ЛАБОРАТОРНА РОБОТА 1

Тема: Використання моделі MVC в Android застосунках

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 проектуються 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.

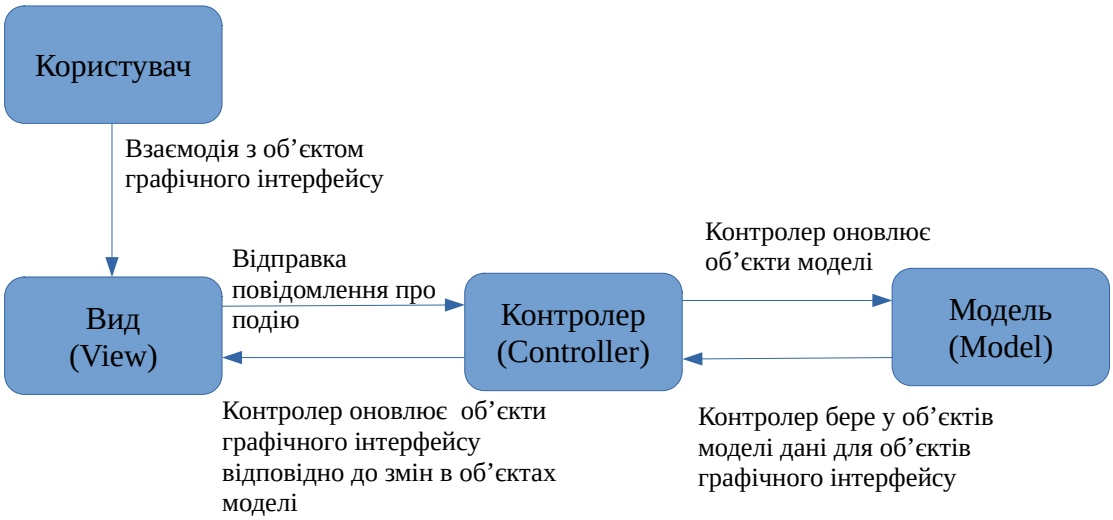

Рис.1. Взаємодія в MVC

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 проектування 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 проектування 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 функціональність 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 функціональність 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.

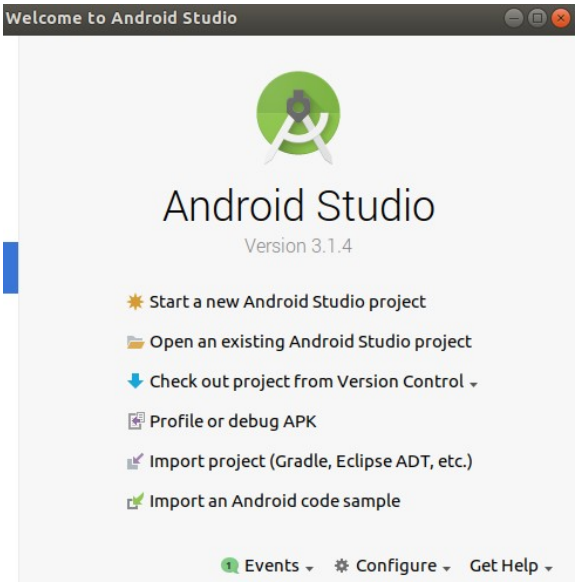

Рис.2. Початок роботи в Android Studio

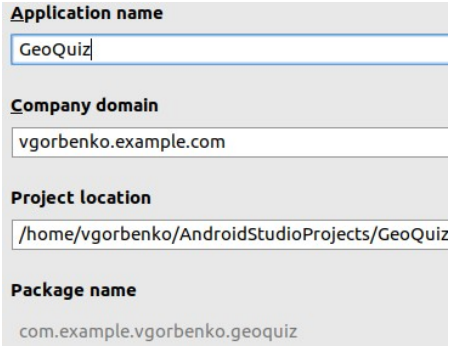

Рис.3. Новий проект в Android Studio з ім'ям програми GeoQuiz

Вибір формфактора (Phone and Tablet) та мінімального SDK можна залишити за замовчуванням. У наступному вікні діалогу «Додавання до проекту Activity» робимо вибір

шаблону EmptyActivity. У вікні конфігурування Activity задаємо імена Activity Name та Layout Name, які взаємопов'язані між собою частиною іменування. Наприклад, задавши в Activity Name ім'я QuizActivity у полі Layout Name, отримаємо activity_quiz.xml. Установку прапорців залишаємо за замовчуванням. Після натискання кнопки Finish Android Studio створює та організує проект. До складу проекту входить значна кількість файлів, частину яких можна побачити за допомогою навігаційної панелі (за замовчуванням знаходиться ліворуч).

