Лабораторна робота №5. Робота з об’єктами intent та bundle

Мета: ознайомитись з використанням об’єктів intent та bundle у життєвому циклі activity.

Теоретичні відомості

Створення layout для landscape\portrait режимів.

Створення додатків передбачає їх використання на пристроях з різними характеристиками розмірів та роздільної здатності екранів. Для коректної роботи додатку на різних пристроях необхідно створювати графічні інтерфейси з найбільш загальними налаштуваннями та декларувати XML представлення для різних екранів.

Для створення представлення для **landscape** орієнтації потрібно створити у папці **res** нову папку **Android resource directory** та обрати кваліфікатор **Orientation** (рис. 1). Після чого, в полі Chosen qualifiers обрати необхідну орієнтацію екрану (рис. 2).

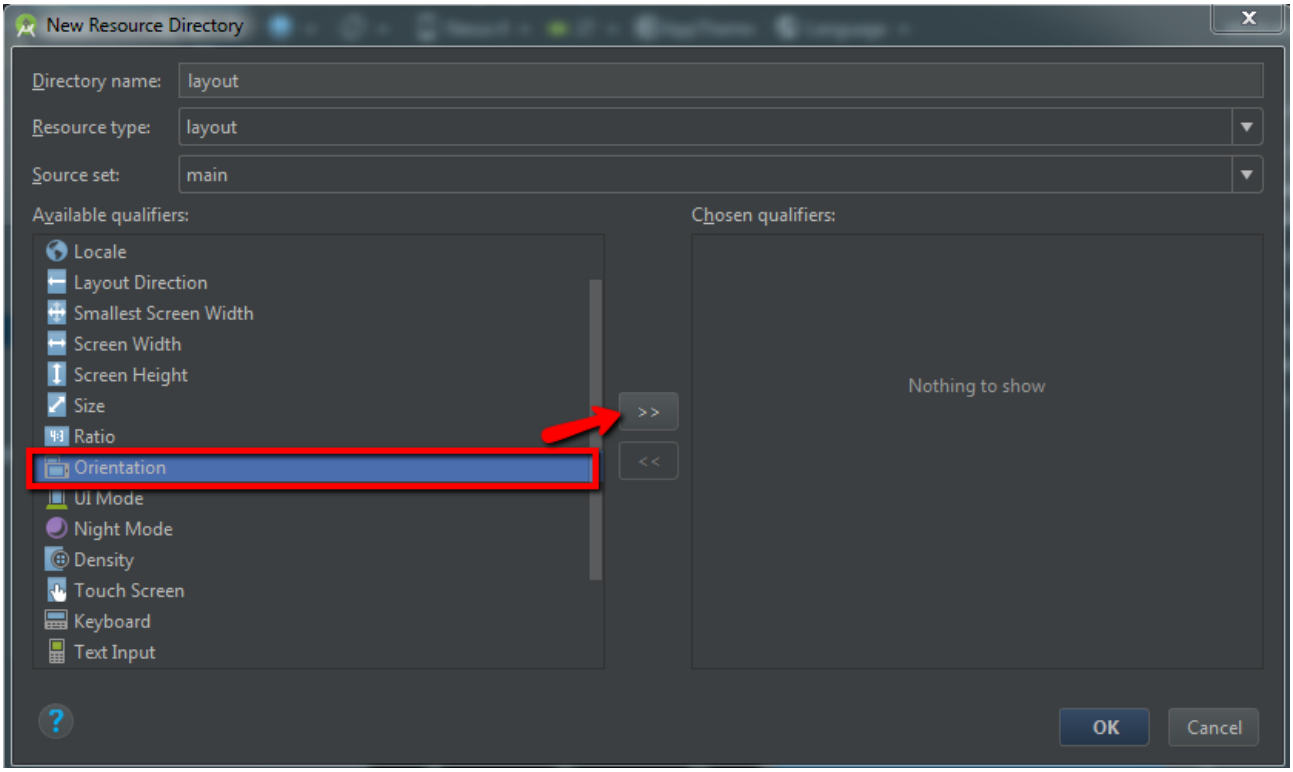

Рис. 1

Для того, щоб побачити створену папку, потрібно переключитись в режим відображення структури **Project** (рис. 2).

