Міністерство освіти та науки України
Національний університет водного господарства та природокористування
Навчально-науковий інститут енергетики, автоматики та водного господарства
Кафедра автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій

04-03-444М

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт
з навчальної дисципліни

«Програмування мобільних пристроїв»

для здобувачів вищої освіти першого (бакалаврського) рівня
за освітньо-професійною програмою
«Автоматизація, комп'ютерно-інтегровані технології та робототехніка»
спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»
Частина 1

Рекомендовано науково-методичною радою з якості ННІЕАВГ

Протокол № 6 від 28 січня 2022 р.

Рівне – 2025

Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни «Програмування мобільних пристроїв» для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійною програмою «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка». Частина 1, денної та заочної форм навчання [Електронне видання] / Сафоник А. П., Клепач М. М. – Рівне : НУВГП, 2025. – 27 с.

Укладачі:

Сафоник А. П., д.т.н, професор, професор кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій;

Клепач М. М., к.т.н., доцент, доцент кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій.

Відповідальний за випуск:

Древецький В. В., д.т.н., професор, завідувач автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій.

Керівник групи забезпечення спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»: Христюк А. О., к.т.н, доцент, доцент кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій.

Лабораторна робота №1

Основні етапи розробки додатку з використанням Android Studio

1.1. Мета роботи

Розробка простого додатку, що допомагає зрозуміти структуру додатку, освоїти основні оператори, користуватися середовищем розробки.

1.2. Теоретичні відомості

1.2.1. Загальні відомості

Android Studio — інтегроване середовище розробки для платформи Android, представлене 16 травня 2013 року на конференції Google I/O менеджером по продукції корпорації Google — Еллі Паверс.

Android Studio прийшов на зміну плагіну ADT для платформи *Eclipse*. Середовище побудоване на базі вихідного коду продукту *IntelliJ IDEA Community Edition*, що розвивається компанією *JetBrains*. Android Studio розвивається в рамках відкритої моделі розробки та поширюється під ліцензією *Apache 2.0*.

IDE можна завантажити та користуватися безкоштовно. У ній є макети для створення UI, з чого зазвичай починається робота над додатком. Studio містить інструменти для розробки рішень для смартфонів і планшетів, а також нові технологічні рішення для Android TV, Android Wear, Android Auto, Glass і додаткові контекстуальні модулі.

Середовище розробки адаптоване для виконання типових завдань, що вирішуються в процесі розробки застосунків для платформи Android. У тому числі у середовище включені засоби для спрощення тестування програм на сумісність з різними версіями платформи та інструменти для проектування застосунків, що працюють на пристроях з екранами різної роздільності (планшети, смартфони, ноутбуки, годинники, окуляри тощо). Крім можливостей, присутніх в *IntelliJ IDEA*, в *Android Studio* реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання застосунків, заснована на складальному інструментарії *Gradle* і підтримуюча використання засобів безперервної інтеграції.

Для прискорення розробки застосунків представлена колекція типових елементів інтерфейсу і візуальний редактор для їхнього компонування, що надає зручний попередній перегляд різних станів інтерфейсу застосунку (наприклад, можна подивитися як інтерфейс буде виглядати для різних версій Android і для різних розмірів екрану). Для створення нестандартних інтерфейсів присутній майстер створення власних елементів оформлення, що підтримує використання шаблонів. У середовище вбудовані функції завантаження типових прикладів коду з *GitHub*.

До складу також включені пристосовані під особливості платформи Android розширені інструменти рефакторингу, перевірки сумісності з минулими випусками, виявлення проблем з продуктивністю, моніторингу споживання пам'яті та оцінки зручності використання. У редактор доданий режим швидкого внесення правок. Система підсвічування, статичного аналізу та виявлення помилок розширена підтримкою Android API. Інтегрована підтримка оптимізатора коду *ProGuard*. Вбудовані засоби генерації цифрових підписів. Надано інтерфейс для управління перекладами на інші мови.

Деякі особливості будуть пізніше розгорнуті для користувачів так як програмне забезпечення розвивається; наразі, передбачені такі функції:

- Живі макети (*layout*): редагувальник *WYSIWYG* — живе кодування — подання (rendering) програми в реальному часі.
- Консоль розробника: підказки по оптимізації, допомога по перекладу, стеження за напрямком, агітації та акції — метрики Google аналітики.
- Резерви бета релізів та покрокові релізи.
- Базування на *Gradle*.
- Android-орієнтований рефакторинг та швидкі виправлення.
- *Lint* утиліти для охоплення продуктивності, юзабіліті, сумісності версій та інших проблем.
- Використання можливостей *ProGuard* та підписів до програм.
- Шаблони для створення поширених Android дизайнів та компонентів.
- Багатий редактор макетів (l*ayouts*) що дозволяє користувачам перетягнути і покласти (*drag-and-drop*) компоненти користувацького інтерфейсу, як варіант, переглянути одночасно макети (*layouts*) на різних конфігураціях екранів.

Android Studio сумісна з платформою *Google App Engine* для швидкої інтеграції у хмари нових API та функцій. У розробці ви знайдете різні API, такі як *Google Play, Android Pay і Health*. Є підтримка всіх платформ Android, починаючи з *Android 1.6*. Є варіанти Android, які суттєво відрізняються від версії Google Android. Найпопулярніша з них – це *Amazon Fire OS*. В Android Studio можна створювати APK для цієї ОС. Підтримка *Android Studio* обмежується онлайн-форумами.

1.3. Програма роботи

1. Вивчити теоретичні відомості.
2. Ознайомитися з основами функціоналу в Android Studio.
3. Розробити додаток «Вгадай число».
4. Виконати звіт.

1.4. Порядок виконання роботи

Для досягнення поставленої цілі в лабораторній роботі створіть додаток у середовищі розробки Android Studio, докладно розглянемо структуру отриманого проекту і розберемо призначення основних його елементів.

Завдання:
Розробити додаток "Відгадай число".

Суть додатку:
Програма випадковим чином "загадує" число від 0 до 100. Користувач повинен відгадати число. При кожному введенні числа - програма повідомляє користувачеві результат: введене число більше загаданого, менше або "Ура, перемога!" число відгадано. Додаток виконує свої функції тільки тоді, коли видимий на екрані, коли він не видимий його робота призупиняється, тобто маємо справу з додатком переднього плану, фонові процеси не передбачені.

