

## ЛАБОРАТОРНА РОБОТА 7

**Тема:** Застосування Java у мобільних застосунках Android

### Теоретична частина

Програми Android будуються на базі архітектури, яка називається «Модель-Вид-Контролер» (Model-View-Controller, MVC). Відповідно до принципів цієї архітектури кожен об'єкт докладання має бути об'єктом моделі, об'єктом уявлення чи об'єктом контролера.

- Об'єкт моделі містить дані програми та «бізнес-логіку». Класи моделі зазвичай проєктуються для моделювання сутностей, з якими працює додаток, - користувач, рахунок у банку, продукт у магазині, фотографія на сервері, питання «так/ні» тощо. Об'єкти моделі нічого не знають про інтерфейс користувача, їх єдиною метою є зберігання даних і управління ними. Класи моделей зазвичай створюються розробником для конкретної задачі, проте об'єкти моделі у додатку становлять його рівень моделі.
- Об'єкти уявлень відображаються на екрані та реагують на певні дії користувача, наприклад, торкання, фокусування та інше. Якщо щось виводиться на екран, це об'єкт подання. В Android є досить широкий набір класів уявлень, що настроюються. Крім того, розробник може створювати власні класи уявлень. Об'єкти подання у додатку утворюють рівень подання.
- Об'єкти контролерів пов'язують об'єкти уявлення та моделі, і містять «логіку застосування». Контролери реагують різні події, ініційовані об'єктами уявлень, і керують потоками даних між об'єктами моделі і рівнем представлення. В Android контролер зазвичай наслідує клас Activity, Fragment або Service.

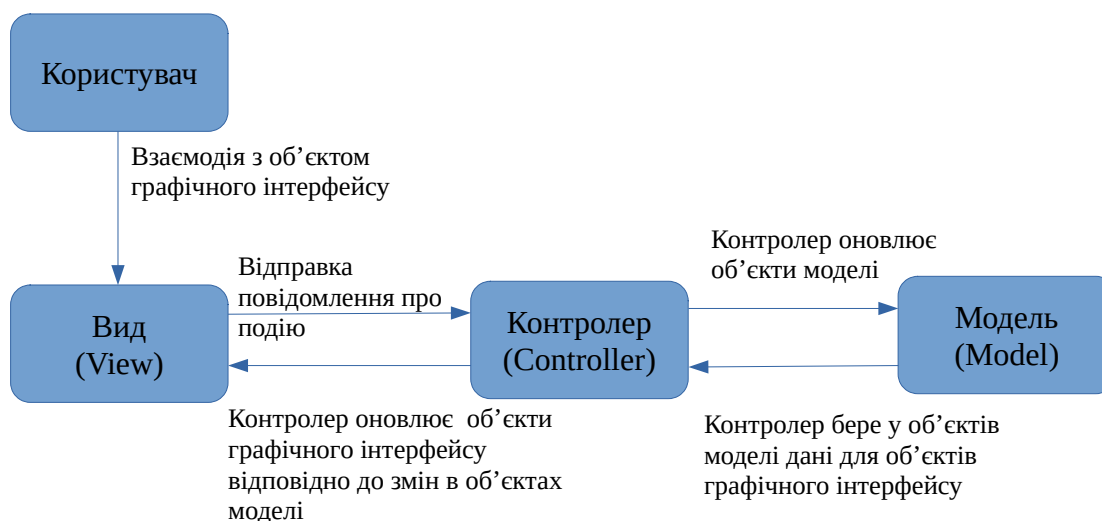


Рис.1. Взаємодія в MVC

Об'єкти моделі та виду не взаємодіють один з одним безпосередньо - у будь-якій взаємодії беруть участь «посередники»-контролери, які отримують повідомлення від одних об'єктів і передають інструкції іншим.

Функціональність програми може збільшуватися до тих пір, поки не стане надто складною для розуміння та супроводу. Поділ коду на класи спрощує проєктування та розуміння програми в цілому, дає можливість розробнику мислити в контексті класів, а не окремих змінних та методів. Аналогічним чином поділ класів на рівні моделі, уявлення та контролера спрощує проєктування та розуміння програми, і надає можливість мислити у контексті рівнів, а не окремих класів. Додавання або зміна класів виконується в контексті певного рівня та на відповідному рівні, що

покращує архітектуру та її розуміння, а отже подальше супроводження та можливі модифікації.

