Програмування мобільних пристроїв

Передача даних між активностями

Слайди до лекцій (5 змістовий модуль)

Створення додаткової активності

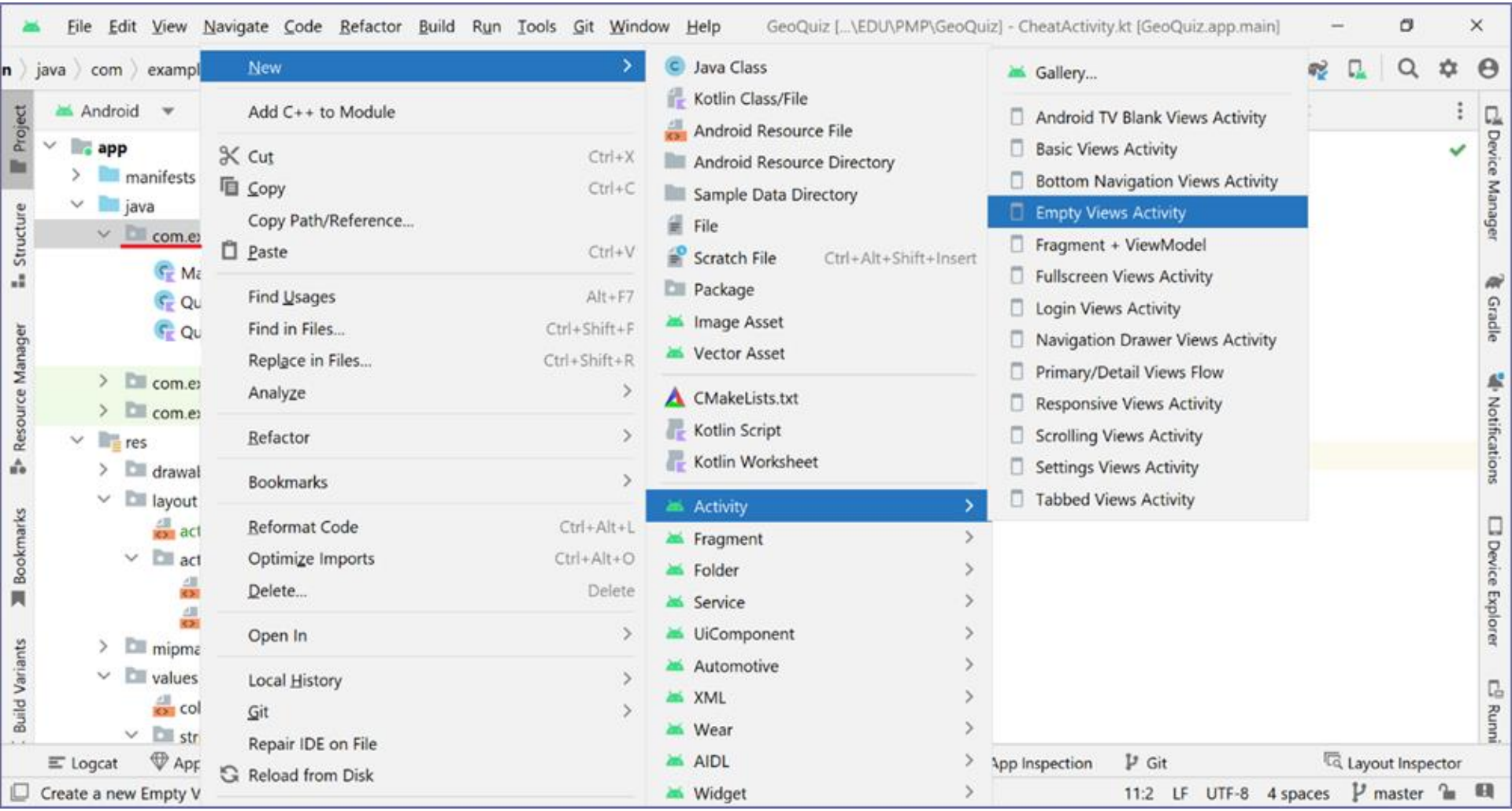
- Android-застосунки, як правило, мають більше однієї активності: внаслідок даних, що вводяться до основної активності відкривається та або інша додаткова активність, застосунки мають пов'язані з активностями екрани конфігурації, виведення звітів, покрокової роботи тощо.
- При наявності декількох активностей, вони, зазвичай, взаємодіють, обмінюючись даними.
- Додаймо до застосунку GeoQuiz додаткову активність, що надасть другий екран, на якому користувачеві буде запропоновано побачити відповідь на поточне запитання.
- Якщо користувач вирішує підглянути відповідь, а потім повертається до MainActivity та відповідає на запитання, він отримує повідомлення, що підглядати недобре :)

Створення додаткової активності - рядкові ресурси

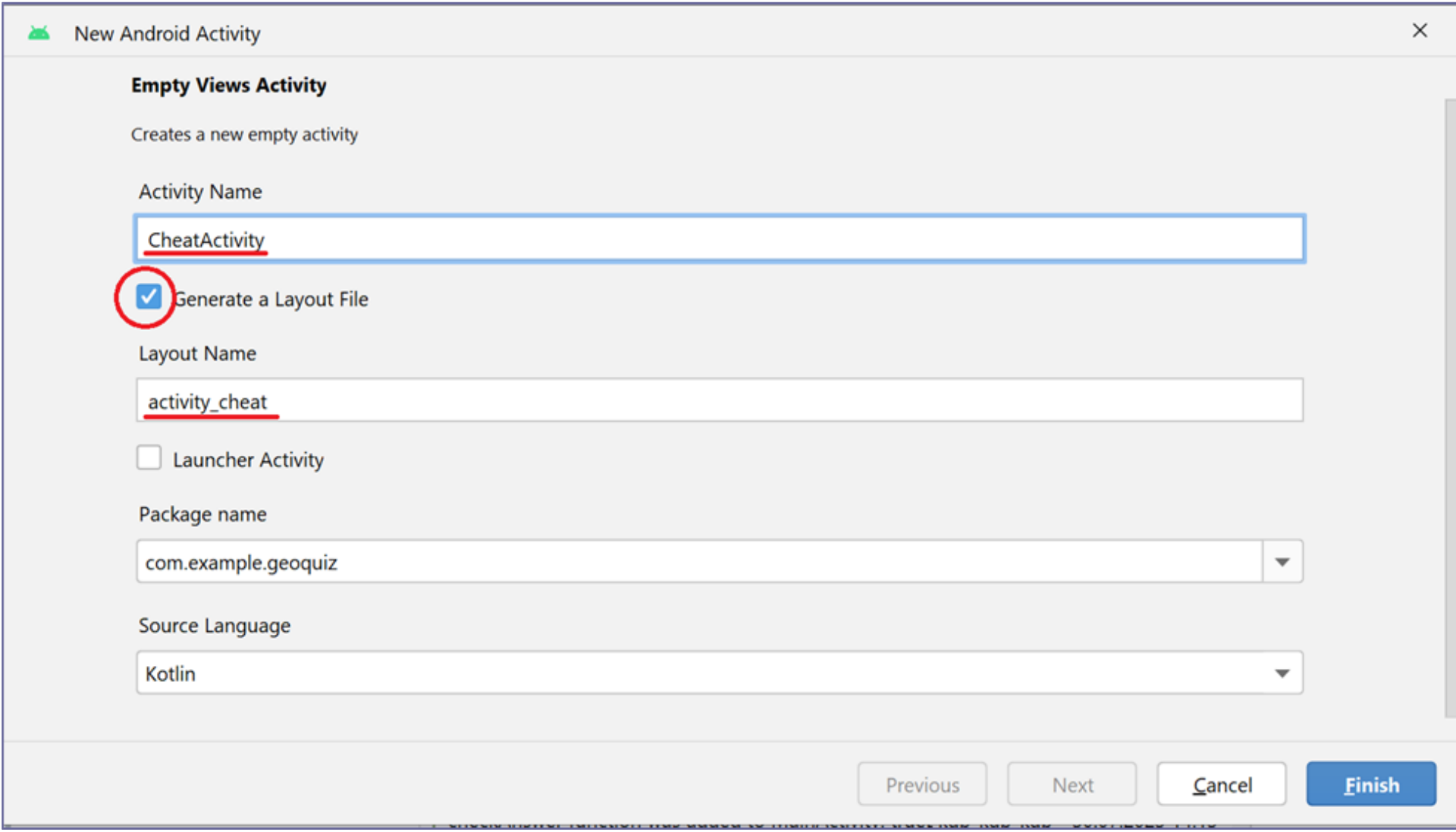
```
<resources>
    ...
    <string name="cheat_button">Cheat!</string>
    <string name="show_answer_button">Show answer</string>
    <string name="warning_text">Are you sure you want to do
this?</string>
    <string name="judgment_toast">Cheating is wrong</string>
</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools">
    ...
    <string name="cheat_button">Підглянути!</string>
    <string name="show_answer_button">Показати
відповідь</string>
    <string name="warning_text">Ви впевнені, що хочете зробити
це?</string>
    <string name="judgment_toast">Підглядати недобре</string>
</resources>
```

