ПОЧАТОК РОБОТИ З ANDROID

План:

Введення в платформу Android

Бурхливий розвиток інформаційних технологій останнім часом призвів до того, що з'явилося багато нових пристроїв і технологій, таких, як планшети, смартфони, нетбуки, інші гаджети. Вони все більше входять в наше життя і стають звичною справою. Лідируючої платформою серед подібних гаджетів на сьогоднішній день є ОС Андроїд.

Android використовується на самих різних пристроях. Це і смартфони, і планшети, і телевізори, і смарт-годинник і ряд інших гаджетів. За різними підрахунками за 2017 рік цієї операційною системою користуються близько 85% власників смартфонів, а загальна кількість користувачів сматрфонів на ОС Android оцінюється в 1,5 млрд. Чоловік по всьому світу.

ОС Андроїд була створена розробником Енді Рубіном (Andy Rubin) в якості операційної системи для мобільних телефонів і спочатку розвивалася в рамках компанії Android Inc. Але в 2005 році Google купує Android Inc. і починає розвивати операційну систему з новою силою. На даний момент (листопад 2018 року) останньою версією є Android 9.0 Oreo, яка вийшла в серпні 2010 року:

версія	кодове ім'я	дата випуску	рівень API	Частка ринку (вересень 2018)
9.0	Pie	6 Серпня 2018	28	<0.1%
8.1	Oreo	5 Грудня 2017	27	7.5%
8.0	Oreo	21 серпня 2017	26	14%
7.1	Nougat	4 жовтня 2016	25	10.1%
7.0	Nougat	22 серпня 2016	24	18.1%
6.0	Marshmallow	5 жовтня 2015	23	21.3%

5.1	Lollipop	9 березня 2015	22	14.4%
5.0	Lollipop	3 листопада 2014	21	3.5%
4.4	KitKat	31 жовтня 2013	19	7.6%
4.3	Jelly Bean	24 липня 2013	18	0.4%
4.2	Jelly Bean	13 листопада 2012	17	1.5%
4.1	Jelly Bean	9 липня 2012	16	1.1%
4.0	Ice Cream Sandwich	16 грудня 2011	15	0.3%
2.3	Gingerbread	9 лютого 2012	10	0.2%

Як видно з таблиці, актуальними версіями є Android Oreo, Nougat, Marshmallow і Android Lollipop, на які слід орієнтуватися.

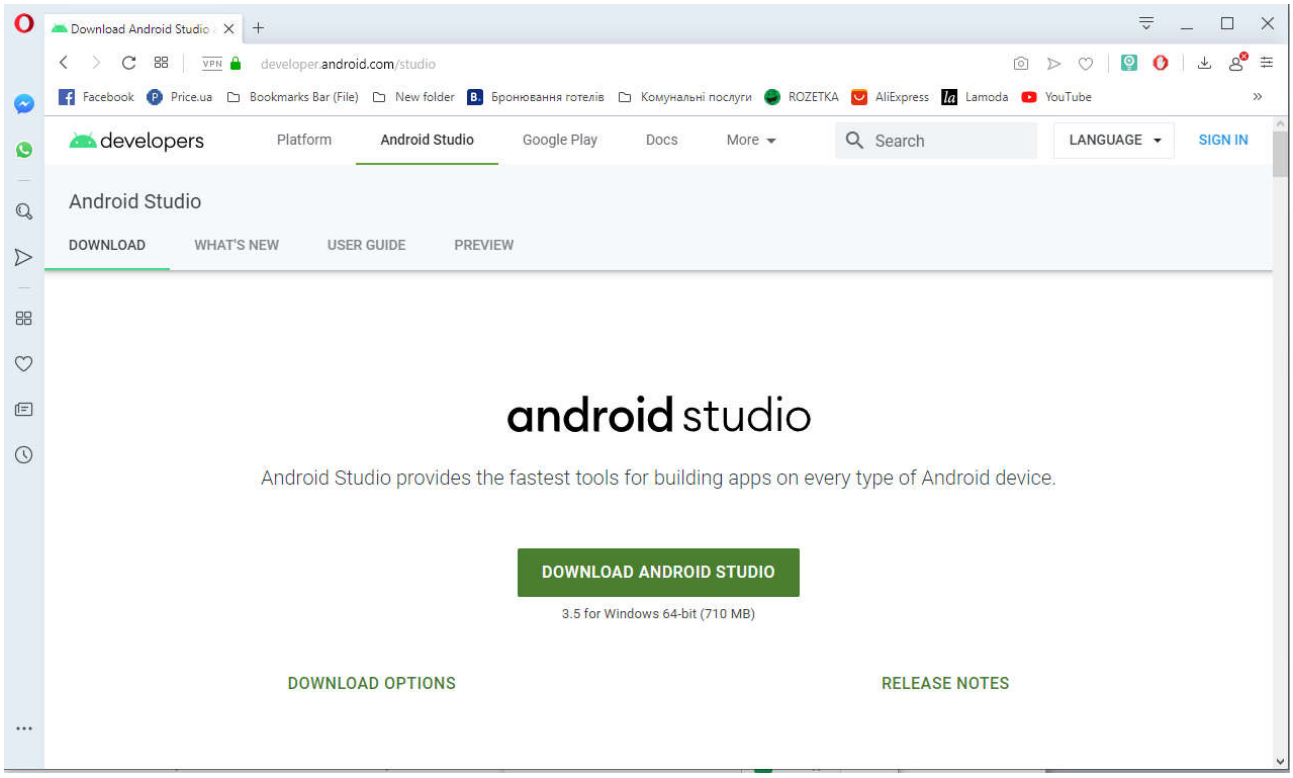
Встановлення засобів розробки

Java

В першу чергу для створення додатків завантажимо і встановимо пакет JDK 8, який необхідний для розробки на мові Java. JDK 8 можна знайти на сайті компанії Oracle: https://www.oracle.com/technetwork/java/javase/downloads/index.html

Встановлення Android Studio

Існують різні середовища розробки для Android. Можна використовувати для розробки такі середовища як NetBeans, Eclipse, Visual Studio. Рекомендованої середовищем розробки є **Android Studio** , тому ми її і будемо використовувати. Завантажити файл інсталятора можна з офіційного сайту: https://developer.android.com/studio. Для скачування пакету установки для OS Windows треба натиснути на кнопку "Download Android Studio for Windows":

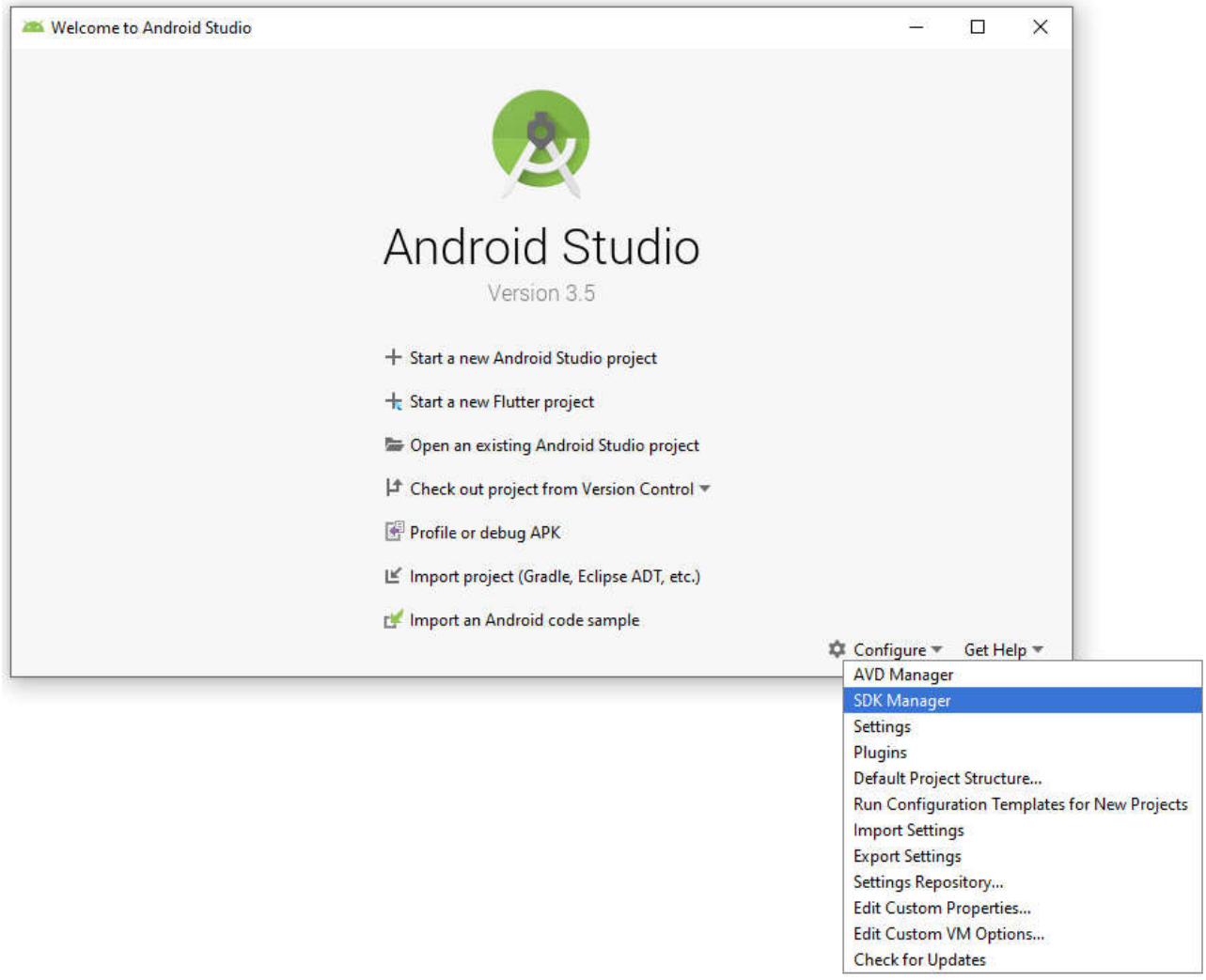


В процесі установки на комп'ютер будуть встановлені крім самого середовища Android Studio також і набір інструментів Android SDK.