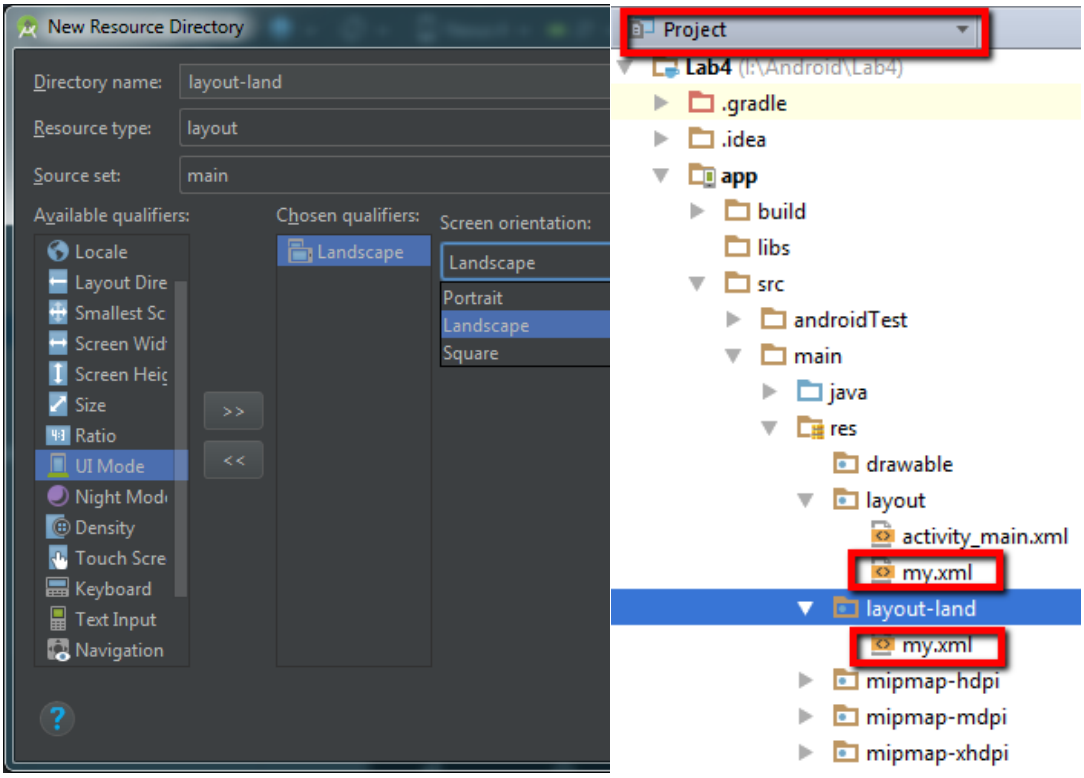

Рис. 2

Activity Stack. Для кожного додатку на Android система підтримує стек activity. Після запуску додатку, перша з activity заноситься на вершину стеку. Після запуску другої активності, вона поміщається на вершину стека і т.д. При натисканні кнопки «Назад» попередня activity з стеку відновлюється (рис. 3).