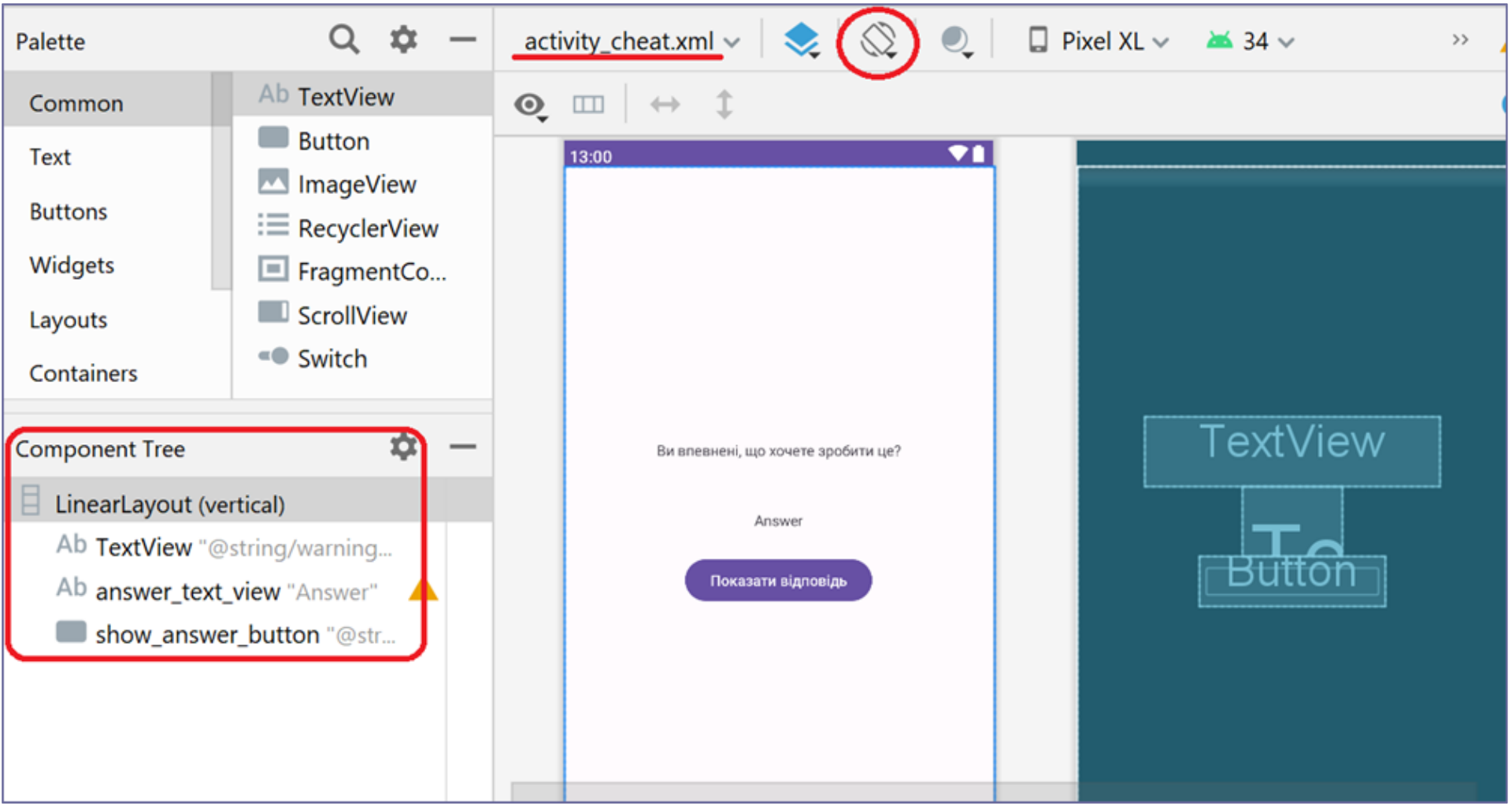
Створення додаткової активності



Створення додаткової активності



Макет додаткової активності



- Макет прийнятно виглядає в обох орієнтаціях, тому макет з альбомною орієнтацією не створюється

Макет додаткової активності

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".CheatActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="24dp"
        android:text="@string/warning_text" />

    <TextView
        android:id="@+id/answer_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="24dp"
        android:text="Answer" />    ...
```

Макет додаткової активності

```
...

    <Button
        android:id="@+id/show_answer_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/show_answer_button" />

</LinearLayout>
```

Оголошення додаткової активності у маніфесті

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

    <application
        ...
        <activity
            android:name=".CheatActivity"
            android:exported="false" />
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

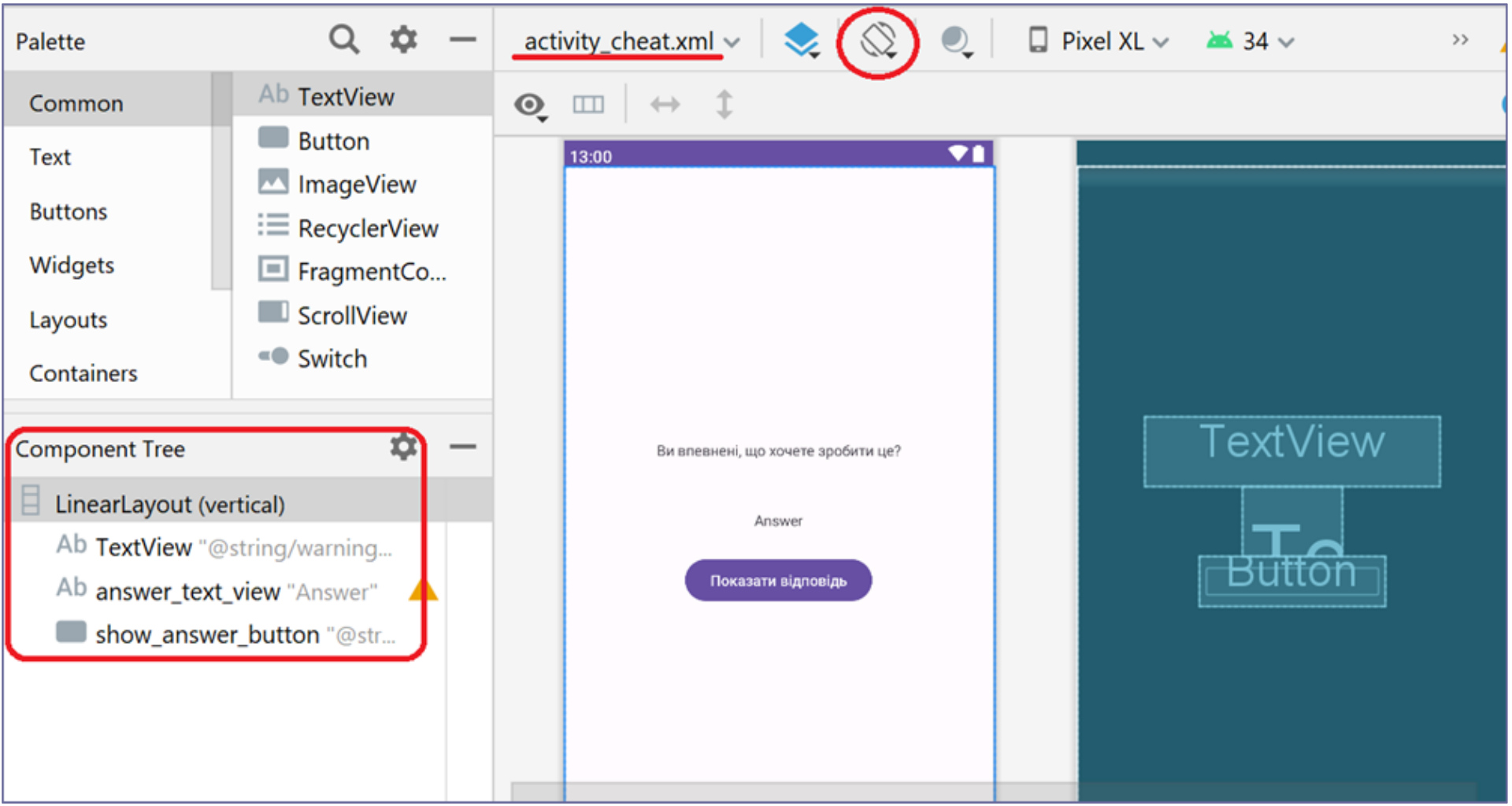
Атрибут android:exported визначає, чи може активність запускатися компонентами інших програм, для додаткової активності встановлене значення, що не дозволяє цього роботи з міркувань безпеки, а для основної активності цей атрибут повинен бути встановлений у true.

Рефакторинг макету головної активності

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

    <LinearLayout ...  <Button ... />   <Button ...  />     </LinearLayout>

    <Button
        android:id="@+id/cheat_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="@string/cheat_button" />

    <Button
        android:id="@+id/next_button"
        ... />

</LinearLayout>
```