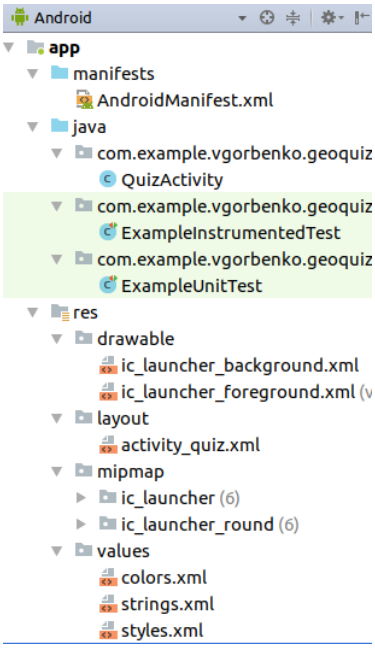

Рис.4. Панель навігації проекту в Android Studio

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

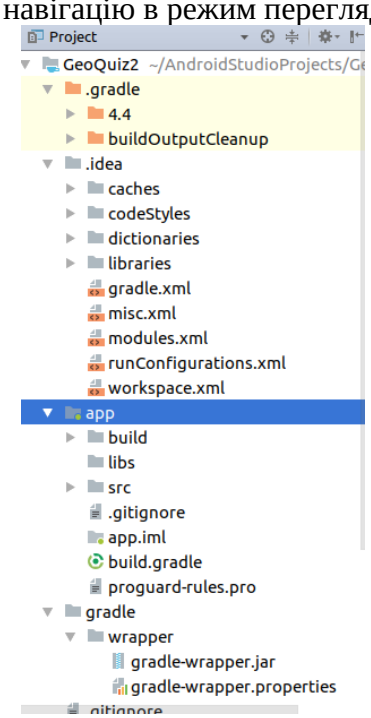

Рис.5. Навігація проекту в режимі Project

За замовчуванням новий проект відкривається в режимі навігації Android, а в панелі редактора відкриваються два файли - у проекті QuizActivity.java і activity_quiz.xml. У контексті моделі MVC activity_quiz.xml відноситься до рівня представлення, а QuizActivity.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 у міру потреби. Наприклад, щоб у проекті можна було оперувати з деякою вибіркою питань потрібно створити додатковий клас, наприклад, Question. Для цього на панелі навігації знаходимо пакет, в якому знаходиться файл QuizActivity.java (рис.7) і додаємо в нього новий клас (File -> New -> Java Class)

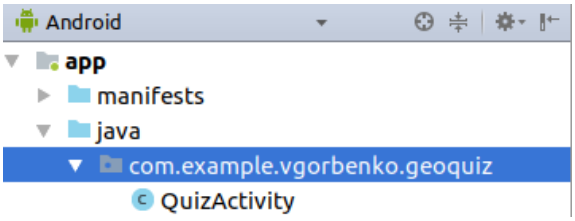

Рис.7. У цей пакет буде додано файл із класом моделі даних

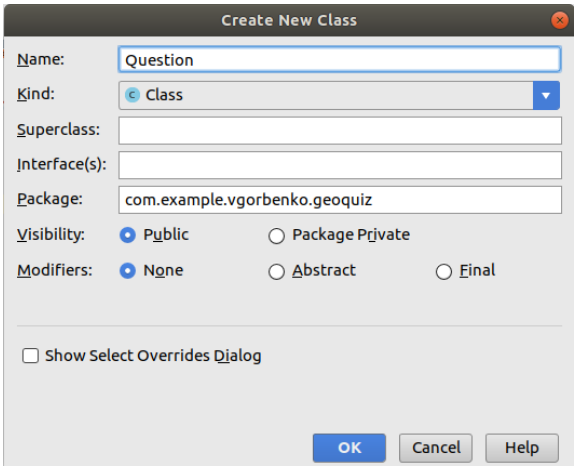

Рис.8. Створення нового класу Question

Після натискання кнопки Ок створюється порожній клас. Очевидно, що цей клас у проекті має дозволяти створювати об'єкти, в яких будуть запитання та відповіді на них. Тому в простому варіанті клас повинен містити два поля - рядкове (для питання) та логічне (для відповіді). Однак, в Android-додатках прийнято рядкові значення містити у файлах ресурсів, наприклад, res/values/strings.xml. Кожен елемент у цьому файлі отримає свій числовий ідентифікатор. Тому звернення до рядкового ресурсу в рамках проекту здійснюватиметься за його ідентифікатором, а його конкретне значення зберігатиметься у певному файлі ресурсів. Такий механізм зберігання та використання рядкових даних дозволяє без зміни програмного коду проекту здійснювати його локалізацію чи коригування рядкових значень. Тому в клас Question додаються числове та логічне поля та конструктор, як показано нижче