Далі в роботі розглянемо найпростіші елементи інтерфейсу користувача і додамо їх в додаток, а також розглянемо питання, пов'язані безпосередньо з програмуванням: навчимося обробляти події, що виникають при взаємодії програми з користувачем; реалізуємо логіку перевірки числа на співпадіння із заданим.

1.4.1. Створення додатку та вивчення його структури

Створіть новий проект в середовищі Android Studio (Рис. 1.1)

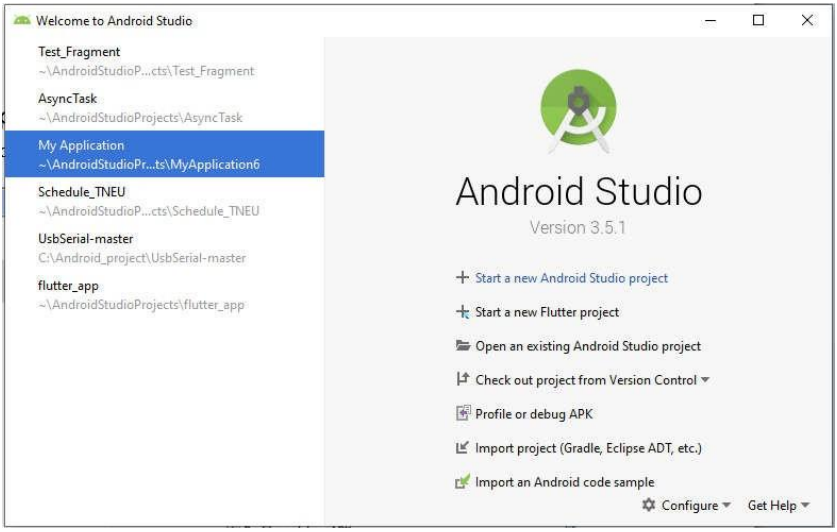

Рис. 1.1 – Головне вікно створення додатку

Вкладка *Configure – SDK Manadger* дозволяє налаштувати *SDK (Software Development Kit)* для підтримки різних версій Android (Рис. 1.2).

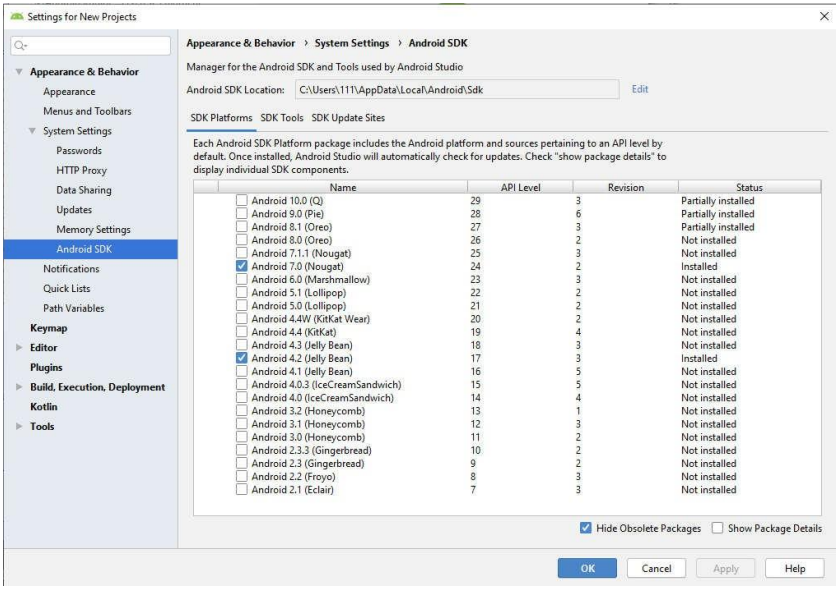

Рис. 1.2 – Вікно налаштувань версій SDK

Після завершення налаштувань повертаємось в головне вікно і вибираємо *Start new Android Studio project* (Рис. 1.3), та вибираємо тим проекту *Empty Activity*.

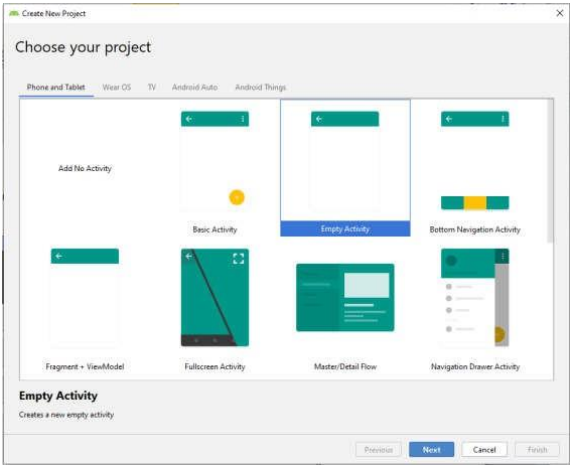

Рис 1.3 – Створення нового додатку

Далі вводимо ім’я проекту, мову програмування та мінімальний рівень *API* (Рис. 1.4).

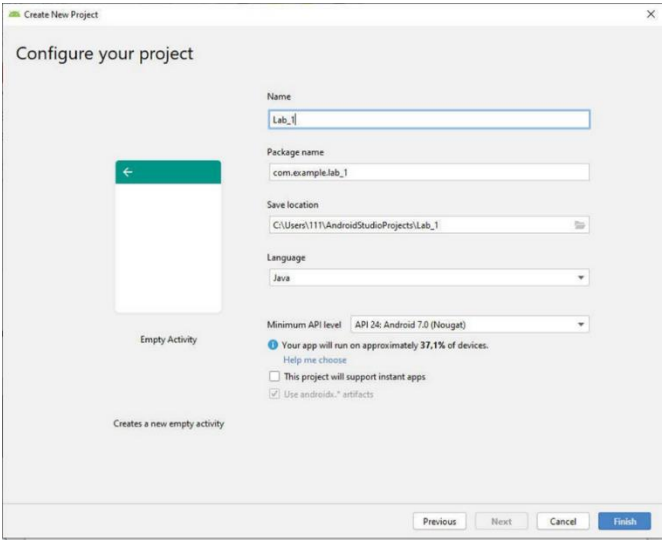

Рис. 1.4 – Ввід назви додатку, вибір мови програмування та мінімального рівня API

У процесі створення проекту, ми назвали його *Lab_1*, середовище розробки готує необхідні папки і файли. Повний ієрархічний список обов'язкових елементів проекту можна побачити на вкладці *Project*, ієрархія отриманих папок і файлів для нашого проекту зображена на рис. 1.5.

