

ПРАКТИЧНА РОБОТА №4

Тема роботи: Робота із анімацією та макетами в ОС Android.

Мета роботи: Отримати досвід роботи із анімацією та ознайомитись із основними макетами для побудови користувацького інтерфейсу додатків під ОС Android.

Теоретичні відомості

У даній практичній роботі розглядаються особливості створення анімації та основних блоків для побудови користувацького інтерфейсу. Показано різні XML макети (компонування) в Android та практично реалізовано RelativeLayout і LinearLayout.

Створення власної реалізації класу Application дає можливість:

- контролювати стан додатку;
- передавати об'єкти між програмними компонентами;
- підтримувати і контролювати ресурси, які використовуються в декількох компонентах однієї програми.

При створенні операційною системою процесу, в якому буде виконуватися програма (з погляду Unix-подібних систем, процеси є контейнерами, в яких виконуються програми) створюється екземпляр класу Application, який описаний в маніфесті додатку. Таким чином, Application за своєю природою є сінглтоном (singleton) і повинен бути правильно реалізований, щоб надавати доступ до своїх методів і змінних.

Нижче показаний каркас для наслідування класу Application і використання його в якості сінглтона:

```
import android.app.Application;
import android.content.res.Configuration;
public class MyApplication extends Application {
    private static MyApplication singleton;
    // Повертає екземпляр данного класу
    public static MyApplication getInstance() {
        return singleton;
    }
    Override
    public final void onCreate() {
        super.onCreate();
        singleton = this; } }
```

Після створення реалізація класу Application повинна бути зареєстрована у маніфесті додатку, для цього використовується елемент <application>:

```
<application
    android:icon="@drawable/icon"
    android:name="MyApplication">
    [... інкапсульовані елементи ...]
```

</application>

Тепер при запуску програми буде створювати екземпляр реалізації класу Application. Якщо є необхідність зберігання стану програми і глобальних ресурсів, необхідно реалізувати відповідні методи у реалізації класу і використовувати їх в компонентах застосунку:

```
SomeObject value = MyApplication.getInstance().getGlobalStateValue();  
MyApplication.getInstance().setGlobalStateValue(someObjectValue);
```

Такий підхід може бути ефективний при передачі об'єктів між слабозв'язаними частинами програми і для контролю за загальними ресурсами і станом додатку.

Обробка подій життєвого циклу додатку

Аналогічно обробникам подій класу Activity, обробники подій класу Application так само можна перевизначати для управління реакцією додатка на ті чи інші події життєвого циклу:

```
import android.app.Application;  
import android.content.res.Configuration;  
public class MyApplication extends Application {  
    @Override  
    public void onConfigurationChanged(Configuration newConfig) {  
        super.onConfigurationChanged(newConfig);  
    }  
    @Override  
    public void onCreate() {  
        super.onCreate();  
    }  
    @Override  
    public void onLowMemory() {  
        super.onLowMemory();  
    }  
    @Override  
    public void onTerminate() {  
        super.onTerminate();  
    }  
}
```

Призначення цих методів наступне:

- **onCreate**: викликається при створенні програми, перевизначається для створення і ініціалізації властивостей тив яких зберігаються стан програми або глобальні ресурси.
- **onTerminate**: викликається при передчасному завершенні роботи програми (але може і не бути викликатись, якщо додаток закривається ядром, для звільнення ресурсів для інших програм.
- **onLowMemory**: надає можливість додаткам звільнити додаткову пам'ять (коли ОС не вистачає ресурсів). Цей метод перевизначається для того, щоб очистити кеш або звільнити непотрібні в даний момент ресурси.

- `onConfigurationChanged`: перевизначається, якщо необхідно відстежувати зміни конфігурації на рівні додатку (такі, наприклад, як поворот екрану, закриття висувної клавіатури пристрою).

Поняття контексту

Клас `android.context.Context` є інтерфейсом для доступу до глобальної інформації про додаток. Це абстрактний клас реалізація якого забезпечується системою Android. `Context` дозволяє отримати доступ до специфічних для даного додатку ресурсів і класів, а також для виклику операцій на рівні додатку, таких, як запуск активності, відправка повідомлень, отримання намірів (`Intent`) та інше.