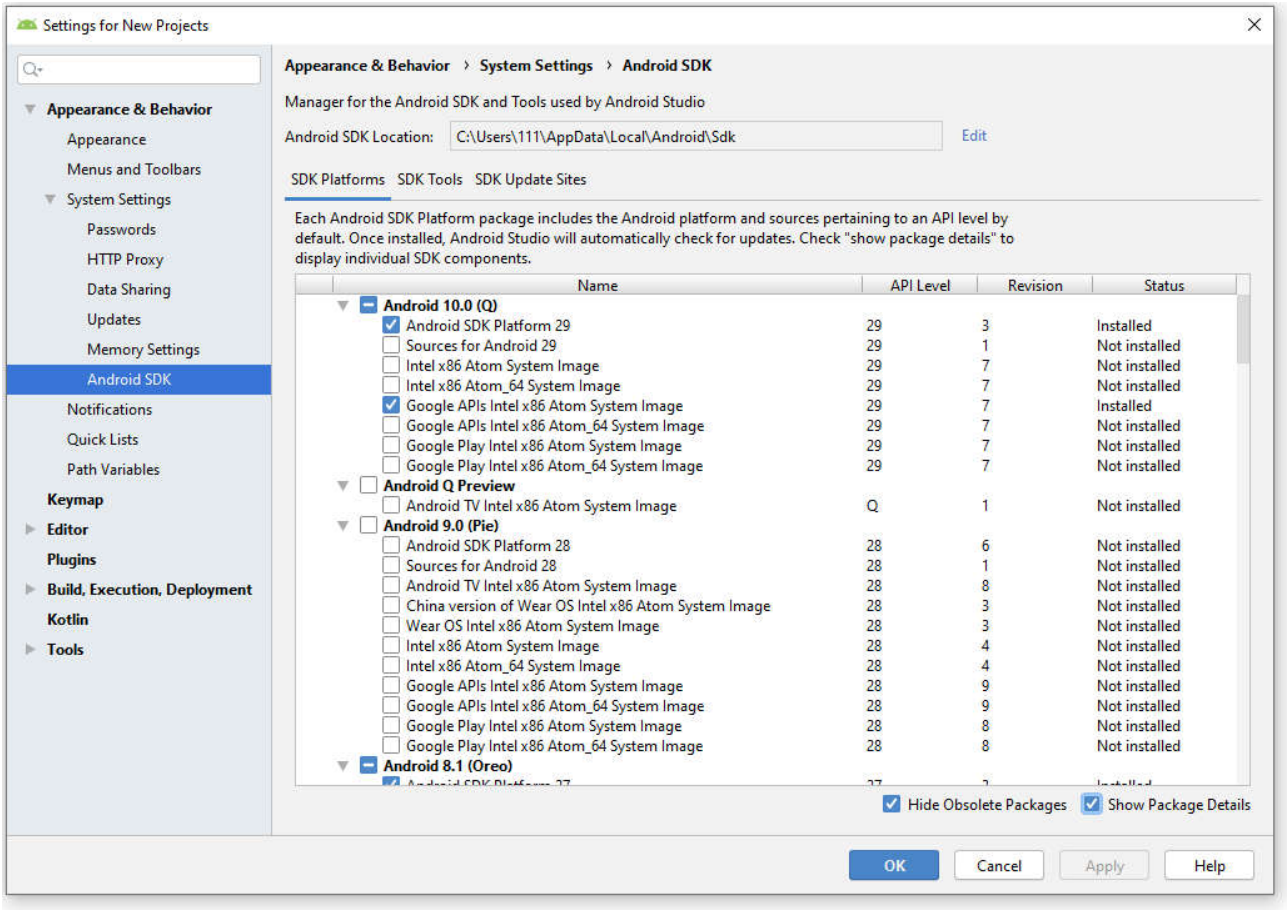
Налаштування Android SDK

Все, що ми робимо на Android за допомогою Java, залежить від Android SDK - якщо ми створюємо додаток під певну версію, наприклад, для Android Nougat, то у нас повинні бути встановлені відповідні іструменти SDK. Це треба враховувати при розробці.

Відкриємо Android Studio. За замовчуванням, якщо ми запускаємо програму в перший раз, то нам відкривається деяке початкове меню. У самому низу стартового екрана програми знайдемо кнопку "Configure" і натиснемо на неї:



Далі в випадаючому меню натиснемо на пункт "SDK Manager". Після цього відкриється вікно з настройками для Android SDK Manager:



Для більш детального перегляду всіх компонентів по кожній платформі натиснемо внизу вікна на посилання **Show Package Details** .

Тут ми детально можемо подивитися, які пакети для кожної платформи встановлені. Всі компоненти згруповані за певною версією SDK, наприклад, Android API 29, Android 10.0 (Q), Android 8.1 (Oreo) і так далі. Кожна версія SDK фактично представляє певну версію або підверсію ОС Android. Але кожна версія SDK передбачає широке коло компонентів, це в тому числі і інструменти для розробки під телевізори, під смарт-годинник і т.д. Не всі ці інструменти можуть знадобитися, тому немає сенсу всі версії SDK абсолютно повністю встановлювати.

В даному випадку нас перш за все буде цікавити пункт **Android SDK Platform** . Android SDK Platform містить весь основний функціонал, який використовується при розробці. Даний пункт можна виділити для всіх тих платформ, під які ми збираємося компілювати додаток. Зокрема, рекомендую вибрати останні платформи - Android 10.0, а також ті, які будуть використовуватися в якості мінімальних платформ, наприклад, Android 7.0.

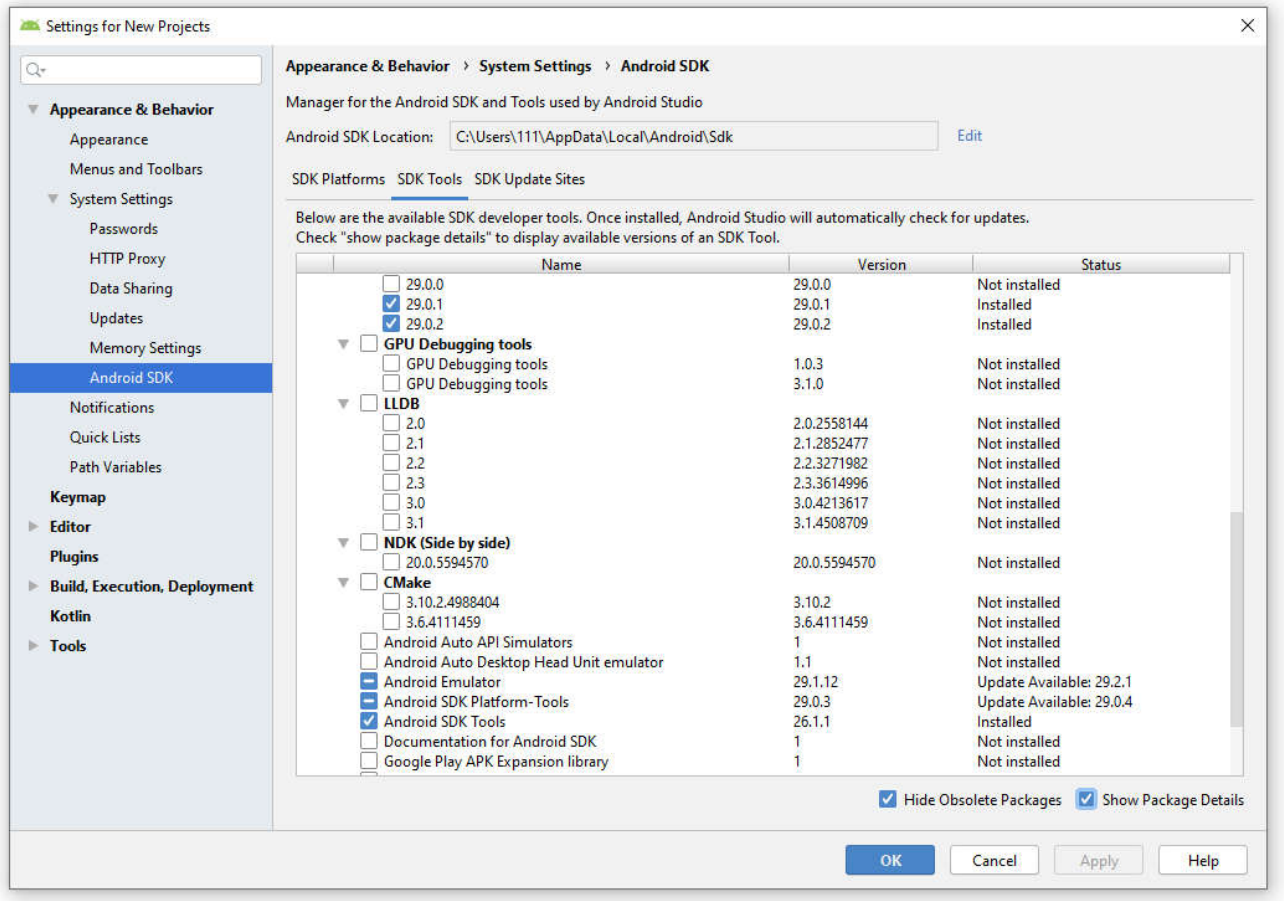
Крім SDK Platform кожна платформи, як правило, містить ще ряд компонентів, які призначені для тестування програми на емуляторі:

- Android TV Intel x86 Atom System Image;
- Android Wear Intel x86 Atom System Image;
- Google APIs Intel x86 Atom System Image;
- Google Play Intel x86 Atom System Image.

Для більш ранніх платформ список компонентів може відрізнятися.

Якщо не планується використання емулятора, то дані компоненти не настільки важливі. І навпаки, якщо тестування буде відбуватися на емуляторі, то слід встановити для цього образ системи **Google APIs Intel x86 Atom System Image** або **Google Play Intel x86 Atom System Image** .

Далі в цьому ж вікні перейдемо на вкладку **SDK Tools**. Тут перераховані додаткові пакети:



Тут для нас насамперед важливі такі пакети як:

- **Android Support Repository**
- **Android Support Library**
- **Google Play Services**
- **Google Repository**
- **Google Usb Driver**

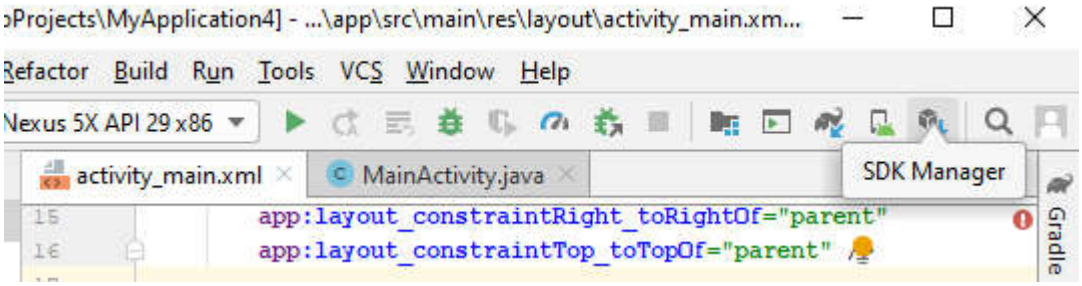
Ці пакети встановлюють репозиторії android і сервіси google play. Крім того, якщо ви хочете використовувати для тестування додатків смартфон від Google - Nexus 5/5X і 6/6P або Google Pixel, то обов'язково треба встановити пакет **Google Usb Driver** , який представляє драйвер для роботи з даними смартфонами. Якщо ж передбачається використовувати смартфон іншого виробника, то в цьому випадку треба буде встановити USB-драйвер безпосередньо від цього виробника. Як правило, при підключенні смартфона система сама намагається встановити драйвер.

Якщо планується використання емулятора, то також слід встановити пакет **Intel x86 Emulator Accelerator (HAXM installer)** .

І якщо в процесі розробки або тестування будуть виникати проблеми з якимись версіями ОС Android, то цілком ймовірно корінь проблем полягає у відсутності потрібних компонентів для певних платформ. В цьому випадку можна буде з Android Studio запустити SDK Manager і вручну встановити відсутні компоненти.

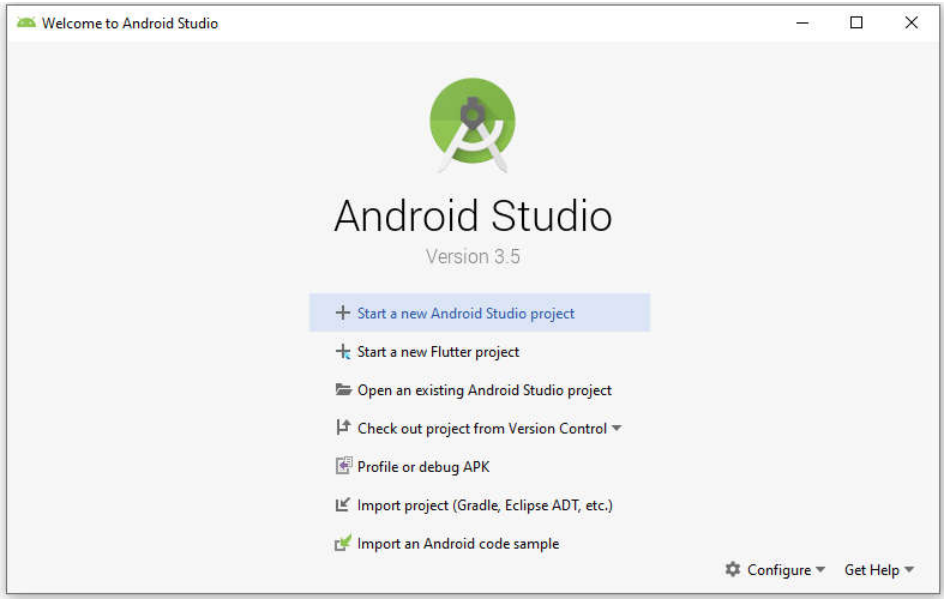
Згодом при кожному запуску Android Studio будуть з'являтися спливаючі сповіщення про доступність оновлень по платформах, що досить зручно і що дозволить проектам не відставати від останніх оновлень від Google.

Якщо згодом нам буде потрібно викликати SDK Manager, то ми можемо зробити це з самої Android Sudio через панель інструментів:

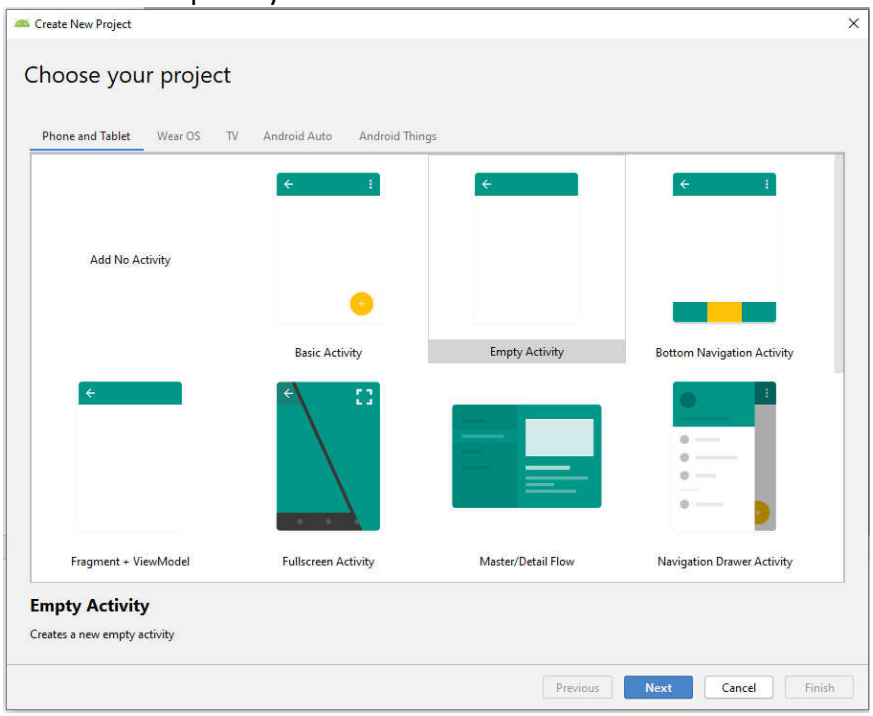


Android Studio і створення першого проекту

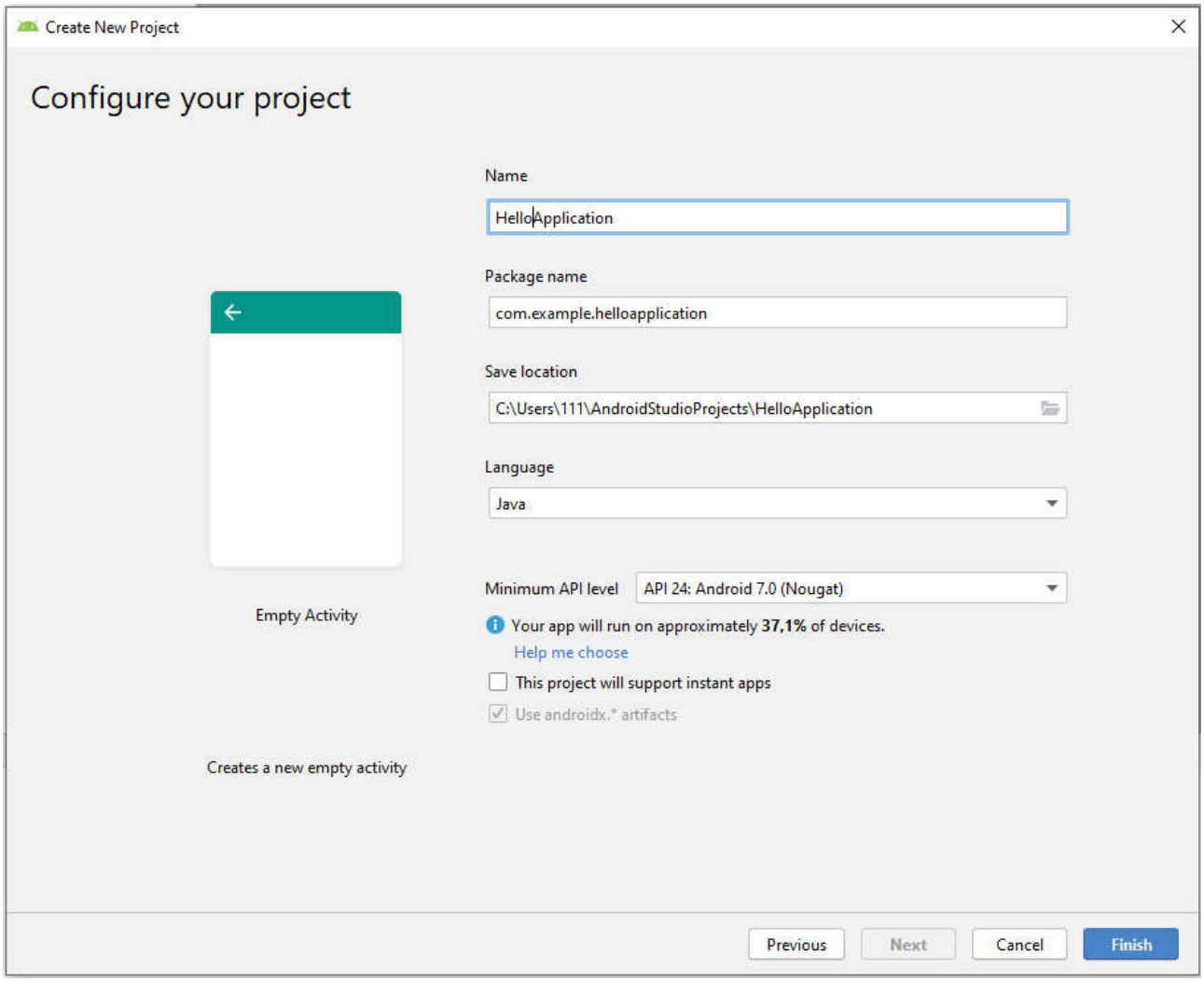
Тепер створимо перший додаток в середовищі Android Studio для операційної системи Android. Відкриємо Android Studio і на початковому екрані виберемо пункт **Start new Android Project :**



(Якщо Android Studio вже запускалася, то в меню треба вибрати пункт File->New->New Project ...). Після цього відобразиться діалогове вікно створення нового проекту, де необхідно вибрати шаблон проекту:



Android Studio надає ряд шаблонів для різних ситуацій, але найпоширенішими є **Basic Activity** і **Empty Activity**. Це найзручніші шаблони для старту для створення більшості додатків. В даному випадку ми виберемо шаблон **Empty Activity**.



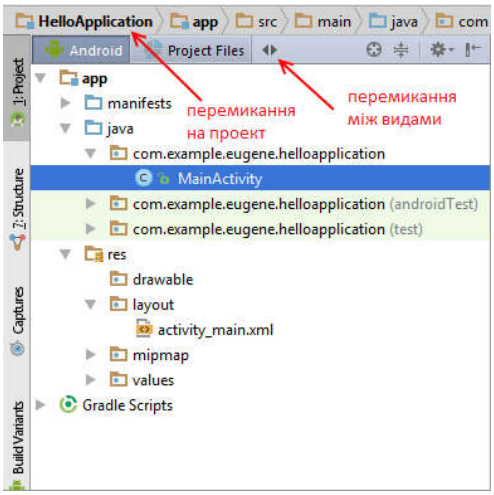
У вікні створення нового проекту ми можемо встановити його початкові настройки:

- В полі **Name** вводиться назва програми. Вкажемо в якості імені назву **HelloApplication**;
- В полі **Package name** вказується пакет класів, де буде розміщуватися головний клас додатку. Краще також відразу поставити яке-небудь своє значення в поле Package name. Справа в тому, що коли ви будете розміщувати додаток в магазині Google Play, то значення для цього поля повинно бути унікальним для всього магазину. Хоча для тестових проектів, як в даному випадку, можна залишити в цьому полі значення за замовчуванням;
- В полі **Save location** можна встановити розташування файлів проекту на жорсткому диску;
- В полі Language вибираємо мову програмування для даного проекту, якщо не встановленно додаткових мов програмування, то доступно дві Kotlin і Java, тут вибираємо Java;
- Наступним встановлюється мінімальний рівень API, він визначає на якому мінімальному пристрої може бути запущений додаток. Та відразу показується кількість пристроїв які зможуть його підтримувати (слід зазначити, що під цей мінімальний API має бути встановлений відповідний SDK).

Далі натиснемо на кнопку Finish.

Структура проекту

Після створення проекту структура проекту Android відображається в наступному вигляді:

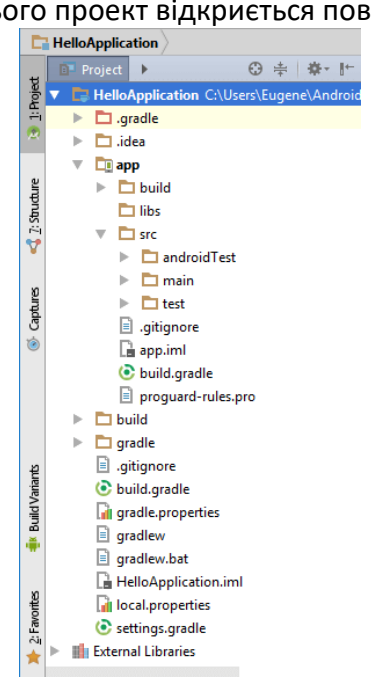


Проект Android може складатися з різних модулів. За замовчуванням, коли ми створюємо проект, створюється один модуль - **app**. Модуль має три папки:

- **manifests**: зберігає файл маніфесту AndroidManifest.xml, який визначає конфігурацію;
- **java**: зберігає файли коду на мові java, які структуровані по окремих пакетах;
- **res**: містить використовувані в додатку ресурси.

Окремий елемент **Gradle Scripts** містить ряд скриптів gradle (як для модуля app або інших можливих модулів, так і для всього проекту), які використовуються при побудові програми.

Перейдемо до повної структурі проекту. Для цього можна подвійним клацанням натиснути на назву проекту. Після цього проект відкриється повністю:



Розглянемо повну структуру проекту програми під ОС Android, яка створюється за замовчуванням. Тут також ми побачимо єдиний модуль проекту - модуль app. Власне весь код, з яким ми будемо працювати, розташовується всередині цього модуля.

Всі модулі в проекті описуються файлом **setting.gradle** . За замовчуванням він має наступний вміст:

```
include ':app'
```

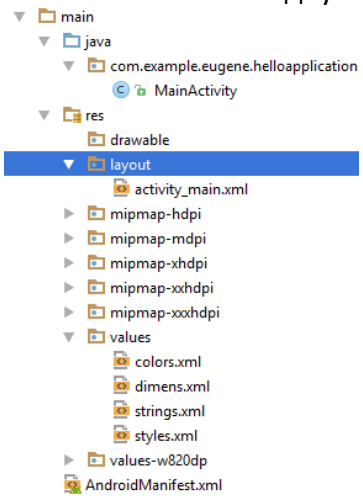
Файл **build.gradle** містить інформацію, яка використовується при побудові проекту.

Кожен модуль має свій файл **build.gradle** , який визначає конфгурацію побудови проекту, специфічну для даного модуля. Так, якщо ми подивимося на вміст папки app, то як раз знайдемо в ній такий файл. На початковому етапі дані файли не настільки важливі, досить лише розуміти, для чого вони потрібні.

У модулі **app** ми можемо побачити кілька папок і файлів, з яких для нас найважливішими є:

- каталог **libs** - призначений для зберігання бібліотек, використовуваних додатком
- каталог **src** - призначений для зберігання вихідного коду. Він містить ряд підкаталогів. Каталоги **androidTest** і **test** призначені для зберігання файлів тестів додатки. А власне вихідні коди розташовуються в папці **main** .