Також концепції MVC полегшують повторне використання класів. Клас з обмеженими обов'язками краще підходить для повторного використання, ніж клас, в якому реалізовано і модель даних, і всю логіку завдання, і подання для інтерфейсу користувача.

Розглянемо просте Android-додаток, що реалізує завдання опитування з відповідями «Так/Ні». Програма має простий інтерфейс користувача, на якому подається питання і кілька кнопок, які дозволяють дати відповідь «Так/Ні» («Правильно/Неправильно») на питання і зробити перехід між питаннями.

Додаток складається з активності (activity) та макета (layout):

- Активність представлена екземпляром Activity – класом з Android SDK. Вона відповідає за взаємодію користувача з інформацією на екрані. Щоб реалізувати необхідну програму функціональності створюються підкласи Activity. У простому додатку може бути достатньо одного підкласу, а для складного може знадобитися кілька.
- Активність представлена примірником Activity – класом з Android SDK. Вона відповідає за взаємодію користувача з інформацією на екрані. Щоб реалізувати потрібну програму функціональності створюються підкласи Activity. У простому додатку може бути достатньо одного підкласу, а для складного може знадобитися кілька.

На сьогоднішній день офіційною системою розробки Android-додатків є Android Studio. Запустіть Android Studio, створіть новий проект та задайте йому ім'я. Вибір формфактора (Phone and Tablet) залишаємо за замовчуванням. Обираємо тип проект “Empty Views Activity”.

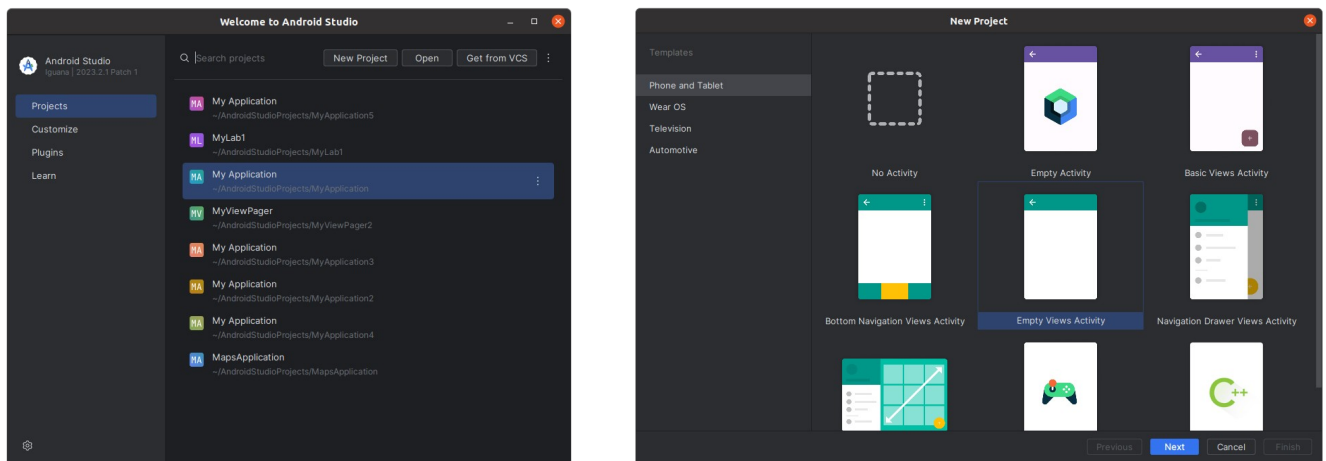


Рис.2. Початок роботи в Android Studio. Обираємо проект “Empty Views Activity”

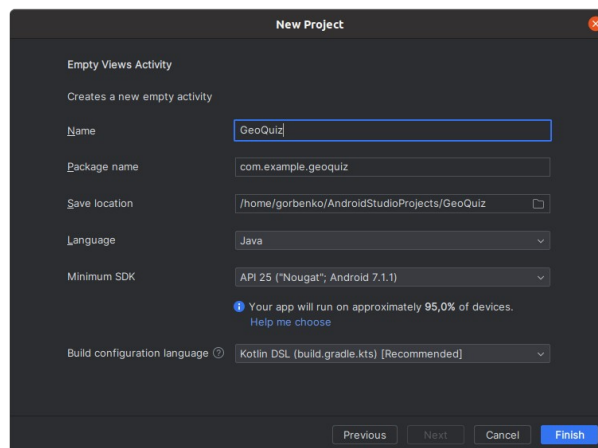


Рис.3. Новий проект в Android Studio з ім'ям програми GeoQuiz

У наступному вікні діалогу задаємо ім'я проекту GeoQuiz, контролюємо або змінюємо ім'я пакету та місце розташування проекту. Обираємо мову програмування для проекту як Java. Значення Minimum SDK або залишаємо за замовчуванням, або встановлюємо у відповідності до наявних SDK в студії. Після натискання кнопки Finish створюються необхідні файли та формується проект. До складу проекту входить значна кількість файлів, частину яких можна побачити за допомогою навігаційної панелі (за замовчуванням знаходиться ліворуч).