Даний клас також є базовим для класів `Activity`, `Application` і `Service`. Отримати доступ контексту можна за допомогою методів `getApplicationContext`, `getContext`, `getBaseContext`, а також просто за допомогою властивості `this` (зсередини активності або сервісу). Одним із способів при виклику статичного методу `makeText` класу `Toast`, що отримує контекст в якості першого параметра:

```
Toast.makeText(this, "onCreate()", Toast.LENGTH_LONG).show();
```

Типове використання контексту може бути таким:

```
TextView myTextView = new TextView(getContext());
```

```
ListAdapter listAdapter = new
```

```
SimpleCursorAdapter(getApplicationContext(), ...);
```

Доступ до стандартних глобальних ресурсів:

```
context.getSystemService(LAYOUT_INFLATER_SERVICE);
```

```
prefs=getApplicationContext().getSharedPreferences("PREFS",MODE_PRIVATE);
```

Неявне використання глобальних компонентів системи:

```
cursor = getApplicationContext().getContentResolver().query(uri, ...);
```

Інтерфейс користувача. Основні поняття і зв'язки між ними

Клас `View` є базовим класом для всіх візуальних елементів UI (елементів управління (`Control`) і віджетів (`Widget`)). Всі ці елементи, в тому числі і розмітка (`Layout`), є розширеннями класу `View`.

Групи (`ViewGroup`) - нащадки класу `View`; можуть містити в собі кілька дочірніх `View`. Розширення класу `ViewGroup` використовується для створення складних `View`, що складаються з взаємопов'язаних компонентів. Клас `ViewGroup` також є базовим для різних розміток (`Layout`).

Активності (`Activity`) – відображають екрани або вікна (з точки зору побудови UI), є «андроїдними еквівалентами» форм. Для відображення UI активності використовують `View`.

Для створення додатків з унікальними інтерфейсом розробнику іноді доводиться розширювати і модифікувати стандартні `View`, комбінуючи їх зі стандартними.

Android надає розробнику можливість використання набору готових елементів користувацького інтерфейсу, що прописані у класі View:

- **TextView**. Стандартний елемент, призначений для виведення тексту. Підтримує багаторядкове відображення, форматування і автоматичний перенос.

- **EditText**. Редаговане поле для введення тексту. Підтримує багаторядкове введення, перенесення слів на новий рядок і текст підказки.

- **ListView**. Група уявлень (ViewGroup), яка формує вертикальний список елементів, відображаючи їх у вигляді рядків всередині списку. Найпростіший об'єкт ListView використовує TextView для виводу на екран значень toString (), що належать елементом масиву.

- **Spinner**. Складовий елемент, що відображає TextView у поєднанні з відповідним ListView, який дозволяє вибрати елемент списку для відображення в текстовому рядку. Сам рядок складається з об'єкта TextView і кнопки при натисканні на яку починається діалог вибору. Зовні цей елемент нагадує тег <SELECT> в HTML.

- **Button**. Стандартна кнопка, яку можна натискати.

- **CheckBox**. Кнопка, що має два стани. Представлена у вигляді прапорця.

- **RadioButton**. «Радіокнопка», яка дозволяє вибрати тільки один з декількох варіантів.

- **ViewFlipper**. Група уявлень (ViewGroup), що дозволяє визначити набір елементів і горизонтальний рядок, в якому може виводитися тільки одне View. При цьому переходи між елементами, які відображаються здійснюються за допомогою анімації.

Android пропонує і інші реалізації View, такі, як елементи для вибору дати і часу, поля введення з автоматичним доповненням, галереї, вкладки і навіть карти (MapView).

Більш повний список підтримуваних системою View можна побачити за адресою <http://developer.android.com/guide/tutorials/views/index.html>

Крім готових View, розробник, при необхідності, може створювати власні, розширюючи клас View або його підкласи.

Розмітки в Android

Розмітка (Layout) є розширенням класу ViewGroup і використовується для розміщення дочірніх компонентів на екрані пристрою. Використовуючи вкладені розмітки, можна створювати користувацькі інтерфейси будь-якої складності.

Найбільш часто використовувані види розмітки:

- **FrameLayout**. Найпростіша розмітка, прикріплює кожне нове дочірнє подання до лівого верхнього кута екрану, накладаючи новий елемент на попередній та затуляє його.

- **LinearLayout**. Поміщає дочірні View в горизонтальний або вертикальний ряд. Вертикальна розмітка являє собою колонку, а

горизонтальна - рядок з елементами. Дана розмітка дозволяє задавати не тільки розміри, але і «відносну вагу» дочірніх елементів, завдяки чому можна гнучко контролювати їх розміщення на екрані.