Папка main має складну структуру:

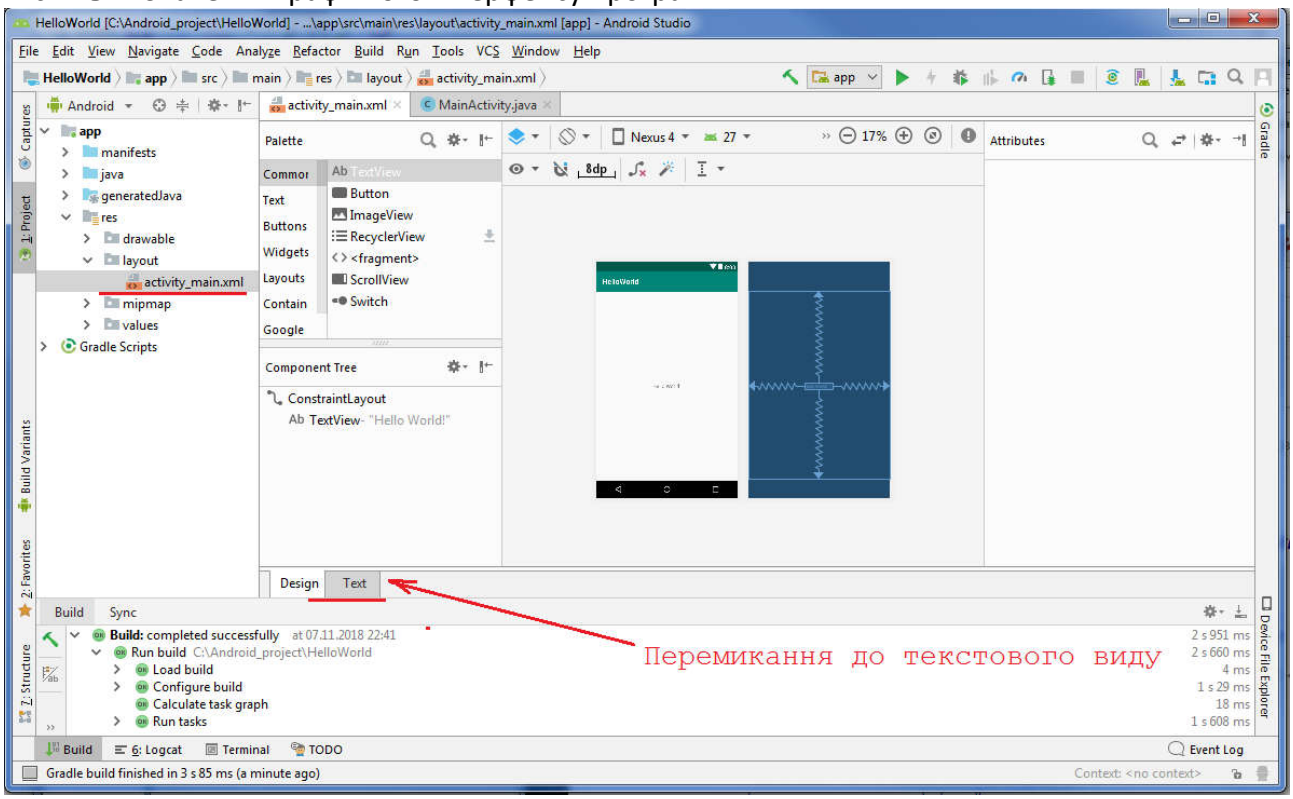


- **AndroidManifest.xml** представляє файл маніфесту, який описує фундаментальні характеристики додатку, його конфігурацію і визначає кожен з компонентів цього додатку.
- Папка **java** містить вихідні файли програми. За замовчуванням в ньому знаходиться файл класу `MainActivity`, який запускається за замовчуванням при старті програми.
- Папка **res** містить каталоги з ресурсами. Зокрема, вона містить такі каталоги:
- папка **drawable** призначена для зберігання зображень, використовуваних в додатку;
- папка **layout** призначена для зберігання файлів, що визначають графічний інтерфейс. За замовчуванням тут є файл *activity_main.xml* , який визначає інтерфейс для єдиною в проекті activity – MainActivity;
- папки **mipmap-xxxx** містять файли зображень, які призначені для створення іконки програми при різних роздільних здатностях екрану. І для кожного виду роздільної здатності тут є свій каталог
- папка **values** зберігає різні xml-файлів, що містять колекції ресурсів - різних даних, які застосовуються в додатку.

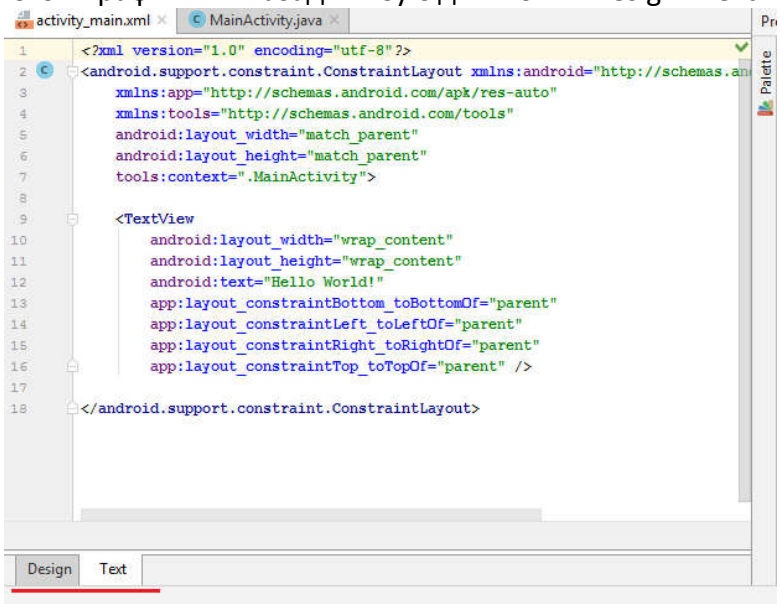
Перший додаток

Стандартний проект, який був створений в минулій темі, вже містить деякий функціонал. Правда, цей функціонал майже нічого не робить, тільки виводить на екран рядок "Hello world!".

У студії за замовчуванням повинен бути відкритий файл *activity_main.xml* , який містить визначення графічного інтерфейсу програми.



Якщо файл відкритий в режимі дизайнера, а в центрі Android Studio відображається дизайн програми, то нам треба перемкнути вид файлу в текстовий. Як змінити спосіб - з текстового в графічний і назад внизу є дві кнопки Design і Text.



Тепер змінимо код програми, щоб воно виводило на екран рядок "Привіт Андроїд". Для цього змінимо код у файлі activity_main.xml , який зараз виглядає приблизно так:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
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">

 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Hello World!"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintLeft_toLeftOf="parent"
 app:layout_constraintRight_toRightOf="parent"
 app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

Змінимо в цьому файлі визначення елемента **TextView**, який і відповідає за виведення текстової інформації на екран мобільного апарату. Сам текст, що виводиться задається за допомогою атрибута **android:text**. Тому змінимо весь код в файлі ***activity_main.xml*** наступним чином:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/activity_main"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:padding="16dp">