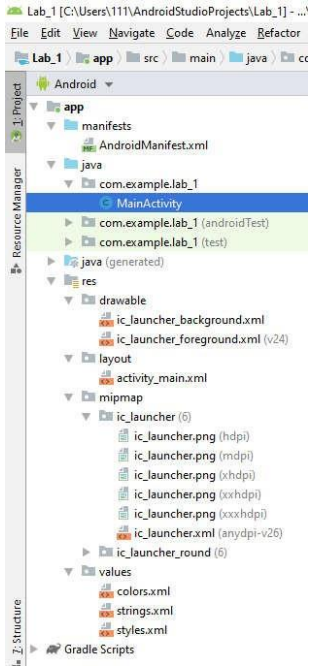

Рис. 1.5 – Структура проекту

В даний час нас буде цікавити призначення декількох файлів і папок. Розглянемо папки:

Папка *manifests* – містить файл *AndroidManifest.xml* - файл у форматі *xml*, який описує основні властивості проекту, дозвіл на використання ресурсів пристрою та ін.

Папка *java* - містить файли з вихідним кодом на мові Java. Саме в цій папці розміщуються всі класи, створювані в процесі розробки програми. Зараз в цій папці в пакеті *com.example.lab_1* розміщується єдиний клас *MainActivity.java*. Цей клас визначає головну і єдину активність в цьому додатку.

Примітка 1: Ім'я пакету присвоюється в процесі створення програми в полі *Package Name*, використовувати *com.example* не рекомендується, так як пакет з таким ім'ям не можна завантажити в *Google Play*. Часто рекомендують використовувати в якості імені пакета назву сайту програміста, записане в зворотному порядку, можна просто використовувати свої ім'я та прізвище. Останнє слово в імені пакету формується автоматично і збігається з ім'ям проекту.

Примітка 2: Ім'я файлу присвоюється в процесі створення програми на етапі налаштування активності. Ім'я визначається в поле *Activity Name*.

Папка *res* - містить структуру папок ресурсів програми, розглянемо деякі з них:

- *layout* - в цій папці містяться xml-файли, які описують зовнішній вигляд форм і їх елементів, поки там знаходиться тільки *activity_main.xml*;
- *values* - містить XML файли, які визначають прості значення, таких ресурсів як, рядки, числа, кольори, теми, стилі, які можна використовувати в цьому проекті;
- *menu* - містить XML файли, які визначають все меню програми.

Розглянемо файл *AndroidManifest.xml* - файл у форматі *xml*, який описує основні властивості проекту, дозвіл на використання ресурсів пристрою та ін. Відразу після створення програми файл *AndroidManifest.xml* виглядає так:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/re
    s/android"package="com.example.lab_1">

    <application
        android:allowBac
        kup="true"
        android:icon="@m
        ipmap/ic_launche
        r"
        android:label="@
        string/app_name"
        android:roundIco
        n="@mipmap/ic_la
        uncher_round"
```

```
        android:supports
        Rtl="true"
        android:theme="@
        style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action
                android:name="android.intent.action.MAIN"
                />

                <category
                android:name="android.intent.category.LAUNC
                HER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Приклад фрагмету файлу маніфесту з додатковими елементами.

```
<manifest
        xmlns:android="http://schemas.android
        .com/apk/res/android"package="string"
        android:sharedU
        serID="string"
        android:sharedU
        serLabel="strin
        g resource"
        android:version
        Code="integer"
        android:version
        Name="string"
        android:installLocation=["string" | "internalOnly"
        | "preferExternal"]
>
    . . .
</manifest>
```

Розглянемо докладно файл маніфесту.

Перший обов'язковий елемент <*manifest*> є кореневим елементом файлу, повинен містити обов'язковий елемент <*application*> і всі інші елементи за потребою. Розглянемо основні атрибути цього елемента:

Таблиця 1.1

xmlns: android	визначає простір імен Android, завжди повинен мати значення: http://schemas.android.com/apk/res/android. *Обов'язковий атрибут*
package	повне ім'я пакета, в якому розташовується додаток. *Обов'язковий атрибут*. Ім'я повинно бути унікальним, може містити великі і малі латинські букви, числа і символ підкреслення. Однак починатися повинно тільки з букви. Для уникнення конфліктів з іншими розробниками рекомендується використовувати ім'я вашого сайту (якщо він є) записаний в зворотньому порядку. У нашому випадку пакет має ім'я *com.example.lab_1*. Увага: якщо Ви опублікували свій додаток, Ви не можете змінювати ім'я пакета, тому що ім'я пакета є унікальним ідентифікатором для програми та в разі його зміни додаток буде розглядатися, як зовсім інший і користувачі попередньої версії не зможуть його оновити.
android: version Code	внутрішній номер версії програми не видно користувачеві. Цей номер використовується тільки для визначення чи є одна версія більш сучасною в порівнянні з іншого, більший номер показує більш пізню версію.
android: version Number	номер версії, є рядком і використовується тільки для того, щобпоказати користувачеві номер версії програми.
android: share UserID, android: Shared UserLabel, android: install Location	ці атрибути в нашому файлі маніфесту не представлені, про їх призначення можна почитати за посиланням: http://developer.android.com/guide/topics/manifest/manifest- element.html.