- **RelativeLayout.** Найбільш гнучкий серед стандартних видів розмітки. Дозволяє вказувати позиції дочірніх View щодо меж вільного простору та інших View.

- **TableLayout.** Дозволяє розміщувати дочірні View всередині комірок «сітки», що складається з рядків і стовпців. Розміри комірок можуть залишатися постійними або автоматично розтягуватися при необхідності.

- **Gallery.** Представляє елементи у вигляді прокручуваного горизонтального списку (зазвичай графічні елементи).

Актуальну інформацію про властивості і можливості різних видів розмітки можна отримати за адресою:

<http://developer.android.com/guide/topics/ui/layout-objects.html>

Найбільш поширений спосіб реалізації розмітки екрану - використання зовнішніх ресурсів: XML-файлів, що описують розміщення елементів на екрані і їх параметри.

В попередніх практичних роботах розглядався вміст файла `res / layout / main.xml` для зміни зовнішнього вигляду додатку HelloAndroidWorld, а тепер докладніше розглянемо його вміст:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        android:textColor="@color/text_color" />
</LinearLayout>
```

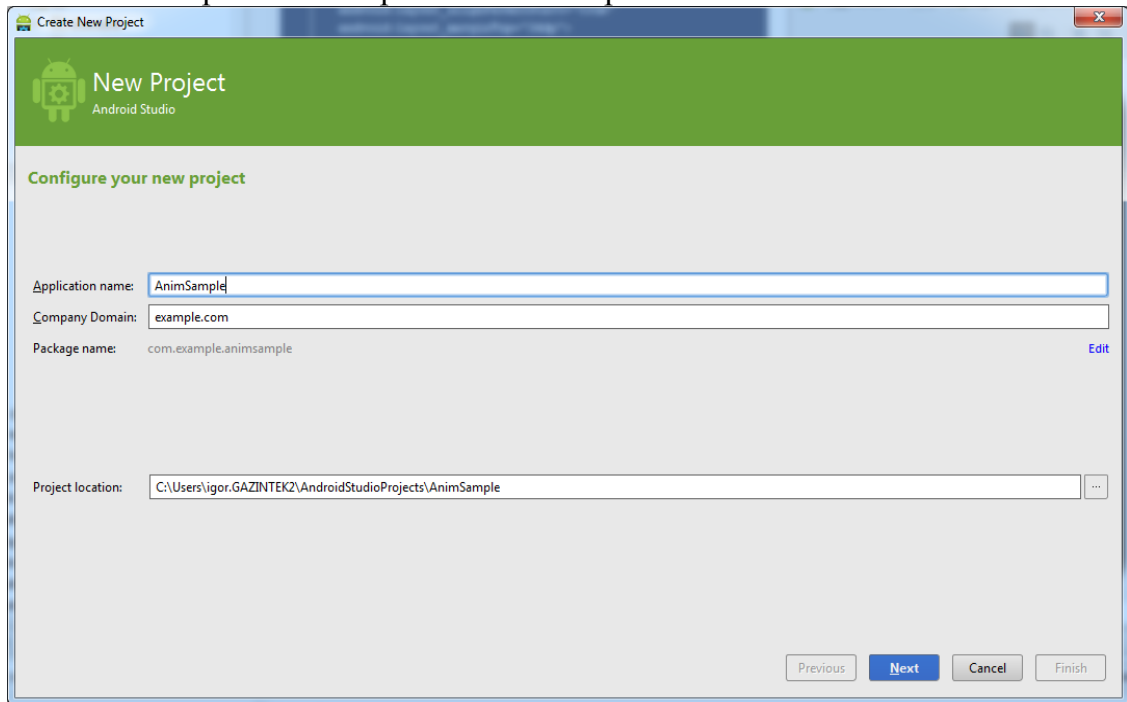
Кореневим елементом цієї розмітки є `<LinearLayout>`, що має один дочірній елемент `<TextView>`. Атрибути `<LinearLayout>` визначають простір імен «android» (`xmlns:android="http://schemas.android.com/apk/res/android"`), ширину (`android:layout_width`), висоту (`android:layout_height`) і орієнтацію (`android:orientation`), яка визначає спосіб розміщення дочірніх елементів всередині `LinearLayout`: вертикально чи горизонтально. Зверніть увагу на відносне (щодо розмірів батьківського елемента), а не абсолютне (в пікселях) задання розмірів (ширини і висоти). Такий спосіб визначення розмірів є кращим і дозволяє створювати дизайн додатків, що не прив'язаний до розмірів екрану пристрою.

Елемент <TextView> в нашому випадку має наступні атрибути: ширину і висоту, а також посилання на рядок з ім'ям «hello» і колір, що використовується для відображення тексту «text_color».

Завдання для виконання

Завдання 1 «Робота із анімацією»

1. Створіть новий проект AnimSample.



2. В каталозі res створіть каталог anim, а в ньому файл з ім'ям ship_anim.xml, що описує анімацію. Відредагуйте файл, щоб він мав наступний вміст:

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" >
<rotate
    android:duration="3333"
    android:fromDegrees="0"
    android:pivotX="50%"
    android:pivotY="50%"
    android:repeatCount="infinite"
    android:repeatMode="reverse"
    android:toDegrees="1080" />
<translate
    android:duration="1900"
    android:fromXDelta="-50%p"
    android:repeatCount="infinite"
    android:repeatMode="reverse"
```

```

        android:toXDelta="50%p" />
<translate
    android:duration="1300"
    android:fromYDelta="-50%p"
    android:repeatCount="infinite"
    android:repeatMode="reverse"
    android:startOffset="123"
    android:toYDelta="50%p" />
<alpha
    android:duration="500"
    android:fromAlpha="1.0"
    android:repeatCount="infinite"
    android:repeatMode="reverse"
    android:toAlpha="0.3" />
<scale
    android:duration="10000"
    android:fromXScale="0.0"
    android:fromYScale="0.0"
    android:pivotX="50%"
    android:pivotY="50%"
    android:repeatCount="infinite"
    android:repeatMode="reverse"
    android:toXScale="2.5"
    android:toYScale="2.5" />
</set>