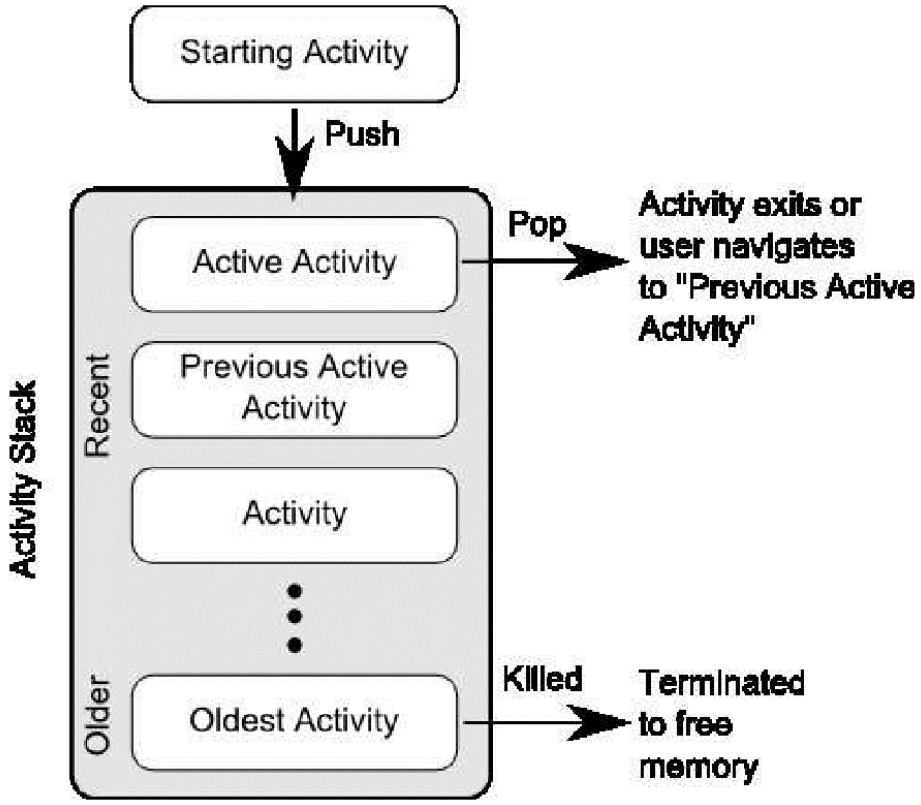

Рис. 3

Збереження activity. Активність може завершити свою роботу планово – в результаті виклику метода **finish()**, або система може знищити activity, якщо вона зупинена і не використовується довгий час, або якщо активності потрібно ресурсів більше ніж наявно в системі в даний момент.

У разі, якщо Android примусово завершує роботу додатку, система запам’ятовує стан активності та при відновленні додатка, відновлює збережений стан.

Системні дані, що використовуються системою для відновлення попереднього стану зберігаються у вигляді набору пар «ключ-значення» та містяться у об’єкті **Bundle.**

Наприклад, activity знищується та відновлюється системою при повороті екрана. Це пов’язано з потенційною необхідністю завантажити інше представлення та пов’язані ресурси при зміні орієнтації екрана. Аналогічна ситуація виникає при будь-якій зміні конфігурації пристрою або зовнішніх перериваннях.

Для збереження додаткових параметрів стану активності необхідно перевизначити метод зворотного виклику **onSaveInstanceState().** Система викликає цей метод коли завершується виконання активності та передає йому об’єкт **Bundle** у якому збережено необхідні параметри. При відновленні activity той же самий **Bundle** передається методам **onRestoreInstanceState()** та **onCreate()** (рис. 4).

Рис. 4

Для збереження стану активності необхідно перевизначити метод **onSaveInstanceState()** та додати до об’єкта **Bundle** необхідні пари «ключ-значення» (рис. 5).

```
static final String STATE_SCORE = "playerScore";
static final String STATE_LEVEL = "playerLevel";
...

@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    // Save the user's current game state
    savedInstanceState.putInt(STATE_SCORE, mCurrentScore);
    savedInstanceState.putInt(STATE_LEVEL, mCurrentLevel);

    // Always call the superclass so it can save the view hierarchy state
    super.onSaveInstanceState(savedInstanceState);
}
```

Рис. 5

Відновлення activity. При відновленні стану активності, попередній стан можна відновити з об'єкту **Bundle**, куди система передає дані activity. Методи **onRestoreInstanceState()** та **onCreate()** отримують той самий **Bundle**.

Оскільки метод onCreate() викликається при створенні нового екземпляру activity або при його відновленні, спочатку потрібно перевірити чи порожній **Bundle** (рис. 6).