Також ця кнопка додана
до макету альбомної орієнтації

Макет головної активності після рефакторингу



- Макет прийнятно виглядає в обох орієнтаціях, тому макет з альбомною орієнтацією не створюється

Передача інформації через інтенти - запуск додаткової активності

- Intent перекладається з англійської, як намір – намір запустити іншу активність.
- Для запуску CheatActivity з MainActivity до останньої необхідно додати оголошення кнопки відкриття додаткової активності cheatButton: Button та у методі onCreate(savedInstanceState: Bundle?) створити кнопку, використовуючи її ресурс-ідентифікатор та зареєструвати слухача подій кліку по цій кнопці.
- У слухачі подій необхідно організувати створення об'єкту класу android.content.Intent який буде пов'язаний з додатковою активністю, та викликати метод класа android.app.Activity startActivity(intent: Intent), що приймає створений інтент як параметр.

Передача інформації через інтенти - запуск додаткової активності

```
package com.example.geoquiz
import ...
private const val TAG = "MainActivity"
class MainActivity : AppCompatActivity() {
    ...
    private lateinit var nextButton: Button
    private lateinit var prevButton: Button
    private lateinit var questionTextView: TextView
    private lateinit var cheatButton: Button
    ...
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        ...
        nextButton = findViewById(R.id.next_button)
        prevButton = findViewById(R.id.prev_button)
        questionTextView = findViewById(R.id.question_text_view)
        cheatButton = findViewById(R.id.cheat_button)
        ...
```

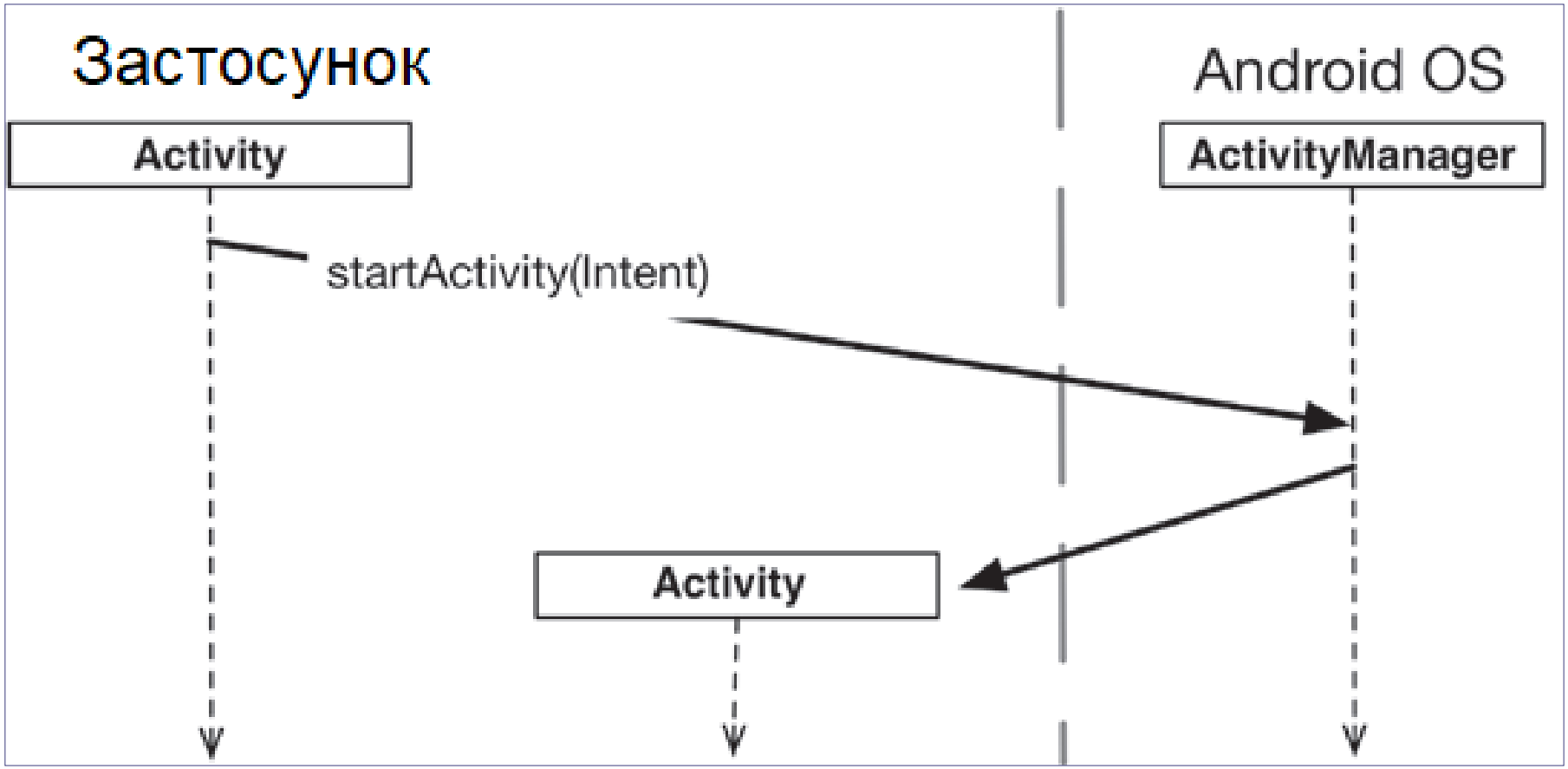
Передача інформації через інтенти - запуск додаткової активності

```
    ...
    prevButton.setOnClickListener {
        quizViewModel.moveToPrevious()
        updateQuestion()
    }

    nextButton.setOnClickListener {
        quizViewModel.moveToNext()
        updateQuestion()
    }

    cheatButton.setOnClickListener {
        val intent = Intent(this, CheatActivity::class.java)
        startActivity(intent)
    }

    updateQuestion()
}
...
}
```

Передача інформації через інтенти - запуск додаткової активності

- Коли у коді активності викликається функція startActivity(Intent), цей виклик передається компоненту операційної системи, який називається менеджером активностей (ActivityManager).
- ActivityManager створює екземпляр зазначеної об'єктом Intent активності та викликає її функцію onCreate(Bundle?).
- Перш ніж запустити активність, ActivityManager шукає у файлі AndroidManifest.xml оголошення активності, що відповідає зазначеному як аргумент об'єкту Class. Якщо таке оголошення буде знайдено, активність запускається, якщо ні – генерується ActivityNotFoundException.

Передача інформації через інтенти - запуск додаткової активності



- Використання ActivityManager, що знаходиться зовні застосунку, дозволяє запускати з одного застосунку активність іншого.