```

3. Додайте малюнок, до якого буде застосовуватися анімація (файл `lander_plain.png`), в каталог `res / drawable-mdpi`.

4. У файлі розмітки `res / layout / main.xml` замініть елемент `TextView` на `ImageView` з наступними атрибутами:

```

<ImageView
    android:id="@+id/shipView"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:src="@drawable/lander_plain" />

```

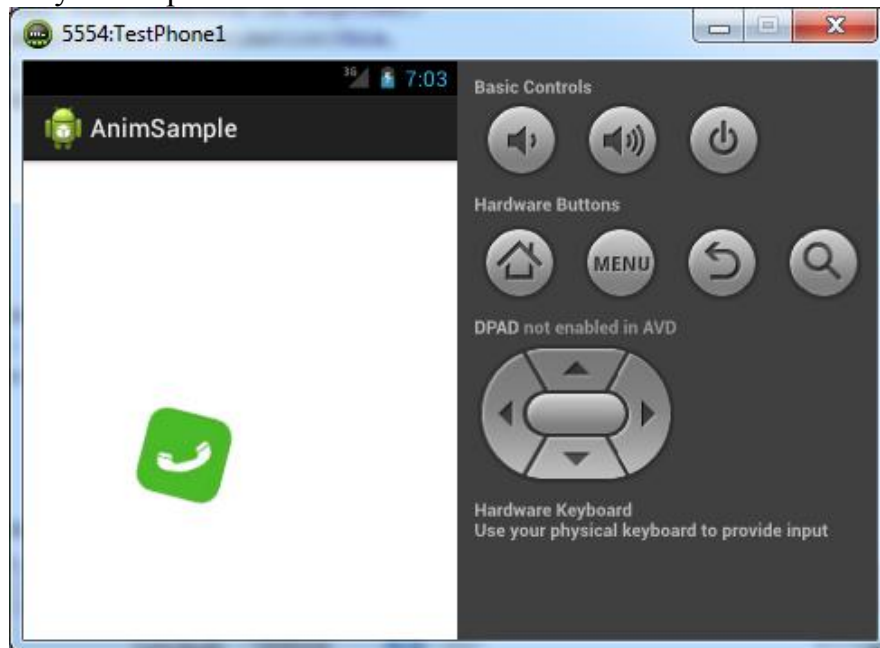
5. В кінці методу `onCreate` Активності (вона в цьому проекті одна) додайте наступні рядки:

```

ImageView ship = (ImageView) findViewById(R.id.shipView);
Animation shipAnim = AnimationUtils.loadAnimation(this,
    R.anim.ship_anim);
ship.startAnimation(shipAnim);

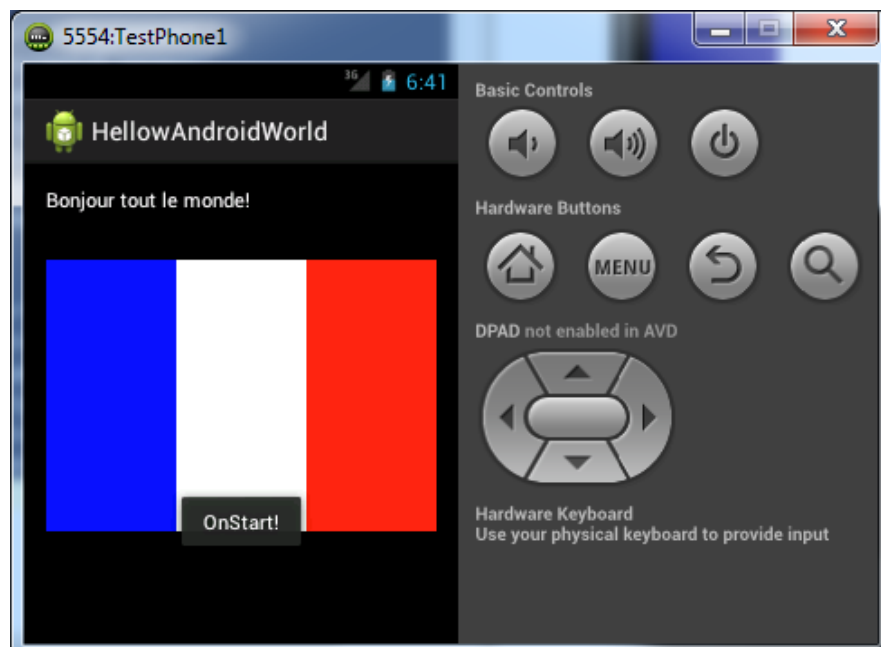
```