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState); // Always call the superclass first

    // Check whether we're recreating a previously destroyed instance
    if (savedInstanceState != null) {
        // Restore value of members from saved state
        mCurrentScore = savedInstanceState.getInt(STATE_SCORE);
        mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);
    } else {
        // Probably initialize members with default values for a new instance
    }
    ...
}
```

Рис. 6

Альтернативним способом є перевизначення метода **onRestoreInstanceState()**, який викликається системою після метода **onStart()**. При цьому, система викликає **onRestoreInstanceState()** тільки при наявності збереженого стану. Тобто, відпадає необхідність перевірки чи порожній **Bundle** (рис. 7).

```
public void onRestoreInstanceState(Bundle savedInstanceState) {
    // Always call the superclass so it can restore the view hierarchy
    super.onRestoreInstanceState(savedInstanceState);

    // Restore state members from saved instance
    mCurrentScore = savedInstanceState.getInt(STATE_SCORE);
    mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);
}
```

Рис. 7

Виклик іншої activity. В додатках з декількома активностями виклик однієї activity з іншої виконується за допомогою метода **startActivity**:

startActivity(intent);

де **intent –** це об’єкт класа **android.content.Intent**.

Припустимо, що треба написати додаток, у якому з головної активності запускається друга активність при натисканні кнопки та відображається відповідне повідомлення у віджеті TextView другої активності.

Файл **AndroidManifest.xml** у випадку використання в проекті двох activity може мати наступний вигляд:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.alex.lab3">

 <application
 android:allowBackup="true"
 android:icon="@mipmap/ic_launcher"
 android:label="@string/app_name"
 android:supportsRtl="true"
 android:theme="@style/AppTheme">
 <activity android:name=".MainActivity">
 <intent-filter>
 <action android:name="android.intent.action.MAIN" />

 <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter>
 </activity>
 <activity android:name=".secondActivity"></activity>
 </application>

</manifest>
```

Для обробки події натискання кнопки в класі MainActivity експортовано клас **OnTouchListener** та перевизначено метод **onTouch**. В цьому методі створюється об’єкт класа **Intent** та поміщається повідомлення методом

intent.putExtra. Тепер можна використовувати метод **startActivity** для запуску іншої activity

В методі **onCreate** другої activity викликається метод **getIntent** який отримує повідомлення від метода **getStringExtra**. Яке потім можна використовувати у методі **setText** віджета TextView. Це можна зробити за допомогою наступної команди:

((TextView)findViewById(R.id.textView1)).setText(message)

Об'єкти Intent, пов'язані з Activity. Таким чином:

```
Intent  intent = new Intent(this, SecondActivity.class);
startActivity(intent);
```

Якщо треба передати …

```
Intent intent = new Intent(this, SecondActivity.class);
intent.putExtra("message", "Message from First Screen");
startActivity(intent);
```

Intent представляє собою об'єкт обміну повідомленнями, за допомогою якого можна видати запит на виконання дії у іншого компонента або компонента іншого додатку.

Об'єкти **Intent** зазвичай використовуються у наступних випадках:

- Для виклику activity. Використовуються методи startActivity() або startActivityForResult()/onActivityResult().
- Для виклику служб (Service). Використовуються методи startService() та bindService().
- Для розсилки широкомовних повідомлень. Використовуються методи sendBroadcast(), sendOrderedBroadcast() или sendStickyBroadcast().

Є два типа об'єктів **Intent**:

Явні об'єкти **Intent** вказують компонент, який потрібно запустити по імені класа. У попередньому прикладі використовувався явний Intent, тобто такий, у якому явно прописана activity, якій він спрямований.

Неявні об’єкти **Intent** не містять імені конкретного компонента. Замість цього, вони визначають дію, яку потрібно виконати. Система сама вирішує, для яких компонентів спрямовано Intent шляхом порівняння змісту об’єкта Intent з фільтрами Intent, які визначені в файлах маніфеста інших додатків. Якщо об’єкт Intent співпадає з фільтром Intent, система запускає цей компонент і передає йому об’єкт Intent.

Фільтр Intent представляє собою вираз в файлі маніфеста додатку, який вказує типи об’єктів Intent, які може прийняти компонент.

Таблиця 1 містить назви можливих дій, які можуть вказуватись в об’єкті Intent спрямованому activity.