 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Привіт Андроїд!!" />
</RelativeLayout>
```

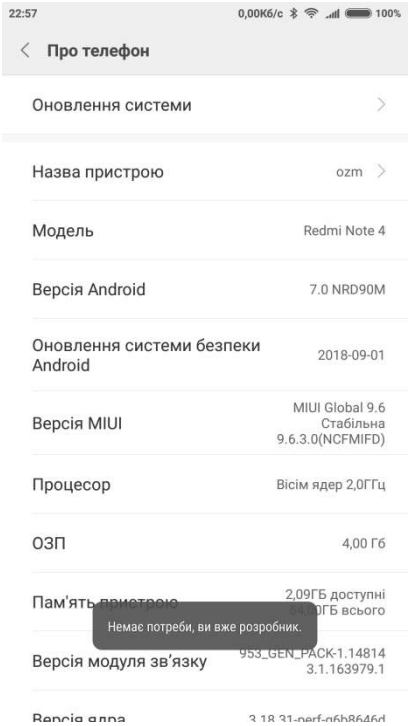
Після збереження файлу ми можемо перейти до графічного виду і побачимо, що графічний дизайнер автоматично оновиться і буде виводити вже рядок, який ми тільки що визначили.

Режим розробника на телефоні

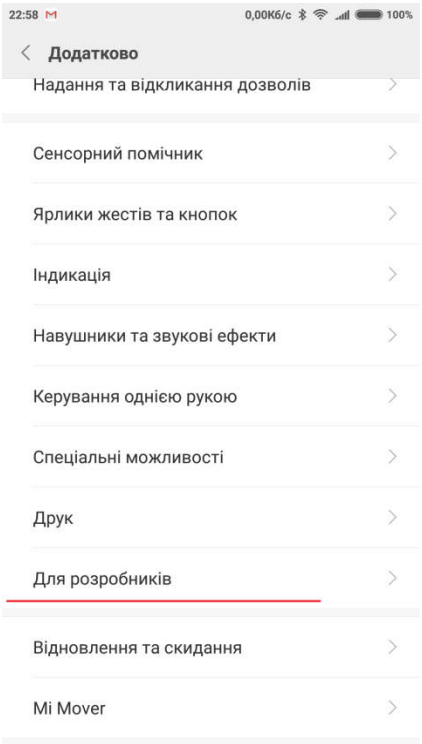
Для запуску і тестування програми ми можемо використовувати емулятори або реальні пристрої. Але в ідеалі краще тестувати на реальних пристроях.

Для використання мобільного пристрою для тестування на робочу машину необхідно встановити драйвер. Якщо смартфон від Google - Nexus 5/6 / 5x/6P або Google Pixel, то для його підтримки необхідно через SDK Manager встановити пакет **Google Usb Driver** . Якщо ж виробник апарату - інший вендор, то треба встановити то USB-драйвер, який поставляється даними вендором. Якщо ОС - Windows 10, то там, як правило, система сама може знайти через центр оновлень драйвер і встановити його.

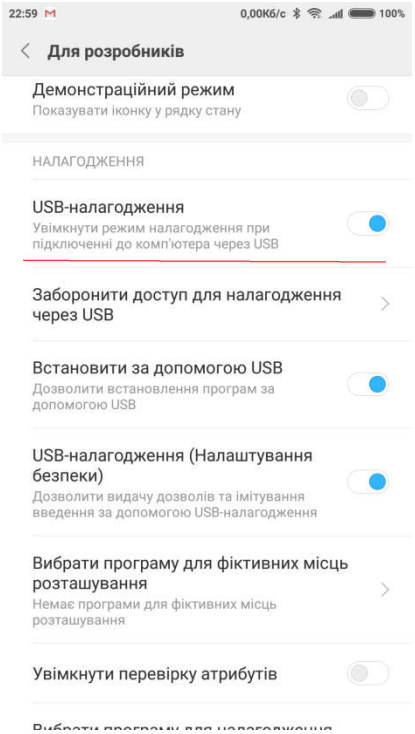
За замовчуванням опції розробника на смартфонах приховані. Щоб зробити їх доступними, треба зайти в **Settings> About phone (Установки> Про телефон)** і сім раз натиснути **Build Number (Номер збірки)** .



Поверніться до попереднього або там ви побачите доступний пункт Developer options (Для розробника) .

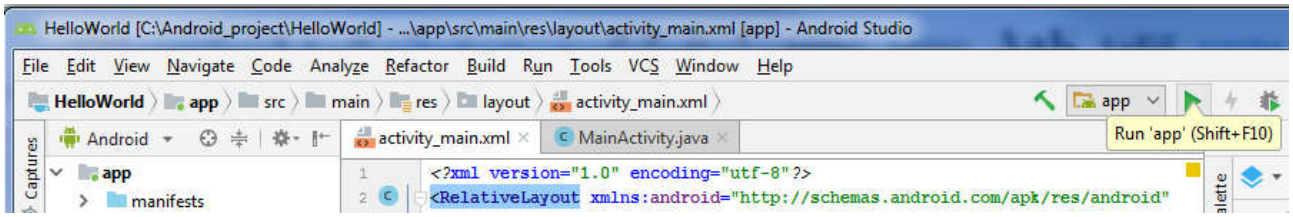


Перейдемо до пункту Для розробників і включимо можливість налагодження по USB:

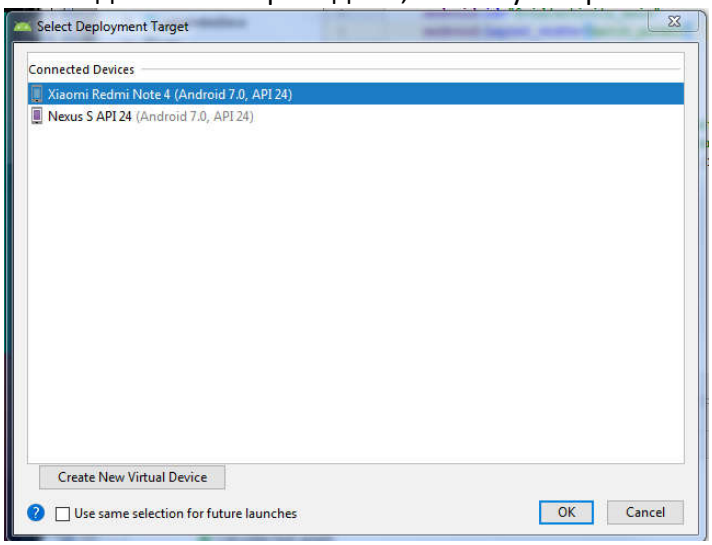


Запуск програми

Підключимо пристрій з ОС Android (якщо ми тестуємо на реальному пристрої) і запустимо проект, натиснувши на зелену стрілку на панелі інструментів.



Потім почнеться побудова проекту. Даний процес може зайняти деякий час, після чого відобразиться діалогове вікно для вибору пристрою для запуску. Тут ми можемо вибрати підключений до комп'ютера гаджет, або емулятор:



Виберемо пристрій і натиснемо на кнопку ОК. І після запуску ми побачимо наш додаток на екрані пристрою.

Клас Activity і ресурси

У минулому параграфі ми розглянули створення найпростішого додатка, змінили вміст файлу activity_main.xml і вивели на екран рядок. Однак ми ще не говорили, як файл activity_main.xml пов'язаний з графічним інтерфейсом. Тому розглянемо клас Activity і файли ресурсів, які визначають візуальний інтерфейс.

Activity є класом, який по суті представляє окремий екран (сторінку) додатків або їхніх візуальний інтерфейс. Окремі activity, які вже безпосередньо використовуються в додатку, є спадкоємцями цього класу. Додаток може мати одну activity, а може і кілька. Кожна окрема activity задає окреме вікно для відображення.

Розглянемо код Activity, використаної в минулій темі, код якої генерується автоматично в Android Studio (файл коду можна знайти в проекті в папці *src/main/java*):

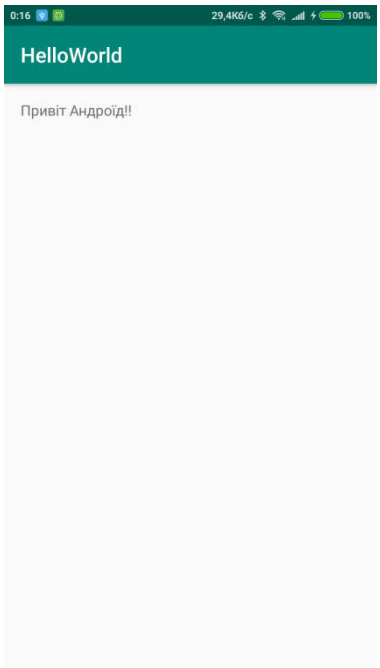
```
package com.example.ozm.helloworld;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

Клас MainActivity представляє звичайний клас java, на початку якого йдуть визначення пакету і імпорту зовнішніх пакетів. За замовчуванням він містить тільки один метод `onCreate()`, в якому фактично і створюється весь інтерфейс програми.

У методі `OnCreate()` йде звернення до методу батьківського класу і установка ресурсу розмітки дизайну:

```
super.onCreate (savedInstanceState);
setContentView (R.layout.activity_main);
```

Згодом ми докладніше розглянемо всі ці вирази, а поки досить розуміти, для чого вони потрібні. І в підсумку виходить приблизно наступний візуальний інтерфейс:



Щоб встановити ресурс розмітки дизайну, викличе метод setContentView, в який передається ідентифікатор ресурсу.

Зверніть увагу на те, як виглядає ідентифікатор ресурсу: **R.layout.activity_main**. Фактично це і є посилання на файл *activity_main.xml*, який знаходиться в каталозі *res/layout* :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/activity_main"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:padding="16dp">

 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Hello World!" />
</RelativeLayout>
```

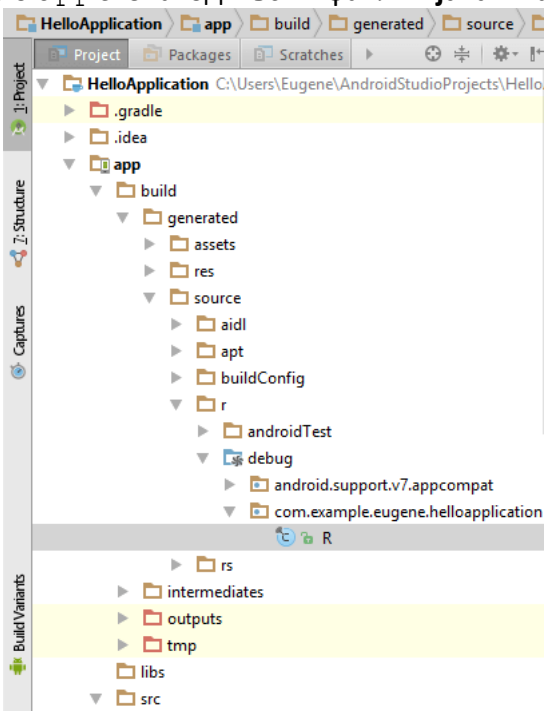
Весь інтерфейс представлений елементом-контейнером **RelativeLayout** , який містить один компонент - текстове поле TextView. Текстове поле встановлює текст за допомогою атрибута **`android:text`**.