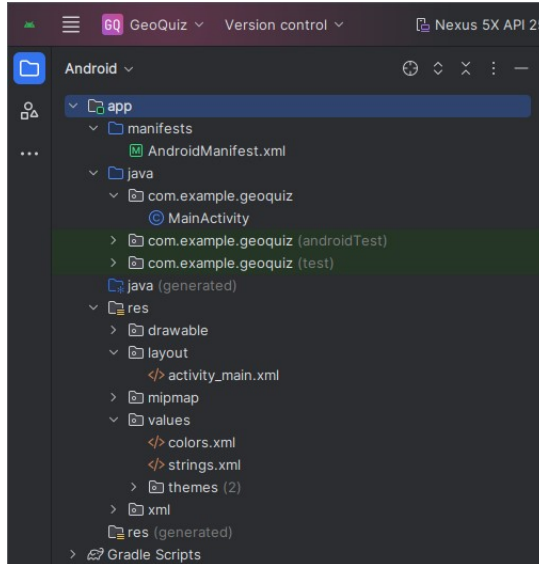


Рис.4. Панель навігації проекту в Android Studio

Насправді навіть початкова структура простого проекту Android-програми набагато складніша, достатньо переключити навігацію в режим перегляду Project

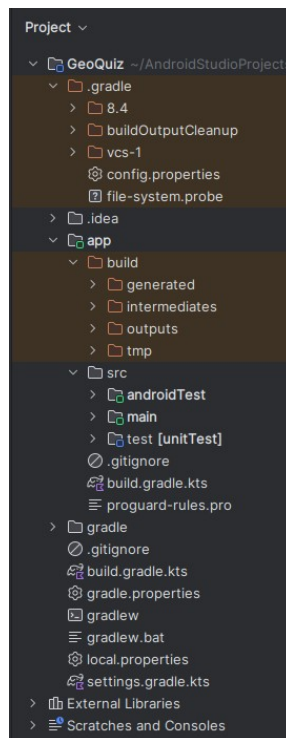


Рис.5. Навігація проекту в режимі Project

За замовчуванням новий проект відкривається в режимі навігації Android, а в панелі редактора відкриваються два файли - у проекті MainActivity.java та activity\_main.xml. У контексті моделі MVC activity\_main.xml відноситься до рівня представлення, а MainActivity.java – до рівня

контролера.

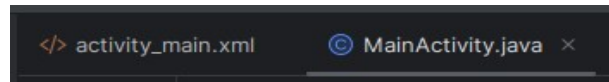


Рис.6. У редакторі за замовчуванням відкриваються два файли проекту

Редагування цих файлів дозволяє внести необхідні зміни в інтерфейс користувача і логіку обробки дій користувача. Рівень моделі даних реалізується додатковими класами, які створюються та додаються у проект за потреби. Наприклад, для того щоб у проекті можна було оперувати із деякою вибіркою питань потрібно створити додатковий клас, наприклад, Question. Для цього на панелі навігації знаходимо пакет, в якому знаходиться файл MainActivity.java (рис.7) і додаємо в нього новий клас (File -> New -> Java Class) (рис.8).