Дія	Опис
ACTION_MAIN	Запускає activity та визначає її як точку входу в додаток.
ACTION_VIEW	Переглядає дані, додані до Intent
ACTION_ATTACH_DATA	Додає дані, які були додані до Intent у іншому компоненті.
ACTION_EDIT	Редагує дані додані до Intent.
ACTION_PICK	Вибрати пункт з даних.
ACTION_CHOOSER	Показати всі додатки, які реагують на Intent.
ACTION_GET_CONTENT	Дозволяє користувачу вибрати певні дані та передати їх.
ACTION_DIAL	Набрати номер, доданий до Intent.
ACTION_CALL	Зателефонувати персоні, вказаній у Intent.
ACTION_SEND	Відіслати дані, додані до Intent.
ACTION_SENDTO	Відіслати повідомлення персоні, вказаній у Intent.
ACTION_ANSWER	Відповісти на вхідний дзвінок.
ACTION_INSERT	Вставити пустий об’єкт у контейнер.
ACTION_DELETE	Видалити вказані дані з контейнера.
ACTION_RUN	Запустити додані дані.
ACTION_SYNC	Виконати синхронізацію даних.
ACTION_PICK_ACTIVITY	Вибрати activity зі списку activity.
ACTION_SEARCH	Виконати пошук, використовуючи строки пошуку з Intent.
ACTION_WEB_SEARCH	Виконати пошук у WEB, використовуючи строки пошуку з Intent.
ACTION_FACTORY_TEST	Визначити як точку входу для factory test.

Створення неявного об’єкта Intent. При визначенні неявного об'єкта Intent описується специфічна дія (перегляд, відсилання, дзвінок), для якої потрібно знайти виконавця (відповідний додаток). Також часто разом з Intent’ом передаються певні дані, які можуть бути об’єктами класа **Uri,** строками і т.д. Наприклад, для виклику телефонного дзвінка, може використовуватись Intent з даними типу Uri.

```
Uri number = Uri.parse("tel:5551234");
Intent callIntent = new Intent(Intent.ACTION_DIAL, number);
```

Рис. 8

Перегляд карти.

```
// Map point based on address
Uri location = Uri.parse("geo:0,0?q=1600+Amphitheatre+Parkway,+Mountain+View,+California");
// Or map point based on latitude/longitude
// Uri location = Uri.parse("geo:37.422219,-122.08364?z=14"); // z param is zoom level
Intent mapIntent = new Intent(Intent.ACTION_VIEW, location);
```

Рис. 9

Перегляд веб-сторінки.

```
Uri webpage = Uri.parse("http://www.android.com");
Intent webIntent = new Intent(Intent.ACTION_VIEW, webpage);
```

Рис. 10

Перевірка існування activity \ додатка для обробки Intent`a**. Для того, щоб переконатися в тому, що існує активність для обробки Intent`a, використовується метод **queryIntentActivities() для отримання списку таких активностей (рис. 11).

```
PackageManager packageManager = getPackageManager();
List activities = packageManager.queryIntentActivities(intent,
        PackageManager.MATCH_DEFAULT_ONLY);
boolean isIntentSafe = activities.size() > 0;
```

Рис. 11

Якщо значення **isIntentSafe** має значення true, то хоча б один додаток має зреагувати на Intent. Наприклад, перегляду карти, може бути застосований наступний код.

```
// Build the intent
Uri location = Uri.parse("geo:0,0?q=1600+Amphitheatre+Parkway,+Mountain+View,+California");
Intent mapIntent = new Intent(Intent.ACTION_VIEW, location);

// Verify it resolves
PackageManager packageManager = getPackageManager();
List<ResolveInfo> activities = packageManager.queryIntentActivities(mapIntent, 0);
boolean isIntentSafe = activities.size() > 0;

// Start an activity if it's safe
if (isIntentSafe) {
    startActivity(mapIntent);
}
```

Рис. 12

Виклик activity з об'єктом Intent. Після створення об'єкту Intent і додавання до нього потрібних даних, викликається метод **startActivity()** для передачі Intent'у системі Android. Якщо система ідентифікує декілька activity для обробки Intent'у, система запускає вікно вибору потрібного додатка (рис. 13).