Передача інформації через інтенти

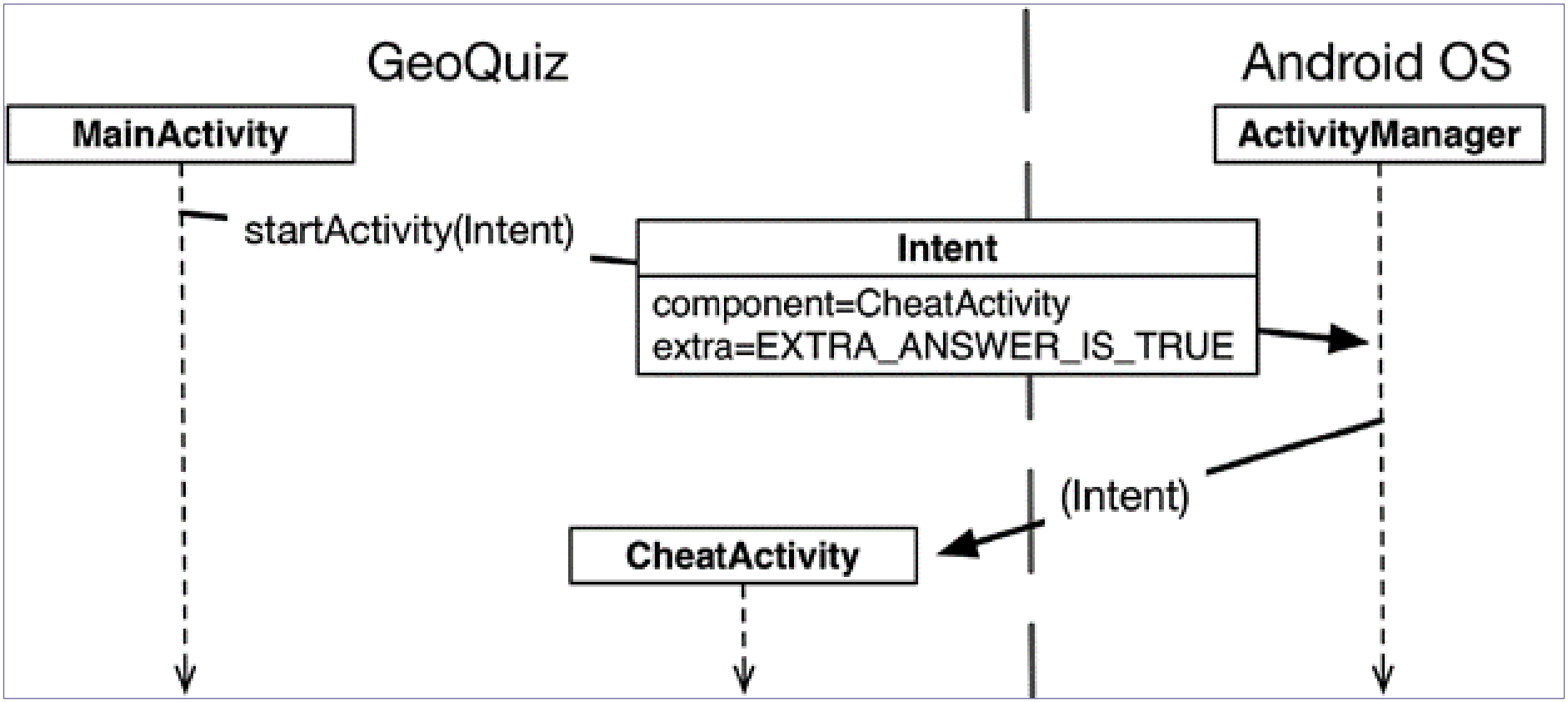
- Якщо об'єкт Intent створюється конструктором, що приймає параметрами об'єкти Context і Class (як у описаному вище прикладі), то говорять, що створюється *явний* (*explicit*) *інтент*. Явні інтенти використовуються для запуску активності у поточному застосунку.
- Якщо ж активність застосунку має запустити активність іншого застосунку (це можливо, оскільки запуск виконує компонент операційної системи ActivityManager), створюється неявний (implicit) інтент, приклади їх використання буде розглянутий далі.

- На цьому етапі можливо запустити застосунок та перевірити відкриття додаткової активності (завершити її роботу поки можна натисканням системної кнопки Назад).

Передача інформації через інтенти - передача даних додатковій активності

- У GeoQuiz від головної активності до додаткової передаватиметься відповідь на поточне запитання.
- Для передачі даних вони оформлюються як *доповнення* (*extra*) інтента, пересилаються разом з ним до цільової активності, де витягуються та використовуються.
- Доповнення інтента є парою "ключ – значення".
- Для включення доповнень до інтенту використовується функція putExtra(name: String, value: Boolean). У першому аргументі передається ключ, а у другому значення того чи іншого типу.
- Активність може запускатися з кількох різних місць, тому ключі повинні визначатися в активності, яка читає і використовує їх.
- Функція повертає об'єкт Intent, так що за потреби можна використовувати ланцюжки зі зчеплених викликів, які додають дані-доповнення до інтенту.

Передача інформації через інтенти - передача даних додатковій активності



Передача інформації через інтенти - передача даних додатковій активності

- Додаймо до CheatActivity функцію-компаньон, яка буде створювати інтент, додавати до нього доповнення і повертати його:

```
private const val EXTRA_ANSWER_IS_TRUE =
                                    "com.example.geoquiz.answer_is_true"


class CheatActivity : AppCompatActivity() {

    companion object {
        fun newIntent(packageContext: Context, answerIsTrue: Boolean): Intent {
            return Intent(packageContext, CheatActivity::class.java).apply {
                putExtra(EXTRA_ANSWER_IS_TRUE, answerIsTrue)
            }
        }
    }
    ...
}
```

Уточнення ключа доповнення іменем пакета запобігає конфліктам імен із ключами доповнень інших застосунків

Передача інформації через інтенти - передача даних додатковій активності

- Використаємо функцію-компаньон newIntent(...) у обробнику події натискання на кнопку *Підглянути* в MainActivity:

```
cheatButton.setOnClickListener {
    val answerIsTrue = quizViewModel.currentQuestionAnswer
    val intent = CheatActivity.newIntent(this@MainActivity, answerIsTrue)
    startActivity(intent)
}
```

- Для читання значення із доповнення використовується функція Intent.getBooleanExtra(String, Boolean). Перший аргумент містить ключ доповнення, а другий – відповідь за замовчуванням, якщо ключ не знайдено.

```
class CheatActivity : AppCompatActivity() {
    ...
    private var answerIsTrue = false

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_cheat)
        answerIsTrue = intent.getBooleanExtra(EXTRA_ANSWER_IS_TRUE, false)
        ...
```

Передача інформації через інтенти - передача даних додатковій активності

```
    ...
    answerTextView = findViewById(R.id.answer_text_view)
    showAnswerButton = findViewById(R.id.show_answer_button)

    showAnswerButton.setOnClickListener {
        val answerText = when {
            answerIsTrue -> R.string.true_button
            else -> R.string.false_button
        }
        answerTextView.setText(answerText)
    }
}
...
}
```

код, який забезпечує використання прочитаного значення відповіді у віджеті `answerTextView`

- Тепер після натискання на кнопку *Показати відповідь* у додатковій активності, буде відображено правильну відповідь.

Передача даних додатковій активності з поверненням результату (startActivityForResult)

- Щоб отримати результат від запущеної активності, можна викликати функцію Activity.startActivityForResult(Intent, Int). Перший параметр містить інтент, що був визначений у основній активності, з якої запущена додаткова актвність. У другому параметрі передається код запиту – визначене користувачем ціле число, яке передається додатковій активності, а потім знову приймається основною. Воно використовується тоді, коли основна активність запускає відразу декілька додаткових активностей, і їй необхідно визначити, яка з них повертає дані.
- У класі MainActivity змінимо слухача кнопки *Підглянути! :*

```
cheatButton.setOnClickListener {
        val answerIsTrue = quizViewModel.currentQuestionAnswer
        val intent = CheatActivity.newIntent(this@MainActivity, answerIsTrue)
//        startActivity(intent)
        startActivityForResult(intent, REQUEST_CODE_CHEAT)
}
```

Передача даних додатковій активності з поверненням результату (startActivityForResult)

- Функції, які можуть викликатися у додатковій активності для повернення даних-результату основній активності:

  setResult(resultCode: Int)

  setResult(resultCode: Int, data: Intent)
- Як правило, resultCode містить одну з констант: Activity. RESULT_OK (=-1) або Activity.RESULT_CANCELED (=0).
- Призначення коду результату є корисним у тому випадку, коли основна активність повинна виконати різні дії залежно від того, як завершилася додаткова активність.
- Виклик setResult(...) є необов'язковим для додаткової активності, операційна система автоматично відправить одну з вище зазначених констанат як resultCode, якщо додаткова активність була запущена функцією startActivityForResult(...). Якщо функція setResult(...) не викликалася, при натисканні користувачем системної кнопки *Назад* основна активність отримає код Activity.RESULT_CANCELED (=0).

