This method will trigger on item Click of navigation menu
        @Override
        public boolean onNavigationItemSelected(MenuItem menuItem) {
            // Set item in checked state
            menuItem.setChecked(true);
            // TODO: handle navigation
            // Closing drawer on item click
            mDrawerLayout.closeDrawers();
            return true;
        }
    });
```

Додамо *mDrawerLayout.openDrawer* (*GravityCompat.START*); в *MainActivity.java* додамо методи для відкриття меню по торканню на кнопку "гамбургер" в меню.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();
    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    } else if (id == android.R.id.home) {
        mDrawerLayout.openDrawer(GravityCompat.START);
    }
    return super.onOptionsItemSelected(item);
}
```

Додамо Floating Action Button (FAB) і викличемо появу SnackBar

Ширяюча кнопка дії - поетичний переклад FAB - використовується для основної дії в вашому додатку і підкреслює свою силу і важливість «літаючи» над UI вашого аппа, як єдина кругла кнопка.
SnackBar - елемент, який забезпечує легку для сприйняття відповідь від інтерфейсу на операцію, проведену користувачем.

Поради по дизайну:

Не кожен екран потребує FAB. FAB використовується тільки для базової дії в застосунку.

Тільки одна кнопка *floating action button* може перебувати на одному екрані, для спрощення і підкреслення її важності. Вона повинна представляти тільки саму загальну дію.

Floating action button не обов'язково повинна бути розміщена в нижньому правому куті екрану. Верхній лівий, верхній правий кут, перекриття *Toolbar* так само є хорошим місцем для її розміщення.

FAB зменшеного розміру може використовуватися в ситуаціях, коли цього вимагає стиль інших елементів на цьому екрані.

1. У *activity_main.xml* додамо *FloatingActionButton* в кінці *CoordinatorLayout*.
2. У *MainActivity.java*, додамо «слухач» на дотик FAB, який буде створювати *SnackBar*

activity_main.xml

```
<android.support.design.widget.FloatingActionButton
            android:id="@+id/fab"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="right|bottom"
            android:layout_marginBottom="@dimen/md_keylines"
            android:layout_marginRight="@dimen/md_keylines"
            android:src="@drawable/ic_add_white_24dp" />
```

MainActivity.java

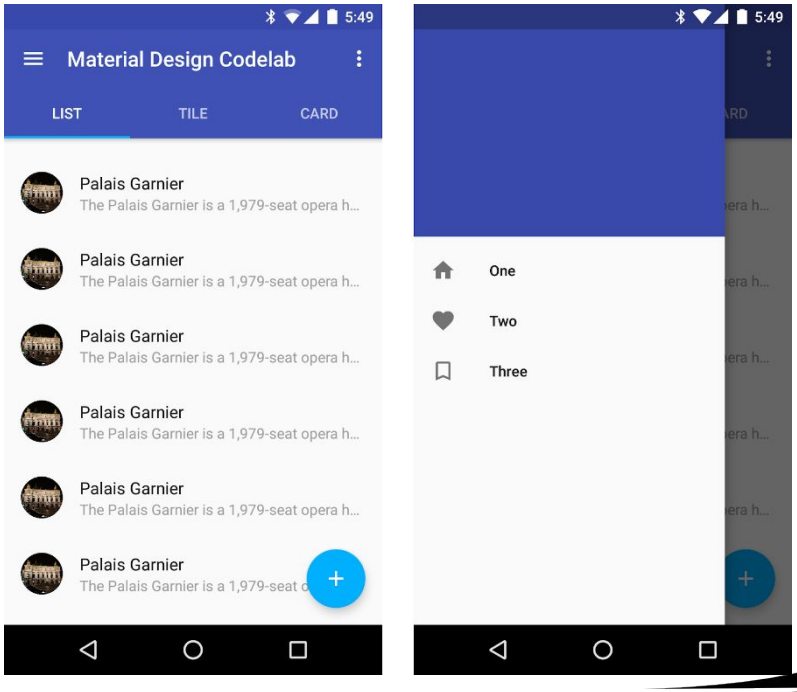
```
// Adding Floating Action Button to bottom right of main view
        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Snackbar.make(v, "Hello Snackbar!",
                        Snackbar.LENGTH_LONG).show();
            }
        });
```