6. Запустіть проект



Завдання 2 «Використання LinearLayout»

1. Продовжіть локалізацію додатку Hello Android World, тепер для французької мови. Для малювання прапора використовуйте вкладений LinearLayout з вертикальною орієнтацією дочірніх елементів:



2. Зверніть увагу, що при розміщенні елементів всередині вкладеного `LinearLayout` зручно вказувати атрибут `android:layout_weight = "1"`, в цьому випадку дочірні віджети будуть розміщені по горизонталі рівномірно.

3. Після отримання потрібного результату поверніть стандартні мовні налаштування у віртуальному пристрої.

Завдання 3 «Використання `RelativeLayout`»

`RelativeLayout` є дуже корисним варіантом розмітки, що дозволяє створювати користувацький інтерфейс без надмірної кількості вкладених елементів.

1. Створіть новий проект з ім'ям `RelativeLayoutSample`.

2. Додайте потрібні рядки в файл `res / values / strings.xml` і видаліть рядок з ім'ям `hello`:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="label_text">Введіть текст:</string>
    <string name="entry_hint">Поле введення</string>
    <string name="app_name">RelativeLayoutSample</string>
</resources>
```

3. Відредагуйте файл розмітки `res / layout / main.xml` так, щоб він мав наступний вміст:

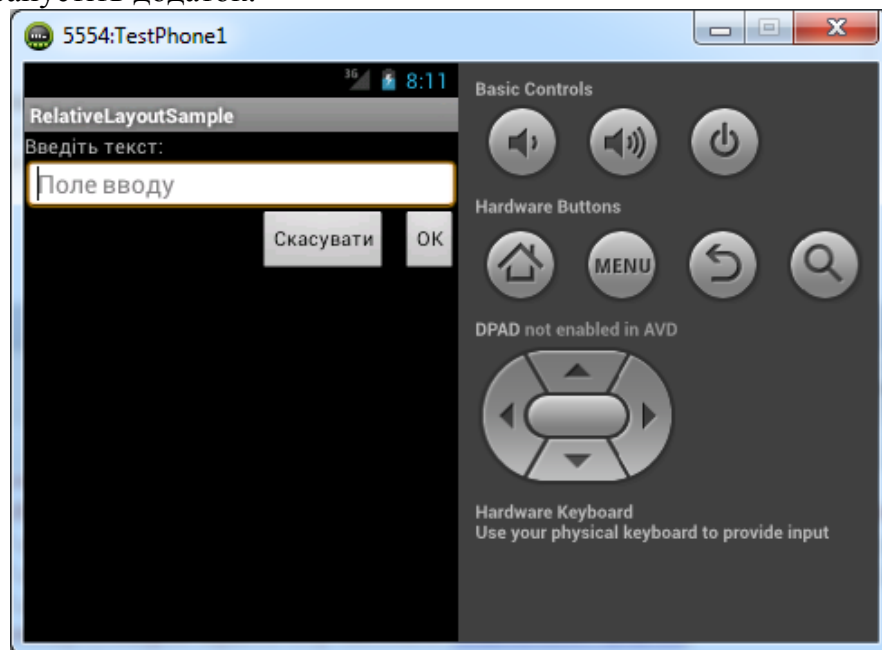
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TextView
        android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/label_text" />
    <EditText
        android:id="@+id/entry"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/label"
        android:background="@android:drawable/editbox_background"
        android:hint="@string/entry_hint" />
    <Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
```

```

        android:layout_below="@id/entry"
        android:layout_marginLeft="10dip"
        android:text="@android:string/ok" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/ok"
        android:layout_toLeftOf="@id/ok"
        android:text="@android:string/cancel" />
</RelativeLayout>