Розглянемо елемент <*application*>, який є обов'язковим елементом маніфесту, повністю визначає склад додатку. Являє собою контейнер для елементів <*activity»,«service»,«receiver»,«provider*>, кожен з яких визначає відповідний компонент програми. Містить набір атрибутів, дія яких поширюється на всі компоненти програми. Розглянемо атрибути елемента <*application*>, представлені в маніфесті:

Таблиця 1.2

android: allow Backup	визначає дозвіл для додатку брати участь в резервному копіюванні та відновленні. Якщо значення цього атрибута *false*, то для додатку не може бути створена резервна копія, навіть якщо проводиться резервне копіювання всієї системи цілком. За замовчуванням значення цього атрибута дорівнює *true.*
android: icon	визначає іконку для додатку в загальному, а також іконку за замовчуванням для компонентів додатку, яка може бути перевизначена атрибутом *android: icon* кожного компоненту. Здається як посилання на графічний ресурс, що містить зображення, в нашому випадку значення цього атрибута дорівнює *"@mipmap/ ic_launcher"*.

android: label	визначає видимий для користувача заголовок додатку в загальному, а також заголовок за замовчуванням для компонентів додатку, який може бути перевизначений атрибутом *android: label* кожного компонента. Здається як посилання на стрічковий ресурс, в нашому випадку значення атрибута дорівнює "*Lab_1*".
android: theme	визначає тему за замовчуванням для всіх активностей додатку, може бути перевизначений атрибутом *android: theme* кожної активності. Здається як посилання на стильовий ресурс, в нашому випадку значення атрибута дорівнює *"@style/AppTheme"*.

Насправді у елемента <*application*> набагато більше атрибутів, ніж нам вдалося розглянути, знайти повний список атрибутів з описами можна за посиланням:
http://developer.android.com/guide/topics/manifest/application-element.html.

У додатку всього одна активність, інших компонентів немає, в зв'язку з цим елемент <*application*> в маніфесті містить рівно один елемент <*activity*> і більше ніяких інших елементів не містить. Розглянемо елемент <*activity*>, який визначає активність. Для кожної активності обов'язково необхідний свій елемент <*activity*> в маніфесті. Розглянемо атрибути елемента <*activity*>, представлені в маніфесті:

Таблиця 1.3

android: name	визначає ім'я класу, який задає активність. Значення атрибута повинно повністю визначати ім'я класу із зазначенням пакета, в якому розташовується клас. Наприклад *"com.tneu.scs.lab_1.MainActivity"*. Можна використовувати скорочений запис *".MainActivity"*, в цьому випадку додається ім'я пакета, визначену відповідним атрибутом елемента <*manifest*>.
android: configChanges	перераховує зміни конфігурації, якими може керувати активність. Якщо конфігурація змінюється під час роботи, то за замовчуванням активність зупиняється і перезапускається. Якщо ж зміна конфігурації вказано в цьому атрибуті, то при появі цієї зміни активність не перезапускається, замість цього вона продовжує працювати і викликає метод *onConfigurationChanged()*. Наприклад: атрибут має значення *"orientation | keyboardHidden | screenSize"*, тобто при зміні орієнтації екрану, зміні розміру екрану і зміні доступності клавіатури не відбудеться перезапуск активності.
android: label	визначає видимий користувачеві заголовок активності, якщо він відрізняється від загального заголовка програми. Здається як посилання на строковий ресурс.
android: theme	визначає тему активності, якщо вона відрізняється від загальної теми програми, заданої відповідним атрибутом елемента <*application*>. Здається як посилання на стильової ресурс, наприклад значення атрибута може дорівнювати *"@style/FullscreenTheme"*.

Насправді у елемента <*activity*> набагато більше атрибутів, ніж нам вдалося розглянути, знайти повний список атрибутів з описами можна за посиланням:
http://developer.android.com/guide/topics/manifest/application-element.html.

У маніфесті для нашого застосунку елемент <*activity*> містить рівно один елемент:

<*intent-filter*>, що визначає типи намірів, які може приймати активність. Цей елемент містить два елементи: <action> і <category>.

Перший елемент визначає дії, які проходять в фільтр намірів, при цьому <*intent-filter*> повинен містити хоча б один елемент <*action*>, в іншому випадку жоден об'єкт-намір не зможе пройти через фільтр і активність не можливо буде запустити. Елемент <*action*> має єдиний атрибут *android: name = "android.intent.action.MAIN"*.

Другий елемент визначає ім'я категорії в фільтрі намірів. Має єдиний атрибут *android: name = "android.intent.category.LAUNCHER"*.

На цьому розбір маніфесту додатки закінчимо, докладно з описом всіх елементів цього файлу можна ознайомитися за посиланням:
http://developer.android.com/guide/topics/manifest/manifest-intro.html.

Найчастіше, при створенні програми доводиться мати справу з папками *java, res/layout і res/values*, тому що там знаходяться основні файли проекту.

1.4.2. Налаштування інтерфейсу додатку

До того, як почнемо формувати інтерфейс, є сенс підготувати можливість перевірки додатку на помилки. Щоб не завантажувати кожен раз додаток на реальний пристрій, в *Android SDK* передбачена можливість використання віртуального пристрою *(AVD або Android Virtual Device)*, котрий емулює роботу реального смартфона.

Для створення зовнішнього виду програми необхідно визначити які елементи графічного інтерфейсу нам потрібні, як ці елементи будуть розташовуватися на формі і яким чином буде реалізовано взаємодію з користувачем.

Так як додаток дуже простий, то і інтерфейс особливою складністю відрізнятися не буде. Нам буде потрібне поле для введення чисел

(TextEdit), текстова мітка для виведення інформації *(TextView)* і кнопка для підтвердження введеного числа *(Button)*. Розташовувати елементи інтерфейсу будемо один під одним, зверху інформаційна частина, нижче поле вводу, кнопку розмістимо в самому низу програми. Взаємодія програми з користувачем організовується дуже просто: користувач вводить число в поле вводу і натискає кнопку, читає результат в інформаційному полі і, або радіє перемозі, або вводить нове число.

Якщо користувач вводить правильне число, то додаток пропонує йому зіграти знову при цьому кнопка буде грати роль підтвердження, а в інформаційне поле буде виведено запрошення до повторної гри.

Схематично інтерфейс програми зображений на рис. 1.6

Рис. 1.6 – Схема інтерфейсу додатку «Відгадай число»

Нам необхідно додати на форму три елементи: інформаційне поле *(TextView)*, поле вводу *(TextEdit)* і кнопку *(Button)*.

Android IDE підтримує два способи для виконання дій по формуванню інтерфейсу програми: перший базується на XML-розмітці, другий належить до візуального програмування і дозволяє перетягувати об'єкти інтерфейсу і розміщувати їх на формі за допомогою мишки. Вважається, що візуальний спосіб підходить для новачків, а більш просунуті розробники можуть писати код вручну, однак найчастіше використовується комбінований підхід.

Для формування інтерфейсу будемо працювати з файлом *res/layout/activity_main.xml*. На рис. 1.7 можна побачити редактор, відповідний візуальному способу формування інтерфейсу, цьому режиму відповідає вкладка *Design*.