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

Для роботи з полями в клас необхідно додати get-і 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 проекті є всі три складові MVC.

Рівень представлення створюваного проекту визначається макетом інтерфейсу користувача, опис якого знаходиться у файлі activity_quiz.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. Кнопки, текстові поля, прапорці тощо. представляють різновиди віджетів. Крім того є віджети, які не мають видимого графічного уявлення, 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 в процесі розробки 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 коректного відображення графічного представлення макету необхідно перевірити правильність посилання в ієрархії класів у res/values/styles.xml. Вона має виглядати як

```
<!-- Base application theme. →
<style name="AppTheme" parent="Base.Theme.AppCompat.Light.DarkActionBar">
```

Якщо на панелі Preview не відображаються віджети макета, слід відкоригувати значення поля `parent`, як показано вище.

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

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".QuizActivity">

    <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
            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>
</android.support.constraint.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">Суецькій канал поєднує Червоне море\n 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 уявлень здійснюється за допомогою логіки контролера, яка, у проекті, що створюється, представляється у файлі QuizActivity.java. Його зміст може бути наступним

```
package com.example.vgorbenko.geoquiz;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
```

```
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

    public class QuizActivity extends AppCompatActivity {
        private Button mTrueButton;
        private Button mFalseButton;
        private Button mNextButton;
        private Button mPrevButton;
        private TextView mTextQuiz;
        private static final String TAG="QuizActivity";
        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_quiz);
            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 функціональністю 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. Внесіть зміни до програми так, щоб і формулювання питань, і відповіді на них витягувалися з файлу ресурсів strings.xml.
3. Змініть спливаюче повідомлення так, щоб воно відображалося у верхній, а не у нижній частині екрана. Для зміни способу відображення сповіщення використовуйте метод setGravity класу Toast. Виберіть режим Gravity.TOP. Додаткова інформація міститься в документації розробника:
developer.android.com/reference/android/widget/Toast.html#setGravity(int, int, int).
4. Внесіть зміни до програми так, щоб після відповіді на запитання замість формулювання питання виводилася відповідь на нього. При цьому, якщо відповідь користувачеві дали правильно, то відповідь повинна виводитися зеленим кольором, а якщо відповідь дана неправильна — червоним.
5. Порівняйте роботу програми на емуляторі та реальному пристрої.
6. Змініть тип віджетів кнопок на графічний ImageButton (замість Button). Віджет ImageButton є похідним від ImageView, на відміну від віджету Button, похідного від TextView. Атрибут text кнопки НАСТУПНИЙ тепер потрібно замінити на атрибут src ImageView із зазначенням джерела графічного представлення:

```
<Button ImageButton
   android:id="@+id/next_button"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:src="@drawable/arrow_right"
/>
```

Графічні файли розміщуються у відповідний каталог проекту res/drawable з ім'ям, яке використовуватиметься для посилання. Файли можуть мати розширення .png, .jpg або .gif. Будь-якому файлу, доданому у теку res/drawable, автоматично призначається ідентифікатор ресурсу. Імена файлів повинні бути записані в нижньому регістрі та не можуть містити пробілів.

7. Розширте функціональність програми додаванням підрахунку балів за правильні відповіді на запитання. Внесіть зміни до коду так, щоб запитання, відповідь на які вже було дано, не відображалися під час прокручування вперед/назад.
8. Розширте базу питань та зробіть їх вибірку для опитування випадковою та обмеженою за кількістю.
9. Підготуйте звіт за всіма пунктами практичного завдання.

Література

1. Phillips B. Android Programming: The Big Nerd Ranch Guide / B. Phillips, Ch. Stewart, K. Marsicano: 3rd Edition. — Big Nerd Ranch, 2017. — 720 p.
2. Android Studio // https://developer.android.com/studio/