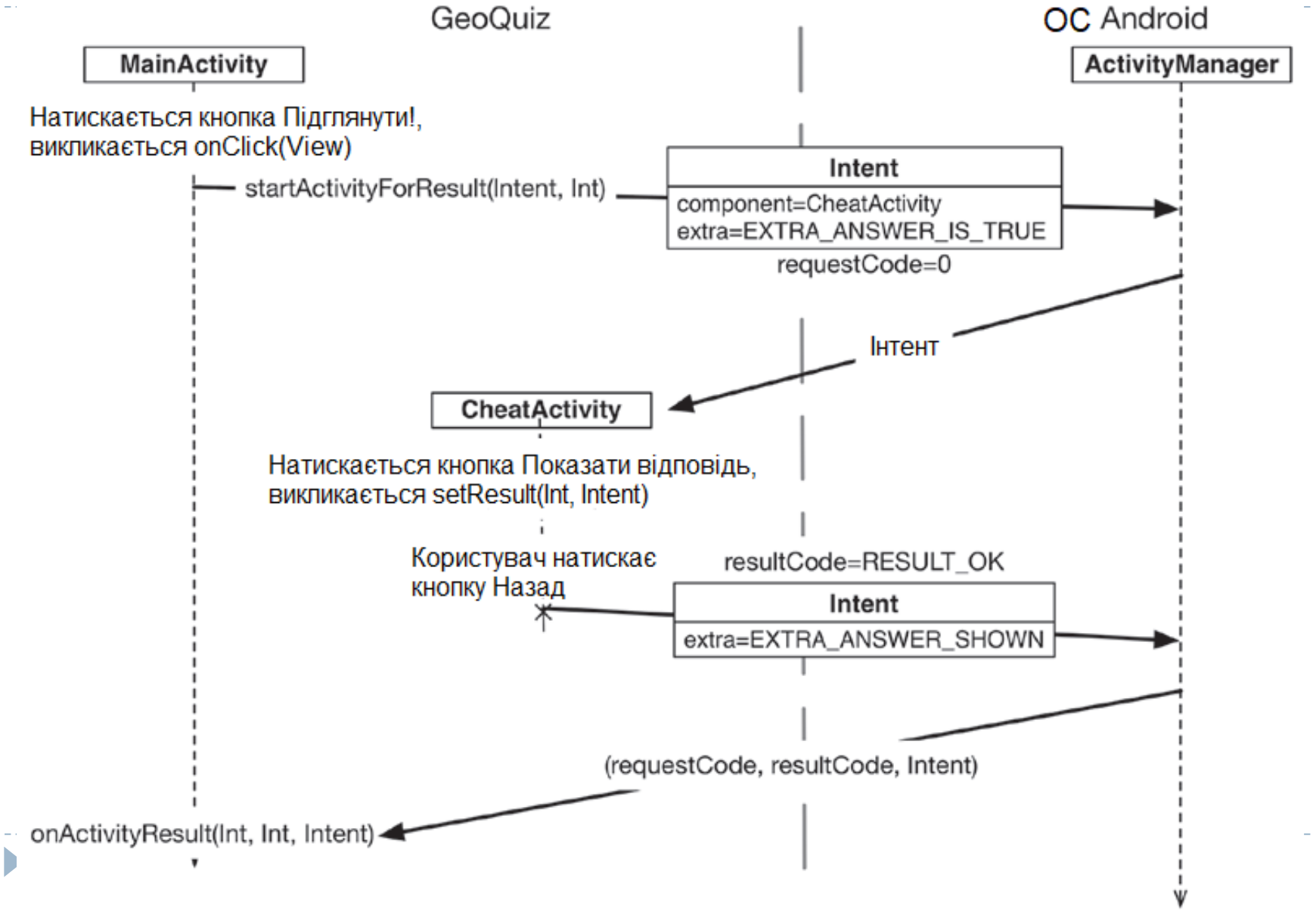
Передача даних додатковій активності з поверненням результату (startActivityForResult)

```
package com.example.geoquiz

import ...

private const val EXTRA_ANSWER_SHOWN =
                                    "com.example.geoquiz.answer_shown"
private const val EXTRA_ANSWER_IS_TRUE =
                                        "com.example.geoquiz.answer_is_true"

class CheatActivity : AppCompatActivity() {
    ...
    private var answerIsTrue = false
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_cheat)
        answerIsTrue = intent.getBooleanExtra(EXTRA_ANSWER_IS_TRUE, false)
        answerTextView = findViewById(R.id.answer_text_view)
        showAnswerButton = findViewById(R.id.show_answer_button)
        ...
```

Передача даних додатковій активності з поверненням результату (startActivityForResult)

```
        ...
        showAnswerButton.setOnClickListener {
            val answerText = when {
                answerIsTrue -> R.string.true_button
                else -> R.string.false_button
            }
            answerTextView.setText(answerText)
            setAnswerShownResult(true)
        }
    }

    private fun setAnswerShownResult(isAnswerShown: Boolean) {
        val data = Intent().apply {
            putExtra(EXTRA_ANSWER_SHOWN, isAnswerShown)
        }
        setResult(Activity.RESULT_OK, data)
    }
}
```

Передача даних додатковій активності з поверненням результату (startActivityForResult)



Передача даних додатковій активності з поверненням результату (startActivityForResult)

- Отриманий результат будемо зберігати у QuizViewModel, тому додаймо до цього класу віповідне поле isCheater (зробимо його пакетно-приватним для спрощення доступу):

```
package com.example.geoquiz

import ...

private const val TAG = "QuizViewModel"
private const val KEY_INDEX = "index"

class QuizViewModel(state: SavedStateHandle) : ViewModel() {
    ...
    private val savedStateHandle = state
    private var currentIndex = getCurrentIndex()
    var isCheater = false
    ...
}
```

Передача даних додатковій активності з поверненням результату (startActivityForResult)

- Для аналізу результату з CheatActivity, необхідно у MainActivity перевизначити метод onActivityResult(requestCode: Int, resultCode: Int, data: Intent):

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (resultCode != Activity.RESULT_OK)
        return
    if (requestCode == REQUEST_CODE_CHEAT) {
        quizViewModel.isCheater = data?.getBooleanExtra(EXTRA_ANSWER_SHOWN,
                                                        false) ?: false
        showToast(R.string.judgment_toast)
    }
}
```

- Тепер після натискання кнопки *Підглянути!* на головному екрані, а потім натискання кнопки Показати відповідь на додатковому екрані і натискання після цього кнопки *Назад,* повинно відобразитися повідомлення з осудом внаслідок підглядування.

Передача даних додатковій активності з поверненням результату (Activity Result API)

- Метод Activity startActivityForResult() оголошений застарілим і Google наполегливо рекомендує використовувати Activity Result API, представлений у класах AndroidX Activity і Fragment.
- Це пов'язано з тим, що startActivityForResult() повертає єдиний об'єкт результату для відповідей від різних активностей і розробник змушений шукати необхідний результат за кодом запиту, що робить метод onActivityResult() перевантаженим кодом.
- Activity Result API надає окремі методи зворотного виклику для кожної активності.
- Окрім того, він пропонує вбудовані контракти для типових активностей при роботі з мобільним пристроєм, як то запуск іншої активності, пропонування обрати контакт з системного застосунку Контакти, отримання зображення з камери тощо (повний список контрактів надається у документації Android).
- Також існує можливість визначити клас користувацького контракту.

Передача даних додатковій активності з поверненням результату (Activity Result API)

- Для нашого випадку можна скористуватися вбудованим контрактом ActivityResultContracts.StartActivityForResult(), який повертає об'єкт StartActivityForResult, що передається як перший аргумент до методу:

```
public final <I, O> ActivityResultLauncher<I>
                                registerForActivityResult(
    @NonNull ActivityResultContract<I, O> contract,
    @NonNull ActivityResultCallback<O> callback)
```

  класу ComponentActivity – нащадка Activity.
- Другий аргумент – це визначення функції зворотного виклику, яка буде обробляти повернутий результат так, як і при використанні onActivityResult(requestCode: Int, resultCode: Int, data: Intent).

Передача даних додатковій активності з поверненням результату (Activity Result API)

- Додаймо до MainActivity змінну getResult, яка отримуватиме результат метода registerForActivityResult() – об'єкт класу ActivityResultLauncher<I>:

```
private val getResult =
    registerForActivityResult(
        ActivityResultContracts.StartActivityForResult(),
        fun(activityResult: ActivityResult) {
            if (activityResult.resultCode == RESULT_OK) {
                quizViewModel.saveCheaterFlag(
                    activityResult.data?.getBooleanExtra(
                        EXTRA_ANSWER_SHOWN,
                        false
                    ) ?: false
                )
                showToast(R.string.judgment_toast)
            }
        }
    )
```

Передача даних додатковій активності з поверненням результату (Activity Result API)

- Після отримання лончера ми можемо його запустити у слухачі натискання на кнопку *Підглянути!*:

```
cheatButton.setOnClickListener {
    val answerIsTrue = quizViewModel.currentQuestionAnswer
    val intent = CheatActivity.newIntent(this@MainActivity, answerIsTrue)
//    startActivity(intent)
//    startActivityForResult(intent, REQUEST_CODE_CHEAT)    //deprecated
    getResult.launch(intent)
}
```

- Для формування результату, що буде переданий з CheatActivity до MainActivity додаймо до CheatActivity функцію:

```
private fun returnForResult(){
    val replyIntent = Intent ()
    replyIntent.putExtra(EXTRA_ANSWER_SHOWN, true)
    setResult(Activity.RESULT_OK, replyIntent)
}
```

Передача даних додатковій активності з поверненням результату (Activity Result API)

та викличемо її у слухачі натискання на кнопку showAnswerButton:

```
showAnswerButton.setOnClickListener {
    val answerText = when {
        answerIsTrue -> R.string.true_button
        else -> R.string.false_button
    }
    answerTextView.setText(answerText)
    returnForResult()
}
```

- Тепер застосунок використовує Activity Result API та має таку ж функціональність, як і при використанні метода startActivityForResult()
- Зауважимо, що ми перейменували у класі QuizViewModel поле isCheater на cheaterFlag та виконали його інкапсуляцію:

Передача даних додатковій активності з поверненням результату (Activity Result API)

```
package com.example.geoquiz

import ...

private const val TAG = "QuizViewModel"
private const val KEY_INDEX = "index"
private const val KEY_CHEATER_FLAG = "cheaterFlag"

class QuizViewModel(state: SavedStateHandle) : ViewModel() {
    ...
    private val savedStateHandle = state
    private var currentIndex = getCurrentIndex()
    private var cheaterFlag = getCheaterFlag()
    ...
    public fun saveCheaterFlag(isCheater: Boolean) {
        Log.i(TAG, "Current isCheater=${isCheater} saved in $this")
        savedStateHandle[KEY_CHEATER_FLAG] = isCheater
    }