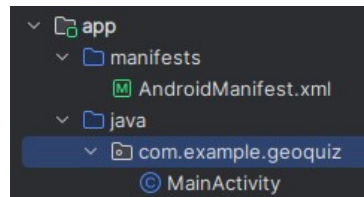


Рис.7. У цей пакет буде додано файл із класом моделі даних

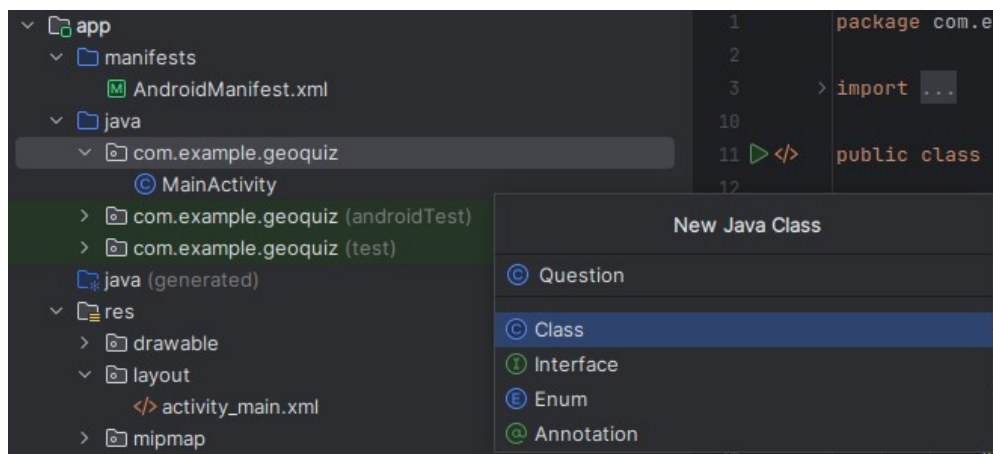


Рис.8. Створення нового класу Question

Після натискання кнопки Ок створюється порожній клас. Цей клас у проекті має дозволяти створювати об'єкти, що містять запитання та відповіді на них. Тому в простому варіанті клас повинен містити два поля: текстове для питання та логічне для відповіді (true або false). У додатках Android прийнято текстові значення розміщувати у файлах ресурсів, наприклад, res/values/strings.xml. Кожен елемент у файлі strings.xml отримує свій числовий ідентифікатор, тому звернення до текстового ресурсу в проекті здійснюється за його ідентифікатором, а його конкретне значення зберігається у файлі ресурсів. Такий механізм зберігання та використання текстових даних дозволяє здійснювати їх мовну локалізацію чи коригування значення без зміни програмного коду проекту. Таким чином у клас Question необхідно додати числове поле для ідентифікатору текстового значення, логічне поле для відповіді та конструктор:

```
public class Question {  
    private int mTextResId;  
    private boolean mAnswerTrue;  
    public Question(int textResId, boolean answerTrue) {  
        mTextResId = textResId;  
        mAnswerTrue = answerTrue;  
    }  
}
```

Також, для роботи з полями в клас додаємо get та set методи:

```

public class Question {
    . . . . .
    public int getTextResId() {
        return mTextResId;
    }
    public void setTextResId(int textResId) {
        mTextResId = textResId;
    }
    public boolean isAnswerTrue() {
        return mAnswerTrue;
    }
    public void setAnswerTrue(boolean answerTrue) {
        mAnswerTrue = answerTrue;
    }
}

```

Тепер у проєкті є всі три складові шаблону MVC.

Рівень представлення створюваного проєкту визначається макетом інтерфейсу користувача, опис якого знаходиться у файлі `activity_main.xml`. Структурні елементи, з яких складається інтерфейс користувача, називають віджетами. Віджет може виводити текст або графіку, взаємодіяти з користувачем або розміщувати інші віджети на екрані. Різновидами віджетів є кнопки, текстові поля, прапорці тощо. Крім того є віджети, які не мають видимого графічного представлення, але використовуються для організації розташування видимих віджетів. Android SDK включає багато віджетів, які можна налаштовувати для отримання потрібного оформлення та поведінки. Кожен віджет є екземпляром класу `View` або одного з його підкласів, таких як `TextView` або `Button`.

Для прискорення процесу розробки інтерфейсу користувача в `Android Studio` є як редактор тексту файлу макета, так і візуальний редактор інтерфейсу.



Між редакторами можна перемикатися при розробці інтерфейсу користувача, а будь-які зміни в текстовій частині призводять до змін у створюваному інтерфейсі і навпаки - будь-які маніпуляції з графічними уявленнями віджетів призводять до відповідних змін в текстовій частині файлу макета.