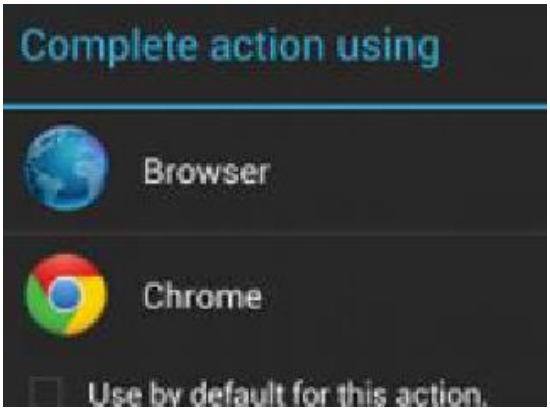

Рис. 13

Для того, щоб відобразити блок вибору додатку, використовується метод **createChooser()**.

```
Intent intent = new Intent(Intent.ACTION_SEND);
...

// Always use string resources for UI text.
// This says something like "Share this photo with"
String title = getResources().getString(R.string.chooser_title);
// Create intent to show chooser
Intent chooser = Intent.createChooser(intent, title);

// Verify the intent will resolve to at least one activity
if (intent.resolveActivity(getPackageManager()) != null) {
    startActivity(chooser);
}
```

Рис. 14

Отримання результату від activity. Для отримання результату від запущеного activity використовується метод **startActivityForResult()** замість методу **startActivity()**. При поверненні результату, в запущеному activity використовується об'єкт Intent, а в початковому activity метод **onActivityResult()**. При цьому, в методах **startActivityForResult()** та **onActivityResult()** потрібно використовувати один числовий ідентифікатор – код запиту (рис. 15).

```
static final int PICK_CONTACT_REQUEST = 1;  // The request code
...
private void pickContact() {
    Intent pickContactIntent = new Intent(Intent.ACTION_PICK, Uri.parse("content://contacts"));
    pickContactIntent.setType(Phone.CONTENT_TYPE); // Show user only contacts w/ phone numbers
    startActivityForResult(pickContactIntent, PICK_CONTACT_REQUEST);
}
```

Рис. 15

Метод **onActivityResult()** має наступні аргументи (рис. 16):

- Код запиту (аналогічний коду в методі startActivityForResult()).
- Код, вказаний другою операцією (RESULT_OK, RESULT_CANCELED).
- Intent, який передає необхідні дані відповіді.

```java
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // Check which request we're responding to
    if (requestCode == PICK_CONTACT_REQUEST) {
        // Make sure the request was successful
        if (resultCode == RESULT_OK) {
            // The user picked a contact.
            // The Intent's data Uri identifies which contact was selected.

            // Do something with the contact here (bigger example below)
        }
    }
}
```

Рис. 16

Запуск власного activity з інших додатків. Для можливості запуску activity сторонніми додатками або системою необхідно додати в файл маніфесту елемент **<intent-filter>** для відповідного activity.

В **<intent-filter>** потрібно якомога точніше вказати дію (**<action>**), тип даних (**<data>**) та категорію (**<category>**).

Більш докладно з можливими значеннями цих параметрів та прикладами можна ознайомитися за посиланням:

https://developer.android.com/reference/android/content/Intent.html

Завдання до лабораторної роботи

1. Додати до застосунку логічної гри з лабораторної роботи №4 дві додаткові activity. Таким чином, в застосунку три activity: перша – activity реєстрації нового користувача, друга – вікно з налаштуваннями та безпосередньо грою, третя – activity з результатами гри. Організувати передачу даних між екранами за допомогою intent.
2. Спроектувати та реалізувати екрани для landscape та portrait орієнтації. Реалізувати збереження обраних налаштувань за допомогою bundle.
3. Реалізувати можливість відсилання результатів гри на e-mail, вказаний при реєстрації.

Контрольні запитання

1. Для чого використовуються об’єкти intent? Їх типи.
2. Призначення об’єктів bundle.
3. Яким чином реалізуються різні екрани для різних орієнтацій?