    private fun getCheaterFlag(): Boolean {
        Log.i(TAG, "Is cheater=${cheaterFlag} retrieved from $this")
        return savedStateHandle[KEY_CHEATER_FLAG] ?: false
    }
}
```

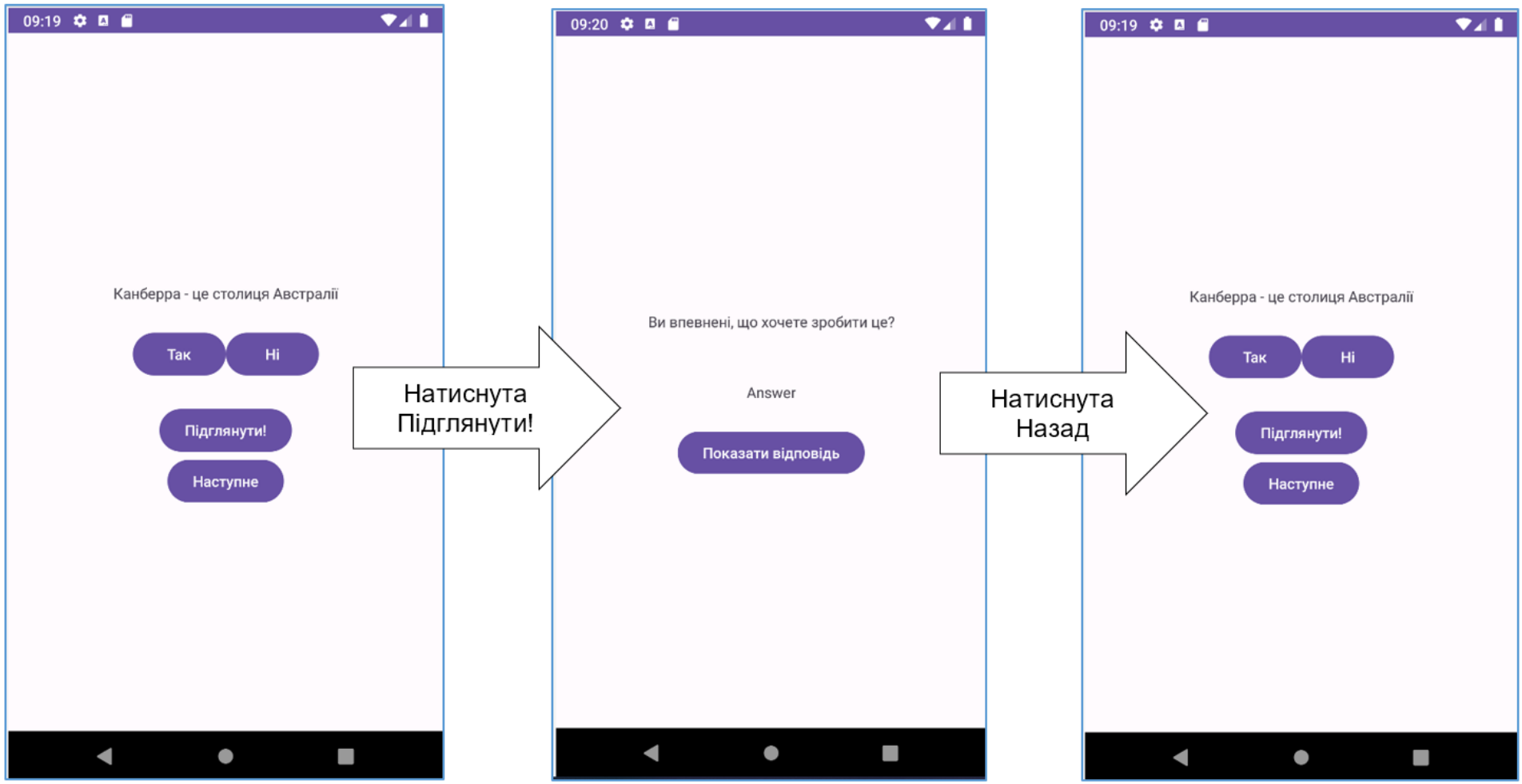
Стек повернення ОС Android

- Коли запускається застосунок, ОС запускає не застосунок, а активність застосунку.
- Але, якщо говорити точніше, запускається активність лаунчера програми.
- Для GeoQuiz активністю лаунчера є MainActivity. Статус активності лаунчера задається в маніфесті елементом intent-filter в оголошенні MainActivity:

Стек повернення ОС Android

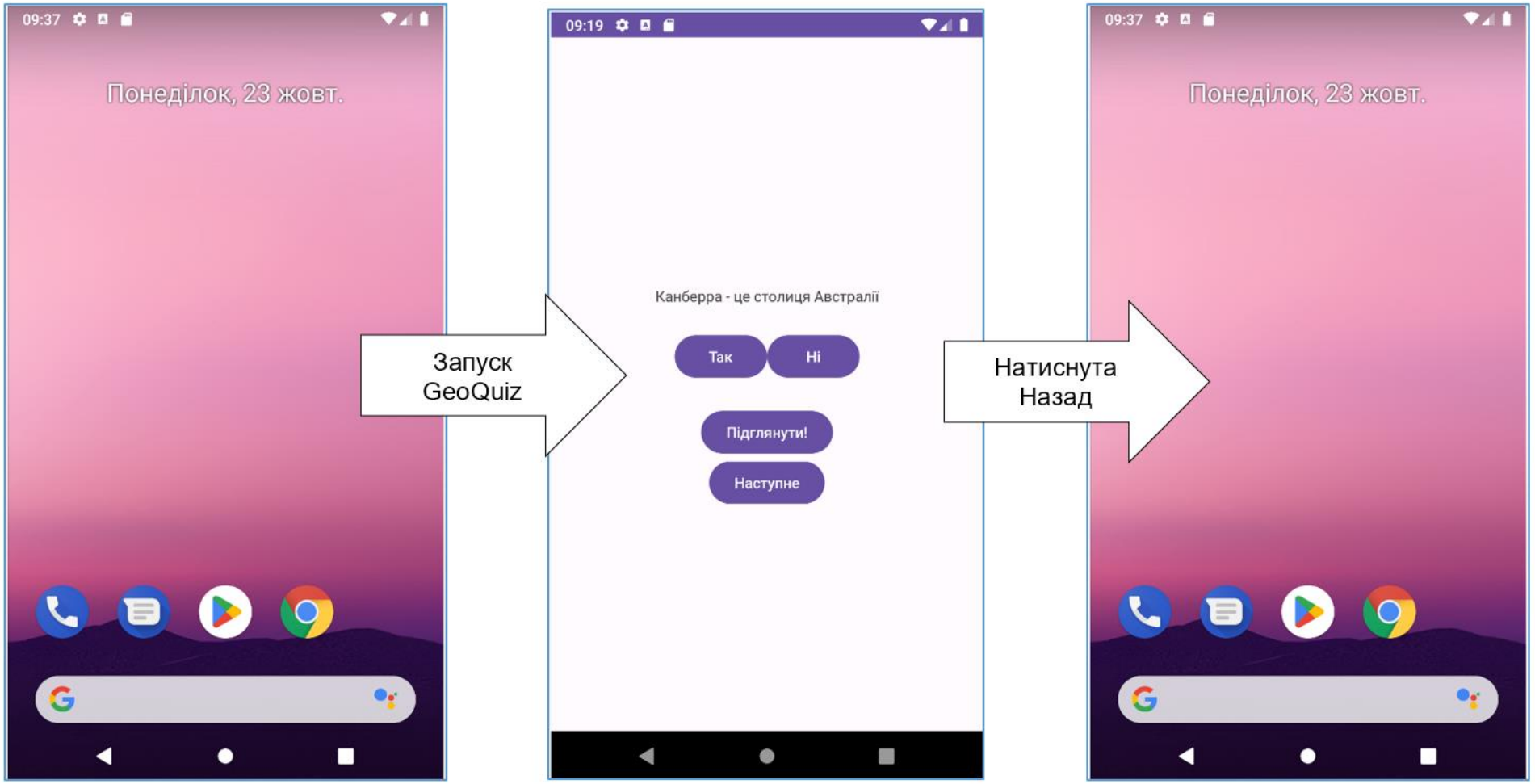
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application
                    ...
        <activity
            android:name=".CheatActivity"
            android:exported="false" />
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Стек повернення ОС Android



Стек повернення застосунку GeoQuiz

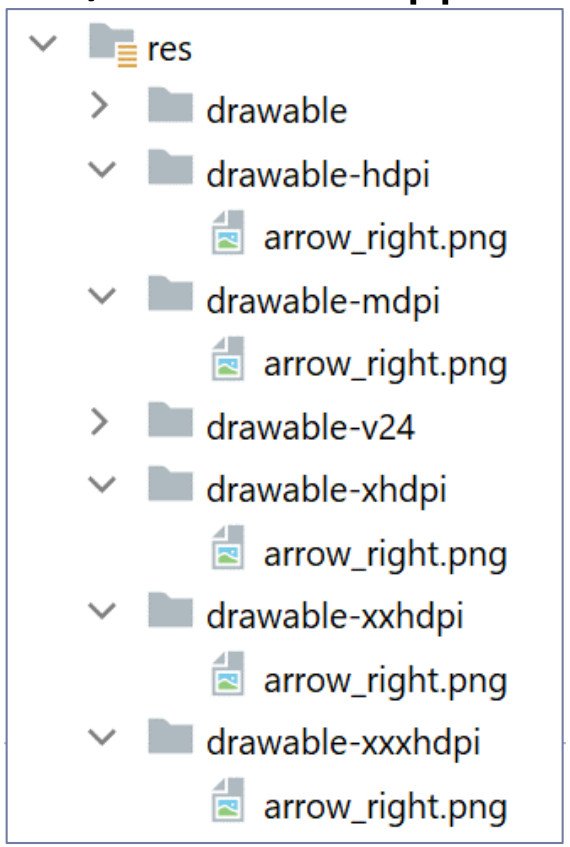
Стек повернення ОС Android

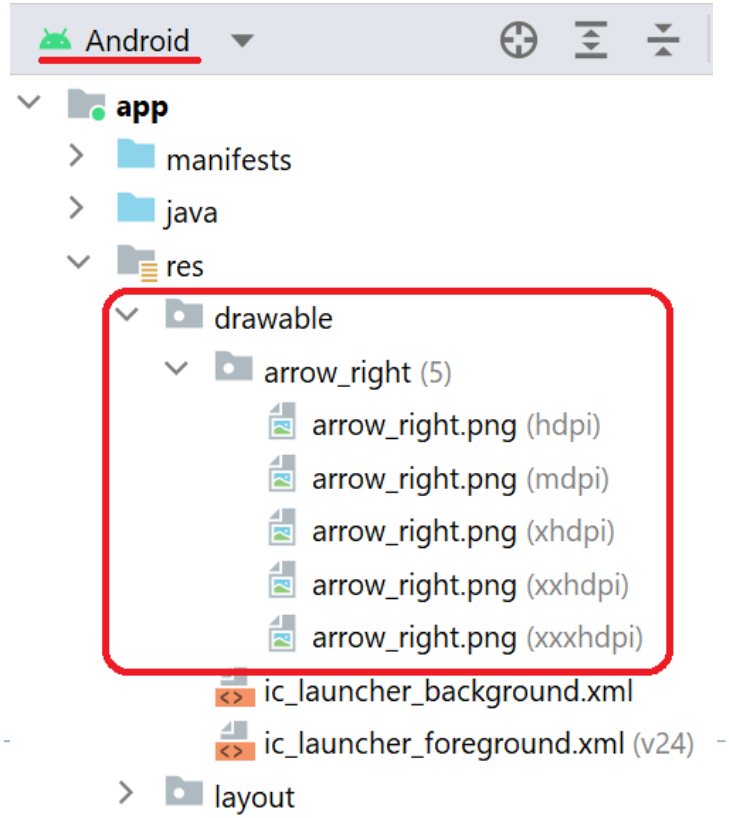


Стек повернення ОС Android

Робота з графічними ресурсами

- Застосунок GeoQuiz працює, але інтерфейс користувача виглядав би більш привабливо, якби на кнопці *Наступне* була зображена стрілка, звернена праворуч.
- Графічні файли розміщуються у підкаталогах drawable-????dpi каталогу GeoQuiz/app/src/main/res:



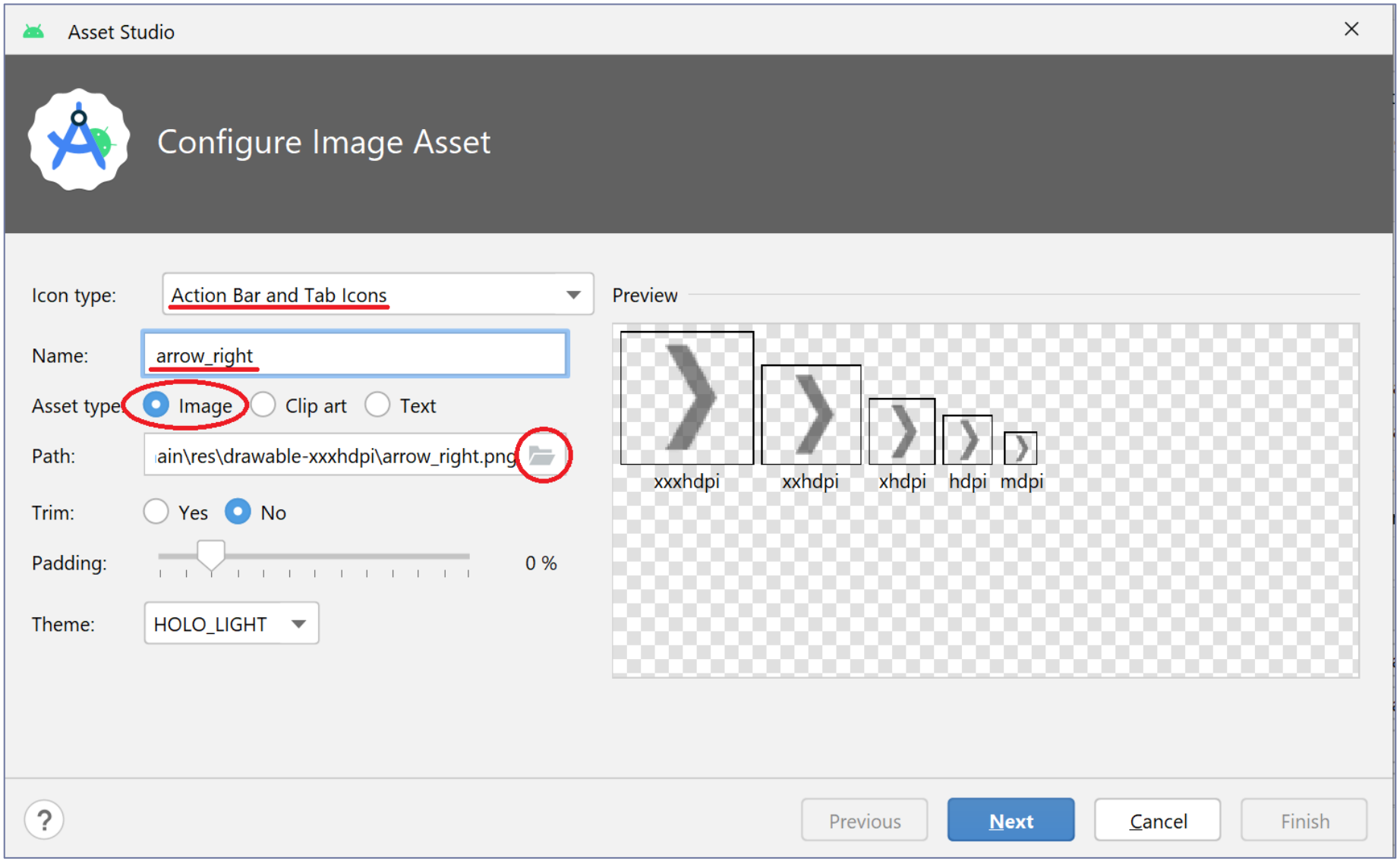


Робота з графічними ресурсами

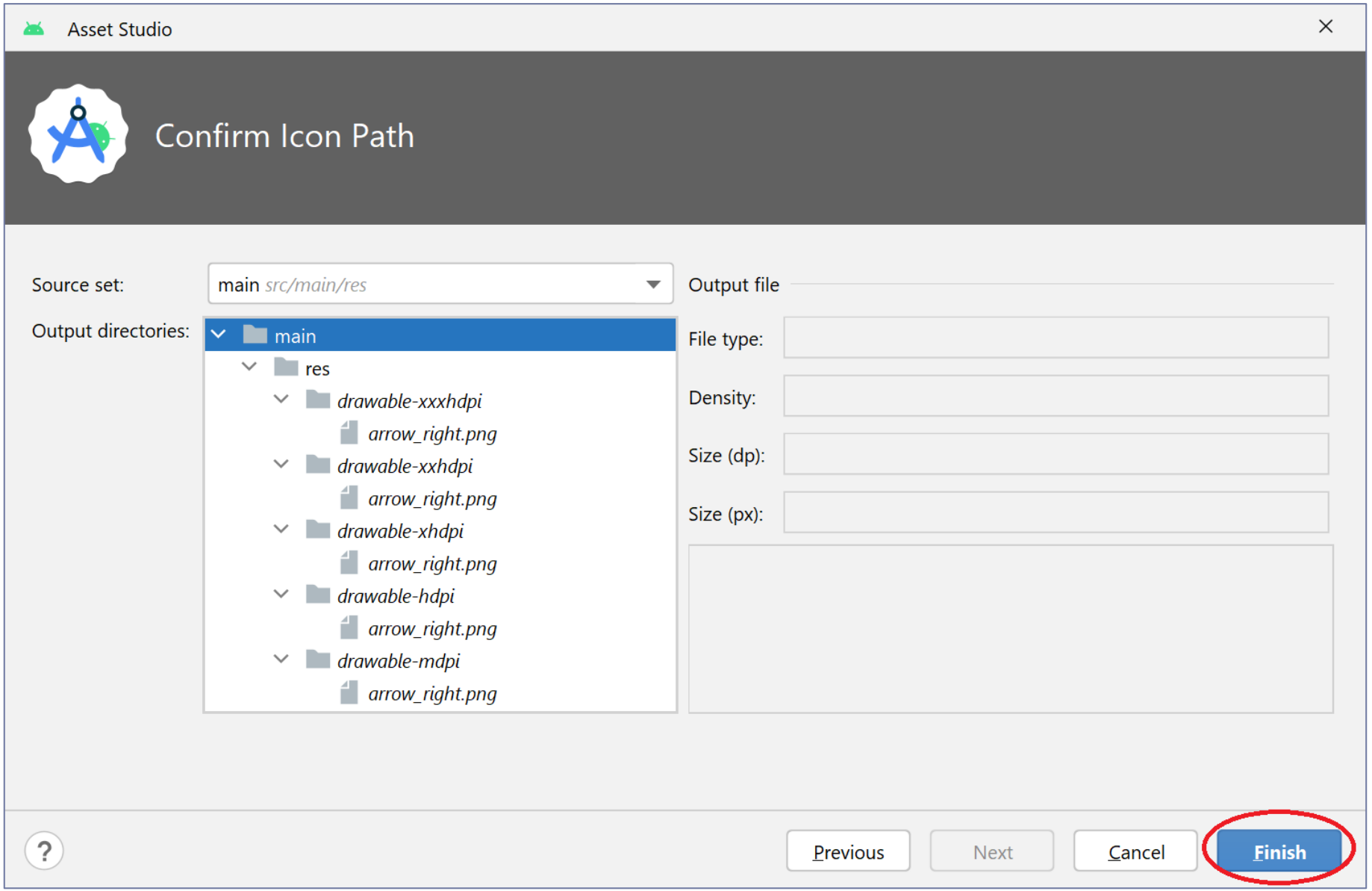
- Суфікси імен каталогів позначають екранну густину пікселів пристрою. Існують такі значення густини пікселів:
  - ldpi – середня щільність (~120 dpi) – у проєктах не використовується;
  - mdpi – середня щільність (~160 dpi) – розмір стрілки 10 x 15 пікселів;
  - hdpi – висока щільність (~240 dpi) – розмір стрілки 15 x 22 пікселів;
  - xhdpi – надвисока щільність (~320 dpi) – розмір стрілки 20 x 30 пікселів;