3. В теці *values-2*, оновимо *styles.xml*, щоб зробити системну плашку прозорою (на Android 5 і вище).

styles.xml

```
<resources>
    <style name="AppTheme" parent="AppTheme.Base">
        <item name="android:windowDrawsSystemBarBackgrounds">true</item>
        <item name="android:statusBarColor">@android:color/transparent</item>
    </style>
</resources>
```

4. У *CardContentFragment.java* додамо метод, коли кнопка натискається, показується snackbar. Додаток має виглядати наступним чином:



Зверніть увагу, що *snack bar* піднімає *floating action bar* при анімації на екрані. Ця поведінка забезпечується *CoordinatorView* в *activity_main.xml.*

В якості експерименту спробуйте винести *FloatingActionButton* за межі *CoordinatorView*. (Для цього необхідно буде оголосити додатковий *RelativeLayout*.) *Snack bar* тепер буде ігнорувати існування FAB при анімації своєї появи!

Оскільки анімація це невід'ємна частина Material Design, перевірте, що всі *view* були поміщені всередину *CoordinatorView*, і зробіть розширення *CoordinatorLayout.Behavior* якщо ваші *views* потребують здійснення деякої анімації.

Detail View з toolbar, що згортається

Використовувані принципи Material Design:

- Meaningful motion
- Print-like design

Створимо Detail View

Використовуємо *Intent*, щоб дозволити користувачам переміщатися між картками та їх повною репрезентацією. Створимо такий *detail view* в нашому додатку:

1. Створимо *DetailActivity.java* і *activity_detail.xml*.

2. Для кожного з елементів списку *ListContentFragment.java*, *TileContentFragment.java*, *iCardContentFragment.java* створимо новий *Intent*, щоб кожен з елементів списку мав посилання на екран з розгорнутою інформацією про контент елемента.

```
public static class ViewHolder extends RecyclerView.ViewHolder {
    public ViewHolder(LayoutInflater inflater, ViewGroup parent) {
        super(inflater.inflate(R.layout.item_list, parent, false));
        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Context context = v.getContext();
                Intent intent = new Intent(context, DetailActivity.class);
                context.startActivity(intent);
            }
        });
    }
}
```

3. Не забудьте прописати ваш *activity* в *AndroidManifest.xml*.

AndroidManifest.xml

```
<activity
    android:name=".DetailActivity"
    android:parentActivityName=".MainActivity">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".MainActivity" />
</activity>
```

Перехід до Collapsing Toolbar

Collapsing Toolbar згортає *toolbar*, коли користувач прокручує екран вниз і розгортає при прокручуванні вгору.

1. У *activity_detail.xml*, додамо *AppBarLayout* і *CollapsingToolbarLayout* разом з *CoordinatorLayout*.