Для створюваної програми файл `activity_main.xml` може бути реалізований таким чином:

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">
        <TextView
            android:id="@+id/TextQuiz"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="14sp" />
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>

```

```

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <Button
        android:id="@+id/buttonTrue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@string/button_true" />
    <Button
        android:id="@+id/buttonFalse"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@string/button_false" />
</LinearLayout>
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <Button
        android:id="@+id/buttonPrev"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@string/button_prev" />
    <Button
        android:id="@+id/buttonNext"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@string/button_next" />
</LinearLayout>
</LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>

```

Віджети `LinearLayout` дозволяють організувати ієрархію в макеті та згрупувати вкладені віджети. Їхній параметр `orientation` дозволяє вказати розміщення віджетів у групі або вказати ієрархію структури. Інші віджети у цьому файлі - `TextView` та `Button` описують конкретні елементи графічного інтерфейсу користувача. Написи на кнопках визначаються за допомогою ідентифікаторів тексту, наприклад `@string/button_next`, значення якого задано у файлі ресурсів `res/values/strings.xml`. Приклад змісту цього файлу наведено нижче

```

<resources>
    <string name="app_name">GeoQuiz</string>
    <string name="question_1">Столиця Туреччини - Константинополь</string>
    <string name="question_2">Тихий океан більший за Атлантичний</string>
    <string name="question_3">Суецький канал поєднує Червоне море\п з Індійським океаном</string>
    <string name="question_4">Ріка Нил починається в Єгипті</string>
    <string name="button_true">Це правда</string>
    <string name="button_false">Це брехня</string>
    <string name="button_next">Наступний</string>
    <string name="button_prev">Попередній</string>
    <string name="toast_true">Ви відповіли правильно! </string>
    <string name="toast_false">Ви відповіли неправильно!</string>
</resources>

```

Цей файл також містить запитання та повідомлення про правильність відповідей.

Згідно MVC зв'язування моделі даних і представлень здійснюється за допомогою логіки контролера, яка, у проекті, що створюється, представляється у файлі `MainActivity.java`. Його зміст може бути наступним

```

package com.example.geoquiz;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

```

```

private Button mTrueButton;
private Button mFalseButton;
private Button mNextButton;
private Button mPrevButton;
private TextView mTextQuiz;
private static final String TAG="MainActivity";
private static final String KEY_INDEX = "index";

private Question[] mQuestionBank = new Question[]{
    new Question(R.string.question_1, false),
    new Question(R.string.question_2, true),
    new Question(R.string.question_3, false),
    new Question(R.string.question_4, false)
};
private int mCurrentIndex = 0;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mTextQuiz = (TextView)findViewById(R.id.TextQuiz);
    updateQuestion();
    addListenerOnClickButtonTrueFalse();
}

public void addListenerOnClickButtonTrueFalse(){
    mTrueButton = (Button)findViewById(R.id.buttonTrue);
    mFalseButton = (Button)findViewById(R.id.buttonFalse);
    mNextButton = (Button)findViewById(R.id.buttonNext);
    mPrevButton = (Button)findViewById(R.id.buttonPrev);
    mTextQuiz = (TextView)findViewById(R.id.TextQuiz);

    mTrueButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            checkAnswer(true);
        }
    });

    mFalseButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            checkAnswer(false);
        }
    });

    mNextButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            mCurrentIndex = (mCurrentIndex + 1) % mQuestionBank.length;
            updateQuestion();
        }
    });

    mPrevButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            mCurrentIndex = (mCurrentIndex - 1) % mQuestionBank.length;
            if(mCurrentIndex<0)mCurrentIndex=mQuestionBank.length-1;
            updateQuestion();
        }
    });
}

private void updateQuestion(){
    int question = mQuestionBank[mCurrentIndex].getTextResId();
    mTextQuiz.setText(question);
}

private void checkAnswer(boolean userPressedTrue){
    boolean answerIsTrue = mQuestionBank[mCurrentIndex].isAnswerTrue();
    int messageResId = 0;
    if(userPressedTrue == answerIsTrue){
        messageResId = R.string.toast_true;
    }else{
        messageResId = R.string.toast_false;
    }
    Toast.makeText(this,messageResId, Toast.LENGTH_SHORT).show();
}
}
}

```

Метод onCreate(Bundle savedInstanceState) викликається при створенні Activity , тому все, що знаходиться в цьому методі буде виконано при запуску програми. А саме два методи updateQuestion() та addListenerOnClickButtonTrueFalse() дозволяють вивести перше запитання та забезпечити функціональність кнопки інтерфейсу: відповіді на запитання «Правильно/Неправильно» та зміну питань у двох напрямках «Наступний/Попередній».

### **Практичне завдання**

1. Вивчіть наведений вище додаток, реалізуйте його в Android Studio і перевірте його роботу.
2. Розширте базу питань до 10.
3. Розширте функціональність програми додаванням підрахунку балів за правильні відповіді на запитання.
4. Підготуйте звіт за всіма пунктами практичного завдання.

### **Посилання**

1. Android Studio // <https://developer.android.com/studio/>