  - xxhdpi – наднадвисока щільність (~480 dpi) – розмір стрілки 30 x 45 пікселів;
  - xxxhdpi – наднадвисока щільність (~640 dpi) – розмір стрілки 40 x 60 пікселів.

Робота з графічними ресурсами - додання файлів зображень

New–Image Asset контекстного меню каталогу res (app\src\main\res)



Робота з графічними ресурсами - додання файлів зображень



Робота з графічними ресурсами

- При запуску ОС Android обирає файл зображення, який найбільше підходить для конкретного пристрою, на якому виконується програма.
- Якщо програма виконується на пристрої, екранна щільність якого не відповідає жодній ознаці в іменах каталогів, Android автоматично масштабує зображення до відповідного розміру.
- Завдяки цій обставині необов'язково надавати зображення всім категорій щільності.
- Будь-якому файлу .png, .jpg або .gif, доданому до каталогів res/drawable-????dpi, призначається ідентифікатор ресурсу у вигляді `@drawable/ім'я_файлу_без_розширення` (імена файлів повинні бути записані в нижньому регістрі і не можуть містити пробілів).
- Ці ідентифікатори ресурсів не уточнюються щільністю пікселів, під час запуску ОС автоматично вибере зображення, яке підходить для конкретного пристрою.

Робота з графічними ресурсами

- Для розміщення зображення стрілки на кнопці *Наступне* необхідно додати до неї атрибут `android:drawableXXX`, де `XXX` вказує місце додання по відношенню до основного контенту віджету (`Start`, `End`, `Left`, `Right`, `Top`, `Bottom`) та, за необхідністю, атрибут android:drawablePadding, що встановлює відступ зображення від основного контенту.
- Розмітка віджета–кнопки Наступне тепер буде виглядати (необхідно додати виділені атрибути і до розміти макету з горизонтальною орієнтацією):

```
<Button
    android:id="@+id/next_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:drawableEnd="@drawable/arrow_right"
    android:drawablePadding="4dp"
    android:text="@string/next_button" />
```