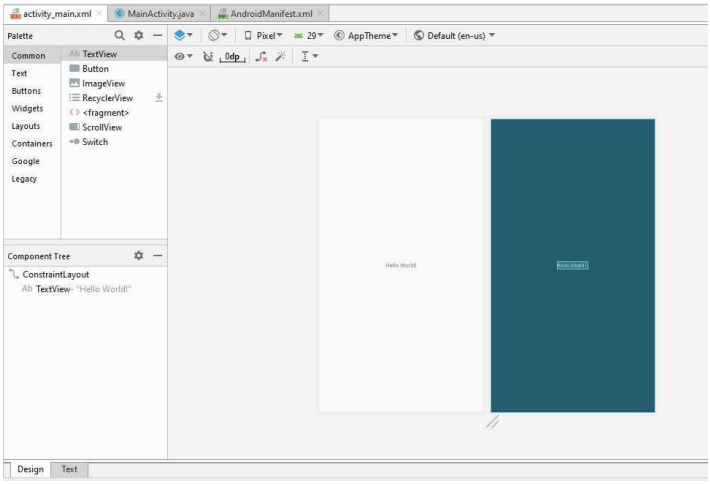

Рис. 1.7 – Графічне зображення активності додатку

На рис. 1.7 поруч з вкладкою *Design* розташована вкладка *Text*. Вона відповідає режиму редагування інтерфейсу шляхом формування XML файлу. На рис. 1.8 можна побачити редактор XML файлу.

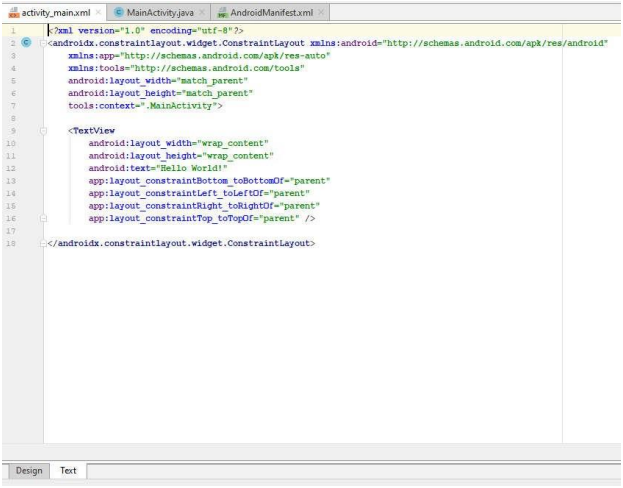

Рис. 1.8 – Опис активності в XML форматі

Задамо лінійне розташування компонентів на формі, для цього виберемо вкладку *Layouts*, знайдемо там *LineralLayout* і додамо його на форму. На рис. 1.9 можна побачити результат цих дій.

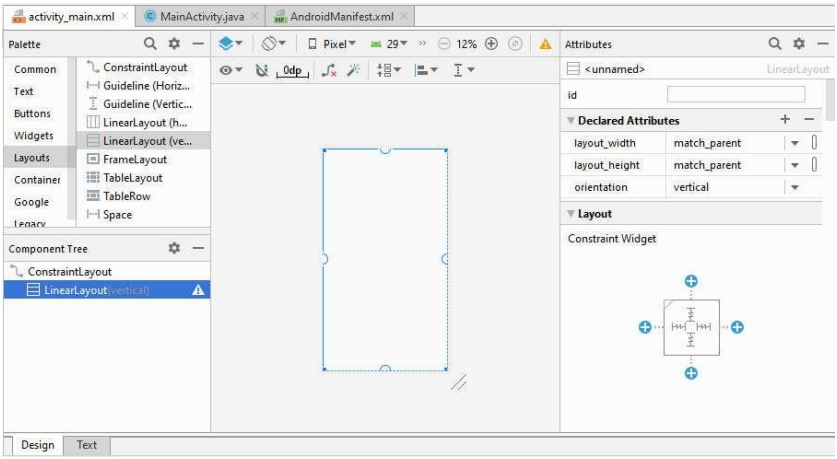

Рис. 1.9 – Налаштування інтерфейсу

Тепер почнемо додавати елементи інтерфейсу, будемо використовувати графічний режим редагування.

Для початку необхідно додати інформаційне поле. Для цього на панелі *Palette* вибираємо вкладку *Text*, на цій вкладці знайдемо поле *TextView*, перенесемо у вікно програми, розмістимо в першому рядку макету *(LinearLayout)*.

Наступним потрібно розмістити поле вводу інформації, на вкладці Text знайдемо текстове поле *Number(Decimal)* і розмістимо у другому рядку макету.

І останнім на вкладці *Button*, виберемо там елемент *Button* і додамо в третій рядок макету рис. 1.10.

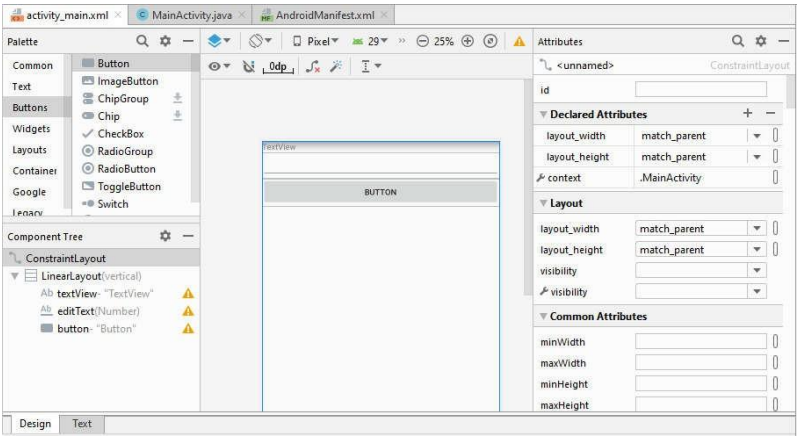

Рис. 1.10 – Інтерфейс програми

Після настройки інтерфейсу можна заглянути в файл *activity_main.xml*, в цьому файлі прописано, що використовується *LinearLayout* і дано опис кожного з трьох рядків.