```
 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Hello World!" />
```

Таким чином, під час запуску програми спочатку запускається клас MainActivity, який в якості графічного інтерфейсу встановлює розмітку з файлу activity_main.xml.

Однак в класі MainActivity ми використовуємо не файли, а ідентифікатори ресурсів: **R.layout.activity_main** .

Всі ідентифікатори ресурсів визначені в класі `R`, який автоматично створюється утилітою `apptі` знаходиться в файлі **R.java** в каталозі *build/generated/source/r/debug* :



Клас `R`містить ідентифікатори для всіх ресурсів, розташованих в каталозі *res* . Для кожного типу ресурсів в класі R створюється внутрішній клас (наприклад, для всіх графічних ресурсів з каталогу *res/drawable* створюється клас `R.drawable`) і для кожного ресурсу даного типу присвоюється ідентифікатор. З цього ідентифікатора згодом можна витягти ресурс у файлі коду.

При оновленні ресурсів під час компіляції цей файл також оновлюється.

Наприклад, за замовчуванням є ресурс розмітки інтерфейсу *activity_main.xml* , який передається через ідентифікатор `R.layout.activity_main`. Але якщо ми додамо в папку res/layout новий ресурс розмітки, наприклад, *my_layout.xml* , то клас R автоматично перекомпілюється, і ми зможемо відразу ж використовувати цей ресурс:`setContentView(R.layout.my_layout);`

Створення графічного додатку

Графічний інтерфейс користувача грунтується на ієрархії об'єктів `View`і `ViewGroup`. Об'єкти `View` представляють віджети, такі як кнопки або текстові поля. Об'єкти `ViewGroup`є контейнери для віждетів, керують їхнім розташуванням і компонуванням.

Отже, створимо новий проект програми по тому способу, який ми використовували раніше. Або використовуємо попередній проект.

Створення лінійної розмітки

Обраний при створенні проекту для activity шаблон EmptyActivity додає всі визначення графічного інтерфейсу в файл *activity_main.xml* , які знаходяться в проекті в папці *res/layout* , тому змінимо файл activity_main.xml наступним чином:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
```

```
 android:orientation="horizontal" >
</LinearLayout>
```

Тут визначається елемент **LinearLayout**, який є підкласом ViewGroup і який має в своєму розпорядженні дочірні елементи в горизонтальний або вертикальний ряд. Орієнтація елементів вказується в атрибуті **android:orientation**. Кожен елемент в контейнері LinearLayout відображається на екрані в тому порядку, в якому він оголошений у файлі XML.

Інші два атрибути - **android:layout_width** і **android:layout_height** потрібні для всіх віджетів для визначення розмірів. Оскільки **LinearLayout** є кореневим елементом компонування, він повинен заповнити весь простір екрану, тому для його висоти і ширини вказується значення **"match_parent"**. Дане значення розтягує віджет до кордонів батьківського контейнера.

Додавання текстового поля

Для додавання текстового поля всередині елемента `LinearLayout`створимо елемент **EditText**. Як і для будь-якого об'єкта View, для `EditText`також треба оголосити певні xml-атрибути:

```
<EditText android:id="@+id/edit_message"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_weight="1"
 android:hint="Введіть повідомлення" />
```

Отже, ми визначили наступні атрибути:

android: id : забезпечує унікальний ідентифікатор віджета, за яким ми можемо посилатися на об'єкт

Знак (`@`) для посилання на контрольний об'єкт у файлі XML. Після нього йде тип ресурсу (в даному випадку `id`), слеш і потім ім'я ресурсу (`edit_message`).

Знак плюса (`+`) перед типом ресурсу необхідно, коли в перший раз визначається ID ресурсу. При компіляції програми SDK використовує ім'я ID для створення нового ресурсу ID у файлі `gen/R.java`. Після цього більше не потрібно вживати знак плюса.

android: layout_width і **android: layout_height** : для цих властивостей встановлюється значення `wrap_content`, яке задає для віджетів величини, достатні для відображення в контейнері

android: layout_weight : дозволяє визначити займану полем ширину. Значення 1 в даному випадку дозволяє розтягнути поле на всю ширину.

android: hint : вказує на текст, який буде відображатися в текстовому полі за замовчуванням, коли воно порожнє.

Додавання кнопки

Тепер додамо в файл activity_main.xml кнопку - елемент `Button` відразу після елемента `EditText`:

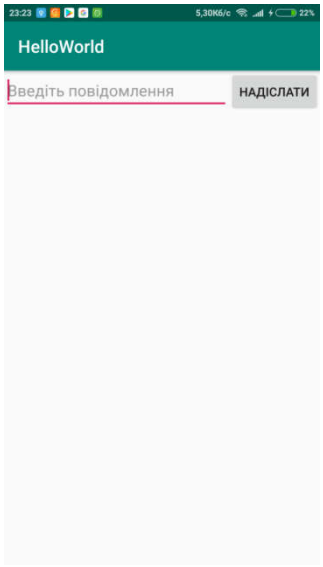
```
<Button
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Надіслати" />
```

У випадку з кнопкою її висота і ширина також мають значення `wrap_content`, тому кнопка буде мати ті розміри, які є достатніми для виведення на ній її тексту. Для кнопки не потрібно вказувати атрибут `android:id`, оскільки ми не будемо на неї посилатися в коді MainActivity

В результаті файл буде мати наступний вигляд:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="horizontal" >
 <EditText android:id="@+id/edit_message"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_weight="1"
 android:hint="Введіть повідомлення" />
 <Button
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Надіслати" />
</LinearLayout>
```

І на даний момент додаток буде мати наступний інтерфейс:



Запуск другої Activity

У попередньому параграфі темі ми створили додаток з використанням візуальних елементів: додали до activity текстове поле для введення повідомлення і кнопку. Тепер продовжимо попередній проект і додамо запуск нової activity після натискання на кнопку.

Обробка натискання кнопки

Для підключення обробника кнопки відкриємо файл **activity_main.xml** і додамо в елемент `Button` атрибут **android: onClick** :

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="horizontal" >
 <EditText android:id="@+id/edit_message"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_weight="1"
 android:hint="Введіть повідомлення" />
 <Button
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:onClick = "sendMessage"
 android:text="Надіслати" />
</LinearLayout>
```

Значення "sendMessage", присвоєне атрибуту **`android:onClick`**, являє собою ім'я методу, визначеного в класі activity, який викликає система при натисканні користувачем на кнопку. Далі ми визначимо цей метод.

Створення об'єкта Intent

Потім перейдемо до класу MainActivity, котрий знаходится в проекті в папці *app/src/main/java/[назва_пакету]*, і змінимо цей клас наступним чином:

```
package com.example.ozm.helloworld;

import android.content.Intent; // підключаємо клас Intent
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View; // підключаємо клас View для обробки натискання кнопки
import android.widget.EditText; // підключаємо клас EditText

public class MainActivity extends AppCompatActivity {

 public final static String EXTRA_MESSAGE = "EXTRA_MESSAGE";
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 }

 // Метод обробки натискання на кнопку
 public void sendMessage(View view) {
 // дії, що здійснюються після натискання на кнопку
 // Створюємо об’єкт Intent для виклику нової Activity
 Intent intent = new Intent(this, DisplayMessageActivity.class);
 // Отримуємо текстове поле в поточній Activity
 EditText editText = (EditText) findViewById(R.id.edit_message);
 // Отримуємо текст даного текстового поля
 String message = editText.getText().toString();
```

```
// Додаємо за допомогою властивості putExtra об’єкт - перший параметр - ключ,
// другий параметр - значення цього об’єкту
intent.putExtra(EXTRA_MESSAGE, message);
// запуск activity
startActivity(intent);
}
}
```

Тут якраз і доданий метод **sendMessage ()**, який буде викликатися після натискання на кнопку. Оброблювач натискання кнопки повинен приймати як параметр об'єкт View, який представляє саму натиснуту кнопку:

```
public void sendMessage(View view) {
```

Далі для запуску другої activity необхідний об'єкт **Intent** . Об'єкт **Intent** представляє деяку задачу додатку, яку треба виконати (наприклад, запуск activity):

Intent intent = **new** Intent(**this**, DisplayMessageActivity.**class**);

Конструктор цього об'єкта приймає два параметри:

- Перший параметр представляє контекст - об'єкт `Context`(ключове слово `this` вживається тут, так як клас `MainActivity` є підкласом класу `Context`)
- Другим параметром йде клас компонента, якому ми передаємо об'єкт Intent. В якості нього буде виступати об'єкт DisplayMessageActivity, який ми створимо трохи пізніше

Усередині методу `sendMessage()` використовуємо метод **findViewById** , щоб отримати елемент `EditText` і передати значення його тексту в об'єкт `intent`:

```
EditText editText = (EditText) findViewById(R.id.edit_message);
String message = editText.getText().toString();
```