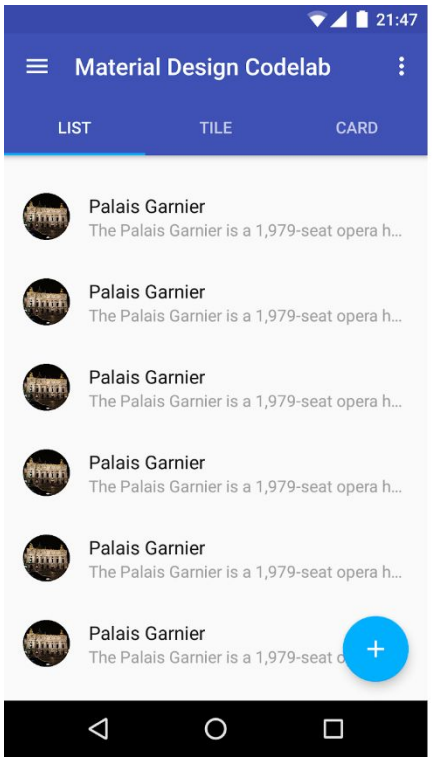
activity_detail.xml

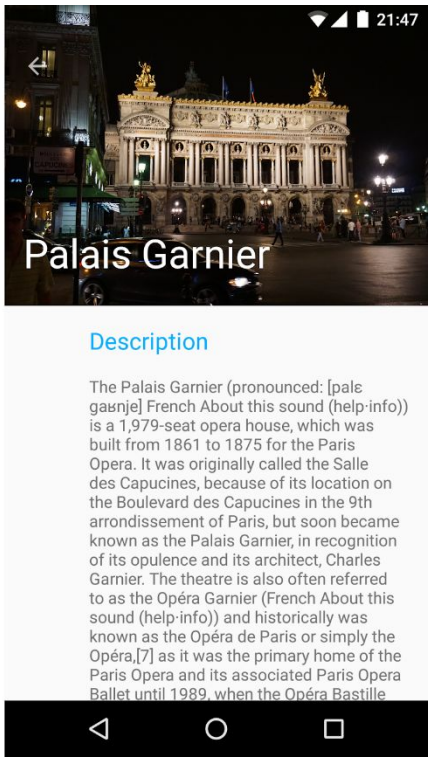
```
<android.support.design.widget.AppBarLayout
    android:id="@+id/appbar"
    android:layout_width="match_parent"
    android:layout_height="@dimen/app_bar_height"
    android:fitsSystemWindows="true"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

    <android.support.design.widget.CollapsingToolbarLayout
        android:id="@+id/collapsing_toolbar"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fitsSystemWindows="true"
        android:theme="@style/ThemeOverlay.AppCompat.Dark"
        app:contentScrim="?attr/colorPrimary"
        app:expandedTitleMarginEnd="@dimen/article_keylines"
        app:expandedTitleMarginStart="@dimen/md_keylines"
        app:layout_scrollFlags="scroll|exitUntilCollapsed">

        <ImageView
            android:id="@+id/image"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@drawable/paris"
            android:fitsSystemWindows="true"
            android:scaleType="centerCrop"
            app:layout_collapseMode="parallax" />

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            app:layout_collapseMode="pin"
            app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />

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

2. У *DetailActivity.java* встановимо заголовок для *CollapsingToolbar*.

DetailActivity.java

```
// Получим объект Collapsing Toolbar для настройки
CollapsingToolbarLayout collapsingToolbar =
        (CollapsingToolbarLayout) findViewById(R.id.collapsing_toolbar);
// Установим заголовок страницы
collapsingToolbar.setTitle(getString(R.string.item_title));
```





Список

Основні компоненти

- візуальний - ListView
- дані для візуалізації - Adapter

Основні події ListView

- onItemClick()
- onItemSelect()
- onScroll()

Список

Задачі ListView

- відображати елементи
- прокручувати елементи

Задачі Adapter

- вибрати набір елементів
- взяти необхідний елемент по індексу
- взяти відповідну розмітку елемента та віддати ListView

Список

Задачі ListView

- відображати елементи
- прокручувати елементи
- **реагувати на події від користувача**
- **видаляти елемент**

Задачі Adapter

- вибрати набір елементів
- взяти необхідний елемент по індексу
- взяти відповідну розмітку елемента та віддати ListView
- **обробити видалення даних**

Список

Основні методи Adapter

- getItem()
- getView()
- getItemId()
- getCount()
- notifyDataSetChanged()

ViewHolder

Основні задачі

- зберігати посилання на необхідні елементи
- економити час на пошук елементів