Тепер необхідно наповнити наші елементи інтерфейсу змістом, нам знадобиться текст для спілкування з користувачем, при програмуванні під Android існує практика розділяти ресурси і код програми. Для зберігання будь-яких рядків, які можуть знадобитися для роботи з додатком, використовується файл *strings.xml*. Зберігання всіх строкових ресурсів в цьому файлі серйозно полегшує локалізацію додатку на інші мови. Цей файл можна знайти в *Project Explorer* в папці *res/values*. Відкриємо його і подивимося, що там є. Приберемо зайві рядки і додамо нові:

```
<resources>
    <string name="app_name">Lab_1</string>
    <string name="behind">Більше</string>
    <string name="ahead">Менше</string>
    <string name="hit">Відгадано</string>
    <string name="input_value">Ввести
    значення</string>
    <string name="play_more">Зіграти ще</string>
    <string name="try_to_guess">Спробуйте відгадати
    число (1 &#8211;
100)</string>
    <string name="error">Невірний ввід!</string>

</resources>
```

Дані змінні будуть виконувати такі завдання:

- *app_name* встановить "видиму" назву додатку;
- *behind, ahead, hit* сповістять користувача про його успіхи в грі;
- *play_more і try_to_guess* встановить назву кнопки, яка пояснить її функції;
- *input_value* запросить користувача до введення числа;
- *error* повідомить при невірному вводі.

Після зміни *strings.xml*, при переході на іншу вкладку, не забудьте зберегти зміни (найшвидший спосіб - натиснути Ctrl + S).

Налаштуємо текст в інформаційному полі. Для цього виберемо елемент *textView* (або в вікні дизайну або в списку елементів *Component Tree*) і на вкладці *Attributes* в правій частині вікна (це і є наше інформаційне поле, є сенс придумати йому більш осмислене ім'я) знайдемо властивість *Text*, підставимо в нього значення рядка з ім'ям *try_to_guess*, див. Рис. 1.11.

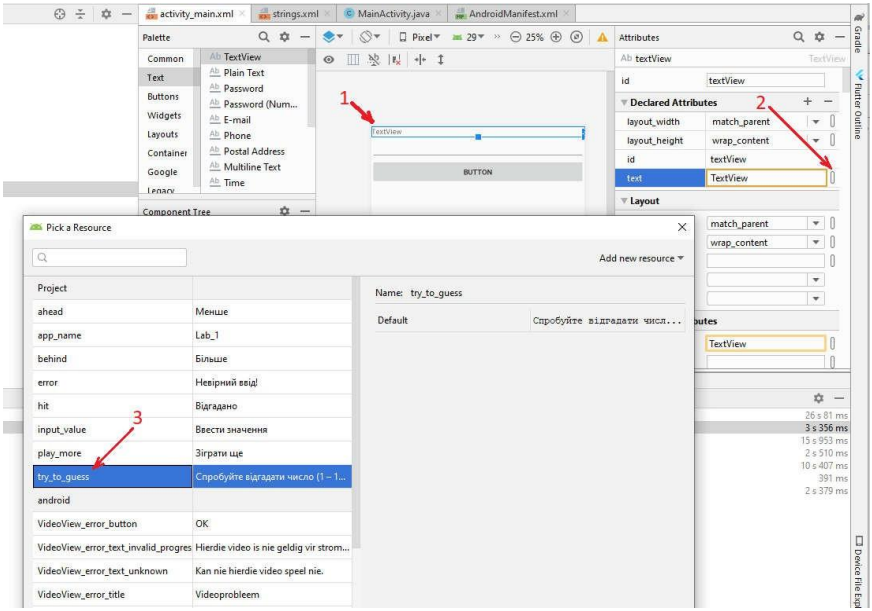

Рис. 1.11 – Налаштування текстової інформації для TextView

Аналогічно можна налаштувати текст, яким нас вітатиме кнопка, тільки в цьому випадку треба працювати з елементом *button*.

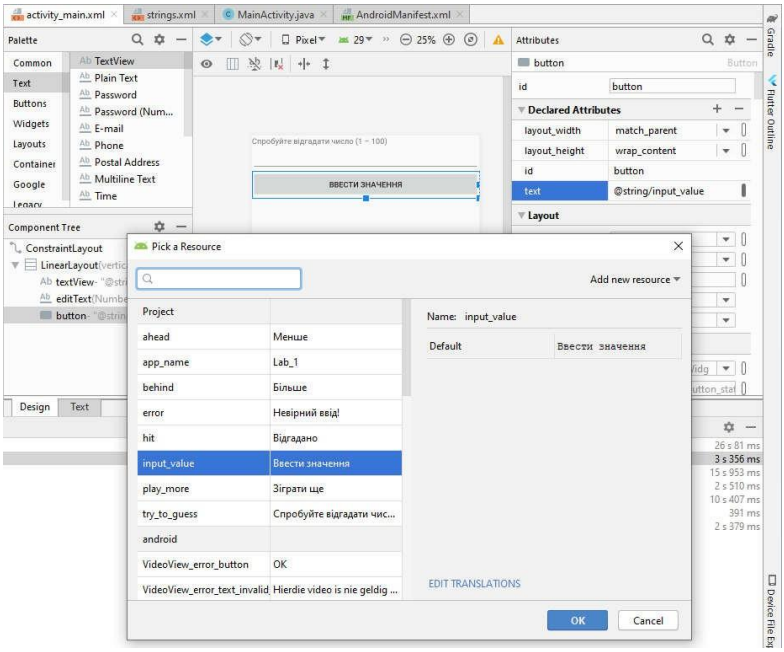

Рис.1.12 - Налаштування тексту для кнопки *button*

Настав час згадати про віртуальний пристрій, якщо додаток працює, вже можна запустити проект і подивитися, як додаток буде виглядати на екрані пристрою, а виглядати він може як показано на рис. 1.13.

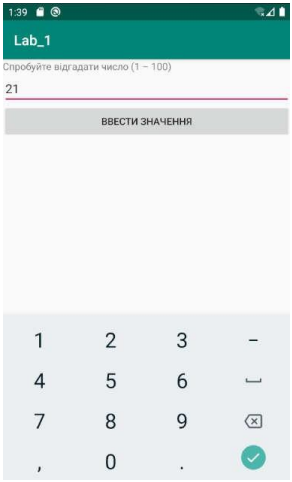

Рис. 1.13 - Запуск програми на реальному пристрої

Додаток виглядає досить просто, але ми на багато і не розраховували. Головне, що нас цікавить, це наявність всіх елементів на екрані, вірний текст в кожному елементі, де він передбачений і можливість вводити числа в полі введення. На рис. 1.13 видно, що всі вимоги виконані. Додаток є, його можна запустити на віртуальному чи реальному пристрої, але він нічого не робить. Наступним кроком буде реалізація логіки додатку, тобто обробка події натискання на кнопку, як було прописано в завданні.