Потім отриманий з текстового поля текст передається в activity, що запускається:

```
intent.putExtra(EXTRA_MESSAGE, message);
```

Параметр "EXTRA_MESSAGE" вказує на ключ переданих даних. Тобто ми можемо в нову activity передати множину даних, і щоб їх можна було розмежувати, для них встановлюється ключ. В даному випадку ключ EXTRA_MESSAGE - це звичайний рядок "EXTRA_MESSAGE":

```
public final static String EXTRA_MESSAGE = "EXTRA_MESSAGE";
```

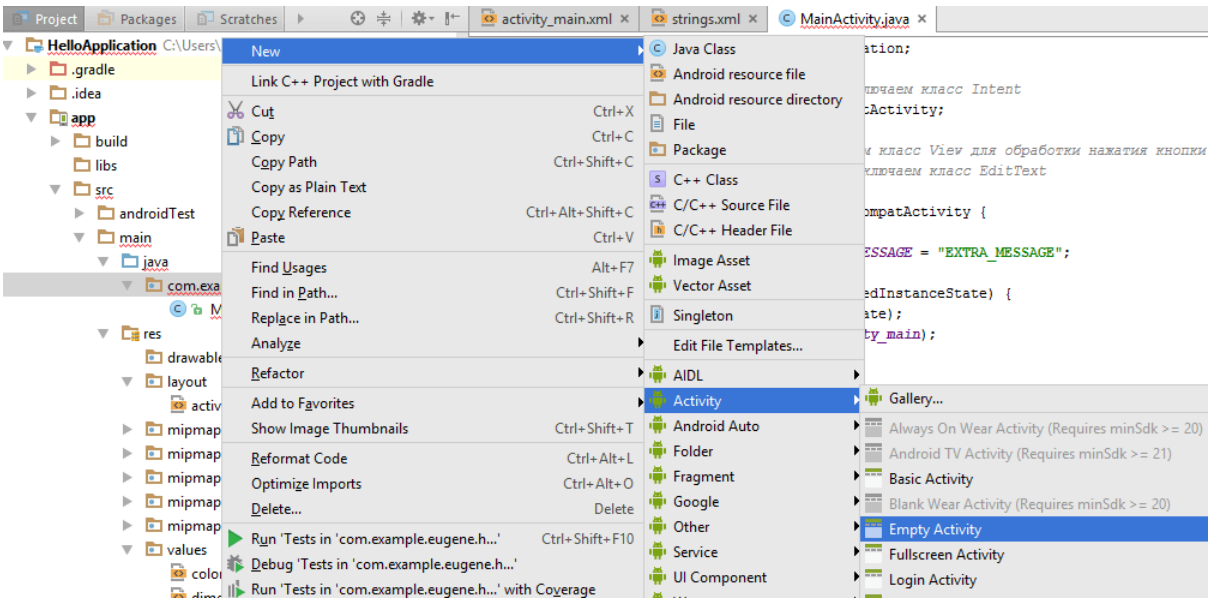
Для запуску activity потрібно викликати метод **startActivity ()** і передати йому в якості параметра об'єкт `Intent`:

```
startActivity(intent);
```

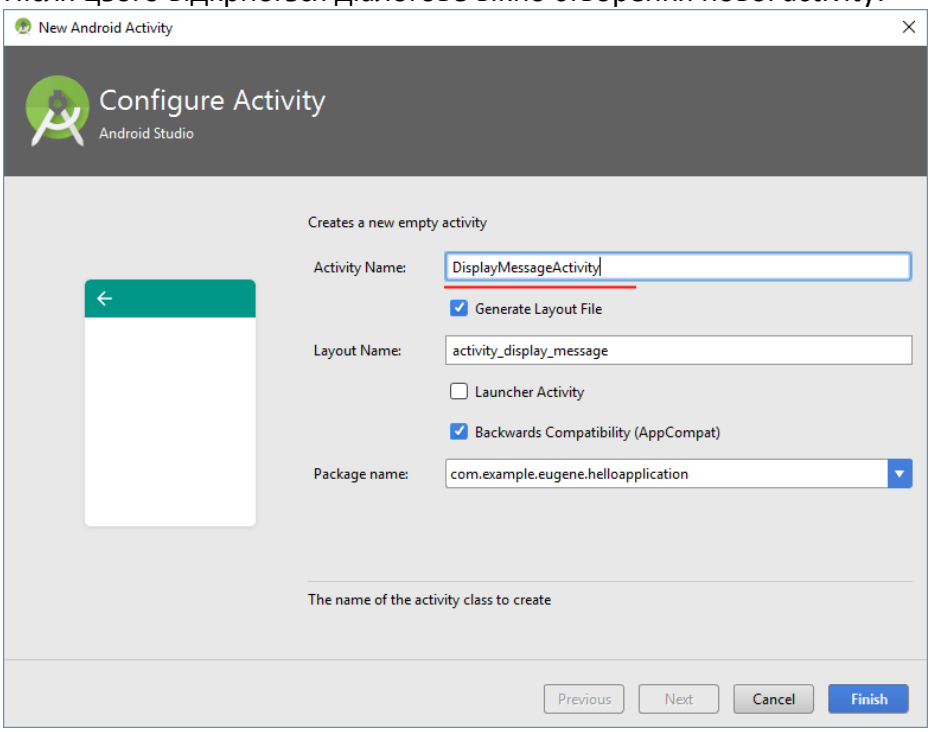
Після виклику цього методу система отримає сигнал і запустить новий об'єкт `Activity`, визначений об'єктом `Intent`.

Створення другого об'єкта Activity

Тепер додамо саму, нову activity - **DisplayMessageActivity**. Для цього натиснемо правою кнопкою миші в структурі проекту на папку пакета, в якому знаходиться клас MainActivity, і потім в контекстному меню виберемо **New-> Activity-> Empty Activity**:



Після цього відкриється діалогове вікно створення нової activity:

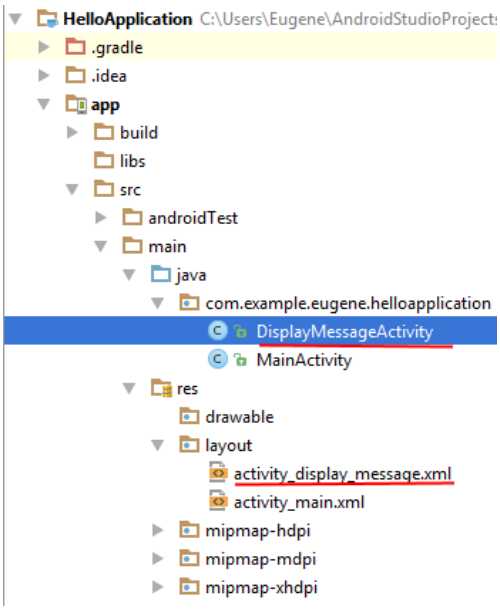


У цьому вікні введемо наступні дані:

- Activity Name: **DisplayMessageActivity**
- Layout Name: **activity_display_message**

І потім натиснемо Finish.

Отже, у нас була створена нова activity: в папку *app/src/main/java/[назва_пакету]* додається новий клас **DisplayMessageActivity**, а в каталог *app/src/main/res/layout* файл *activity_display_message.xml* з визначенням інтерфейсу для нової activity.



Відкриємо клас DisplayMessageActivity і змінимо його наступним чином:

```
package com.example.ozm.helloworld;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class DisplayMessageActivity extends AppCompatActivity {

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 // Отримуємо повідомлення з об’єкту intent
 Intent intent = getIntent();
 String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

 // Створюємо текстове поле
 TextView textView = new TextView(this);
 textView.setTextSize(40);
 textView.setText(message);

 // Встановлюємо текстове поле в системі компоновки activity
 setContentView(textView);
 }
}
```

Також, як і в MainActivity (і в інших activity), створення поточної activity відбувається в методі `onCreate()`. Всі підкласи Activity повинні реалізувати метод `onCreate`, так як система викликає його при створенні нової activity. Саме в цьому методі задається компоновка нового об'єкту activity за допомогою методу **setContentView** і саме тут відбувається початкова настройка компонентів.

Вивід повідомлення на екран

Отже, як ми пам'ятаємо, спочатку ми передавали з нашої першої і головної activity в DisplayMessageActivity текстове повідомлення, яке було введено в текстове поле. Тепер отримаємо його в DisplayMessageActivity і виведемо на екран

Кожен об'єкт `Activity` викликається об'єктом `Intent`. Ми можемо отримати викликаний об'єкт `Intent` за допомогою методу `getIntent` і, таким чином, отримати передані з ним дані.

```
Intent intent = getIntent();
String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
```

Для виведення повідомлення на екран створимо віджет `TextView` і за допомогою його властивості `setText` встановимо його текст. А потім додамо новий віджет в DisplayMessageActivity за допомогою методу **setContentView** .

```
TextView textView = new TextView(this);
textView.setTextSize(40);
textView.setText(message);
setContentView(textView);
```

Після запуску програми на емуляторі ми побачимо знову ж текстове поле з кнопкою, але після введення тексту і натиснення на кнопку буде запущена нова activity і відобразиться інший екран з введеним раніше в текстове поле повідомленням.