```

4. Запустіть додаток:



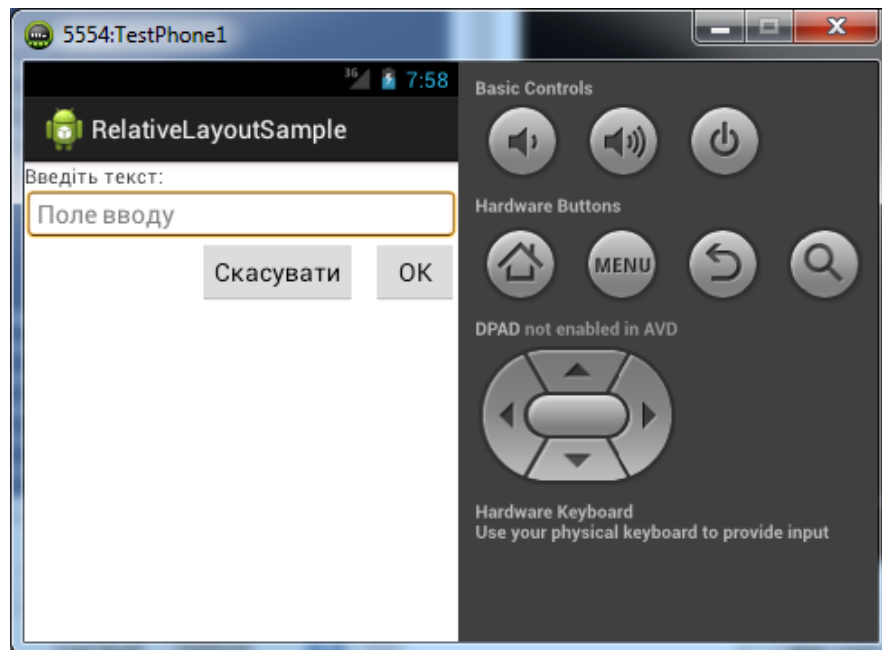
5. Відредагуйте файл AndroidManifest.xml, щоб змінити тему, використовувану додатком. Для цього у вузол <application> внесіть такі зміни:

```

<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Light" >

```

6. Запустіть додаток з новою темою:



7. Поекспериментуйте з налаштуваннями мови у віртуальному пристрої і зверніть увагу, як при запуску програми змінюються написи на кнопках. Очевидно, що використання стандартних строкових значень дозволяє мінімізувати витрати на локалізацію додатків.

Завдання для виконання

Виконайте завдань до виконання у відповідності до теоретичних відомостей поданих вище.

Вимоги до звіту

В звіт по роботі включіть текст програм, що реалізують описані завдання та екранні форми, що відображають їх виконання.

Контрольні запитання

1. Наскільки добре платформа Android справляється з анімацією?
2. Чи існує спосіб зупинки відтворення анімації?
3. Які типи операцій підтримуються анімаціями руху?
4. Чи може бути використаний елемент-контейнер LinearLayout для розміщення всіх дочірніх елементів-уявлень View один за одним (по вертикалі)?
5. Назвіть віджети, які використовуються для створення користувацького інтерфейсу.