1.4.3. Реалізація логіки додатку

Приступимо безпосередньо до програмування, працювати будемо з файлом *java/com.example.lab_1/MainActivity.java*. Знайдемо цей файл в *Project Explorer* див. Рис. 1.14, відкриємо і почнемо редагувати.

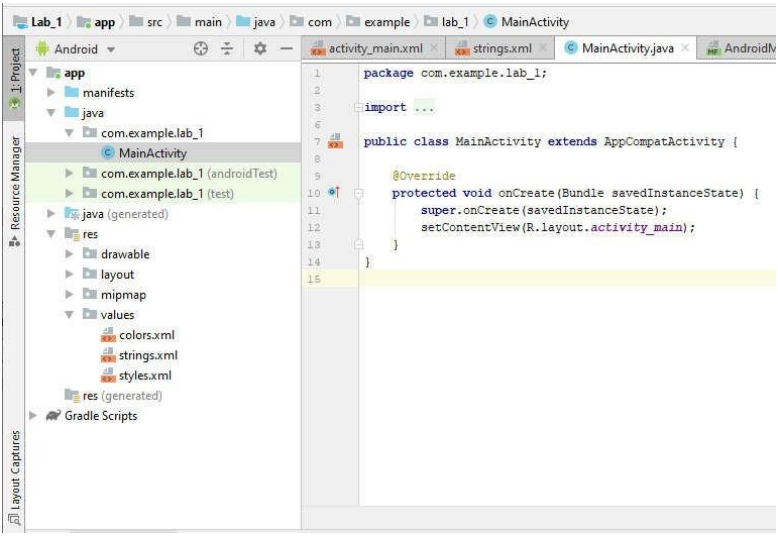

Рис. 1.14 - Файл *MainActivity.java*

Можна помітити, що клас *MainActivity* є спадкоємцем класу *Activity* і в ньому вже реалізований метод *onCreate()*, який запускається при початковому створенні активності, нам буде потрібно його доповнити, але про це трохи пізніше.

Ми припускаємо програмно змінювати інформацію в поле *TextView*, отримувати значення з поля *EditText* і обробляти події натискання на кнопку *Button*, тому необхідно оголосити відповідні змінні, як поля класу *MainActivity*:

- *TextView tvInfo;*
- *EditText etInput;*
- *Button bControl;*

Як можна побачити на рис.1.15, дані класи виділені червоним кольором, це тому що їх немає в проекті і необхідно додати відповідні бібліотеки для їх використання, проте якщо навести курсор мишки на назву класу виводиться підказка, що можна поставити курсор на цю назву і відповідна бібліотека буде додана до проекту, це стосується стандартних бібліотек, при використанні сторонніх чи власних бібліоте їх необхідно додавати вручну.

Рис.1.15 – Помилки при відсутності необхідних класів чи бібліотек

Тепер необхідно пов'язати ці змінні з елементами інтерфейсу, вже доданими нами в activity_main.xml, зробити це необхідно в методі *onCreate()*, а для отримання вже створеного елементу інтерфейсу скористаємося методом *findViewById()*. Отже в метод *onCreate()* додамо наступні рядки:

tvInfo = (TextView)findViewById(R.id.textView1);
etInput = (EditText)findViewById(R.id.editText1);
bControl = (Button)findViewById(R.id.button1);

Метод *findViewById()* повертає об'єкт класу *View*, який є спільним предком для всіх компонентів для користувацького інтерфейсу, для того щоб уникнути можливих помилок в дужках перед викликом методу вказуємо до якого конкретно компонента необхідно звузити можливості об'єкту *View*.

Прийшов час виконати обробку натискання на кнопку. Повернемося до файлу *activity_main.xml* в графічний режим редагування, виберемо елемент *Button* і на вкладці з властивостями елементу знайдемо

властивість *OnClick* і запишемо в нього *onClick* - ім'я методу, який буде обробляти натискання на кнопку. Як це виглядає показує рис. 1.16.

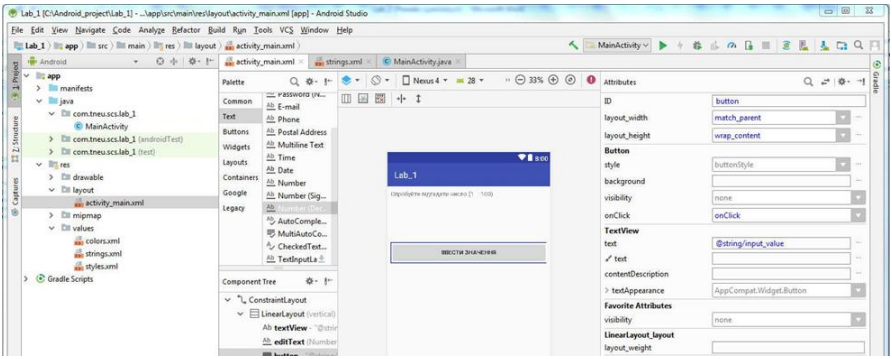

Рис. 1.16 - Налаштування властивості OnClick для кнопки

Ці ж дії можна виконати в файлі activity_main.xml, досить дописати виділений рядок:

```
<Button android:id="@+id/button"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:onClick="onClick"
android:text="@string/input_value"  />
```

Для налаштування властивостей елементів інтерфейсу потрібно використовувати будь- який спосіб: графічний або редагування XML файлу.

Повернемося в файл *MainActivity.java*, в клас активності необхідно додати метод: *public void onClick(View v){...}*

Ім'я методу не обов'язково має бути *onClick()*, головне, щоб воно збігалося з ім'ям, зазначеним у властивості *OnClick*. У цьому методі і буде відбуватися все наше програмування в цій лабораторній роботі.

Нам будуть потрібні дві змінні:

- цілочисельна для зберігання задуманого числа (випадкове число від 1 до 100);
- логічна для зберігання стану закінчена гра чи ні.

Обидві ці змінні слід оголосити як поля класу активності, початкові значення привласнити в методі *onCreate*.

Отримати цілочисельне значення з поля вводу, можна за допомогою наступної конструкції:

Integer.parseInt(etInput.getText().toString())

Змінити значення тексту в інформаційному полі можна за допомогою наступної конструкції:

tvInfo.setText(getResources().getString(R.string.ahead));

В даному випадку в інформаційному полі з'явиться значення строкового ресурсу з ім'ям *ahead*.

Залишилося реалізувати логіку додатку в методі *onClick()*. Пропонуємо написати код цього методу самостійно, для контролю в додатку запропонований лістинг, який містить один з варіантів коду описаного додатку.

Зрозуміло, що запропонована в додатку реалізація не ідеальна, вимагає доопрацювань:

- Що станеться, якщо кнопка буде натиснута до введення будь-якого числа? Швидше за все додаток буде зупинено, необхідно якось відстежувати цей варіант розвитку подій і адекватно реагувати. Пропонуємо попрацювати самостійно над цією проблемою.
- Що станеться, якщо користувач введе число менше нуля або більше 100? Швидше за все додаток обробить цей ввід, але було б краще, якби з'явилося повідомлення про те, що введене число не відповідає умовам завдання.
- Як завершити додаток? І чи треба це робити?

1.4.4. Робота з емулятором

При виконанні даної роботи ми використовували емулятор для перевірки працездатності програми. Розглянемо деякі можливості, що полегшують і прискорюють взаємодію з віртуальним пристроєм.

По-перше, існує набір корисних комбінацій клавіш для управління віртуальним пристроєм:

- *Alt + Enter* - розгортає емулятор до розмірів екрану;
- *Ctrl + F11* - змінює орієнтацію емулятора з портретної на альбомну і назад;
- *F8* - вмикає / вимикає мережу.

Повний список комбінацій клавіш для роботи з емулятором можна знайти за посиланням:

http://developer.android.com/tools/help/emulator.html.

По-друге, хто б не працював з емулятором, той на собі відчув наскільки терплячим треба бути, щоб взаємодіяти з ним, так повільно він працює. Існує рішення для прискорення роботи емулятора Android і цим рішенням є *Intel Hardware Accelerated Execution Manager (Intel® HAXM).*

Intel Hardware Accelerated Execution Manager (Intel® HAXM) - це додаток з підтримкою апаратної віртуалізації (гіпервізор), який використовує технологію віртуалізації Intel для прискорення емуляції додатків Android на комп'ютері для розробки.

(http://software.intel.com/ru-ru/android/articles/intel-hardware-accelerated-execution-manager)

Intel HAXM здатний прискорити роботу емулятора для x86 пристроїв. При цьому емулятор буде працювати зі швидкістю, наближеною до швидкості роботи реального пристрою, що допоможе скоротити час на запуск і налагодження програми. Детально ознайомитися з установкою *Intel HAXM* і налаштуванням емулятора на роботу з прискорювачем можна в статті за посиланням: *http://habrahabr.ru/company/intel/blog/146114/*

По-третє, не варто переривати процес запуску віртуального пристрою, наберіться терпіння, на старих комп'ютерах перший запуск *AVD* може зайняти до 10 хвилин, на сучасних - від однієї до трьох хвилин. Після того, як створили і запустили віртуальний пристрій є сенс залишити його відкритим, при всіх наступних запусках програми буде використовуватися вже відкритий віртуальний пристрій, що дозволить заощадити час.

1.4.5. Лістинг

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.*;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {
    TextView tvInfo;
    EditText etInput;
    Button bControl;

    int guess;
    boolean gameFinished;

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

        tvInfo = (TextView)findViewById(R.id.textView);
        etInput = (EditText)findViewById(R.id.editText);
        bControl = (Button)findViewById(R.id.button);

        guess = (int)(Math.random()*100);
        gameFinished = false;
    }

    public void onClick(View v){
        if (!gameFinished){
            int inp=Integer.parseInt(etInput.getText().toString());

            if (inp > guess)
                tvInfo.setText(getResources().getString(R.string.ahead));
            if (inp < guess)
                tvInfo.setText(getResources().getString(R.string.behind));
            if (inp == guess)
            {
                tvInfo.setText(getResources().getString(R.string.hit));
                bControl.setText(getResources().getString(R.string.play_more));
                gameFinished = true;
            }
        }
        else
        {
            guess = (int)(Math.random()*100);
            bControl.setText(getResources().getString(R.string.input_value));
            tvInfo.setText(getResources().getString(R.string.try_to_guess));
            gameFinished = false;
        }
        etInput.setText("");
    }

}
```

1.5. Контрольні запитання

1. Що таке Android Studio?
2. Особливості Android Studio.
3. Охарактеризуйте алгоритм створення проектів в Android Studio.
4. Яка папка описує властивості проекту?
5. Яким чином присвоюється ім'я пакету?
6. За що відповідає папка res?
7. Які компоненти містить папка res?
8. Що таке файл маніфесту? Що містить файл маніфесту?
9. Які особливості файлу маніфесту?
10. Алгоритм налаштування інтерфейсу додатку

Список літературних джерел

1. Android Studio provides the fastest tools for building apps on every type of Android device: веб сайт. URL: https://developer.android.com/studio
2. Андрей Рожков. Андроид Студио. Шаг первый.: Android Studio. The first step: підручник, 2014. 200 с.
3. Clifton Craig, Adam Gerber. Learn Android Studio: Build Android Apps Quickly and Effectively: підручник. Видавництво Apress, 2015 р. 484 с.
4. Kyle Mew. Mastering Android Studio 3: підручник. Видавництво Packt Publishing Ltd, 2017 р. 220 с.
5. Neil Smyth. Android Studio 3.4 Development Essentials - Java Edition: підручник. Видавництво eBookFrenzy, 2019 р. 780 с.
6. Neil Smyth. Android Studio Arctic Fox Essentials - Java Edition: Developing Android Apps Using Android Studio 2020.31 and Java: підручник. Видавництво eBookFrenzy, 2021 р. 778 с.
7. Пирская Л. Розробка мобільних додатків в середовищі Android Studio : підручник. Видавництво Litres, 2021. 123 с.
8. Android Studio: веб сайт. URL: https://en.wikipedia.org/wiki/Android_Studio