Bộ môn An toàn Thông tin – Khoa MMT&TT
Trường Đại học Công nghệ Thông tin (UIT)

BÁO CÁO THỰC HÀNH

Môn học: Bảo mật web và ứng dụng

Kỳ báo cáo: Buổi 5 (Session 5)

Tên chủ đề: Basic Android Secure Programming

GVHD: Nghi Hoàng Khoa

Ngày báo cáo: 1/4/2020

Nhóm: 1

1. THÔNG TIN CHUNG:
   Lớp: NT213.L21.ANTN

STT	Họ và tên	MSSV	Email
1	Đoàn Thanh Phương	18521267	18521267@gm.uit.edu.vn
2	Phạm Ngọc Tâm	18521371	18521371@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá
1	Yêu cầu 2	100%
2	Yêu cầu 3	100%
3	Yêu cầu 4	100%
4	Yêu cầu 5	100%
5	Yêu cầu 6	100%

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

[1] Ghi nội dung công việc, các kịch bản trong bài Thực hành

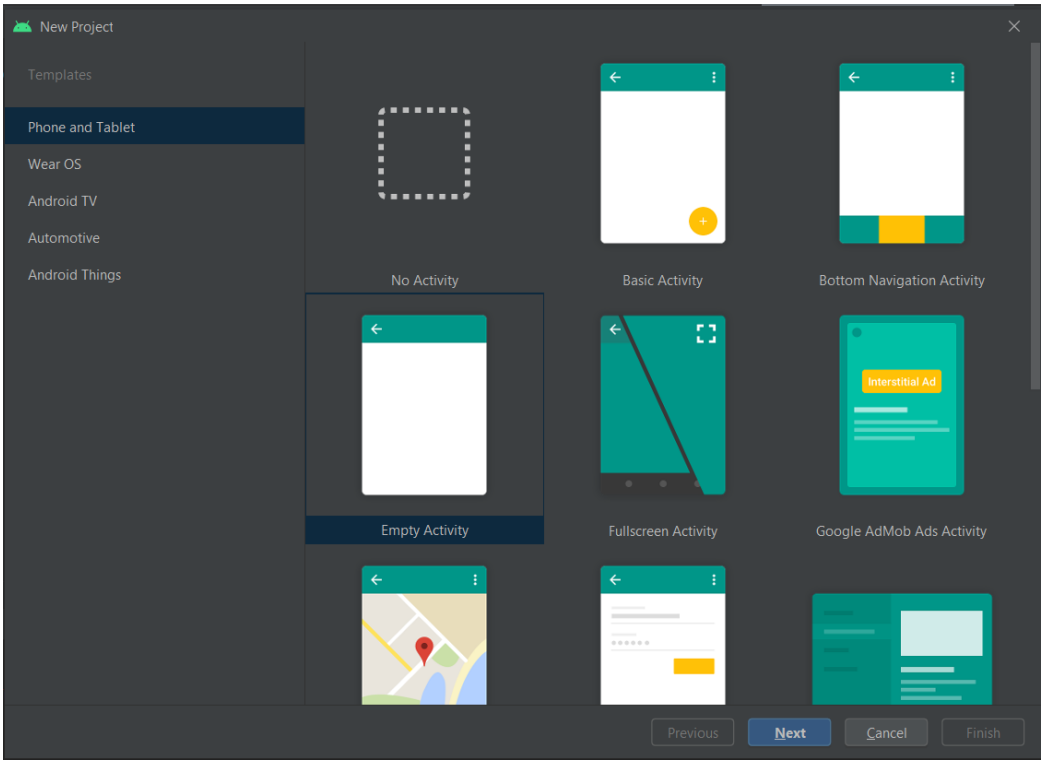
BÁO CÁO CHI TIẾT

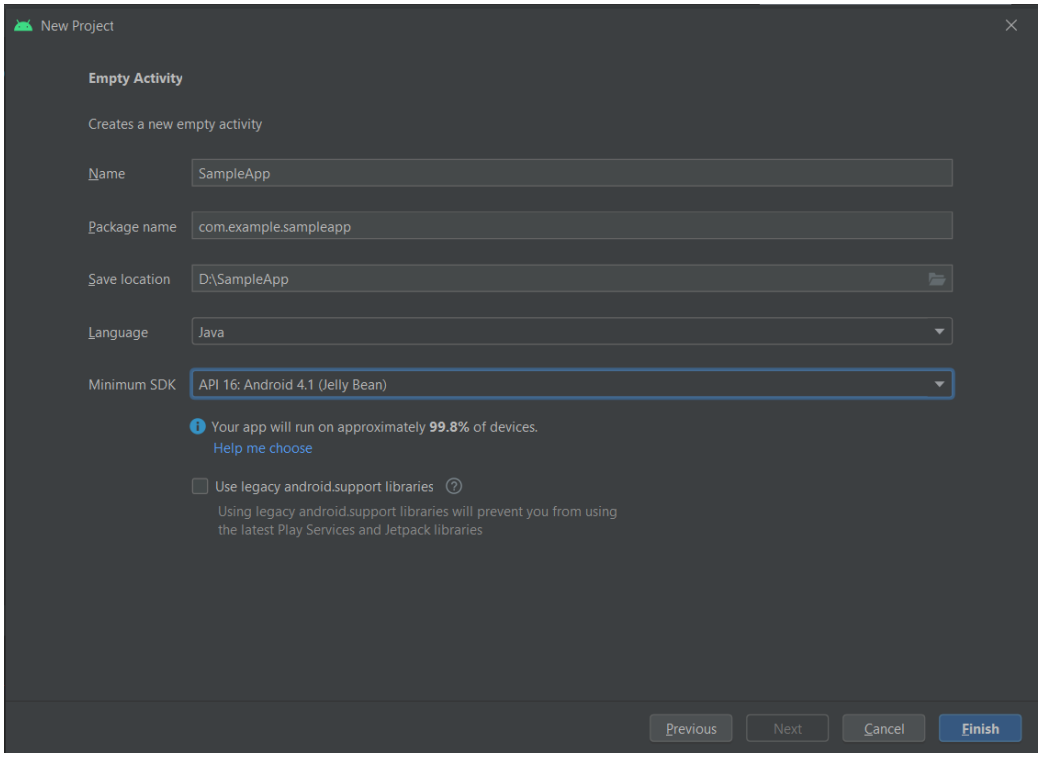
1. Yêu cầu 1,2,3,4

> Sinh viên xây dựng ứng dụng Android gồm 3 giao diện chức năng chính:
>
> 1) Register - Đăng ký thông tin với ứng dụng (email, username, password).
>
> 2) Login - Đăng nhập username, password).
>
> 3) Hiển thị thông tin người dùng (một lời chào có tên người dùng).

Bước 1: Tạo project

Thực hiện theo hướng dẫn bài lab

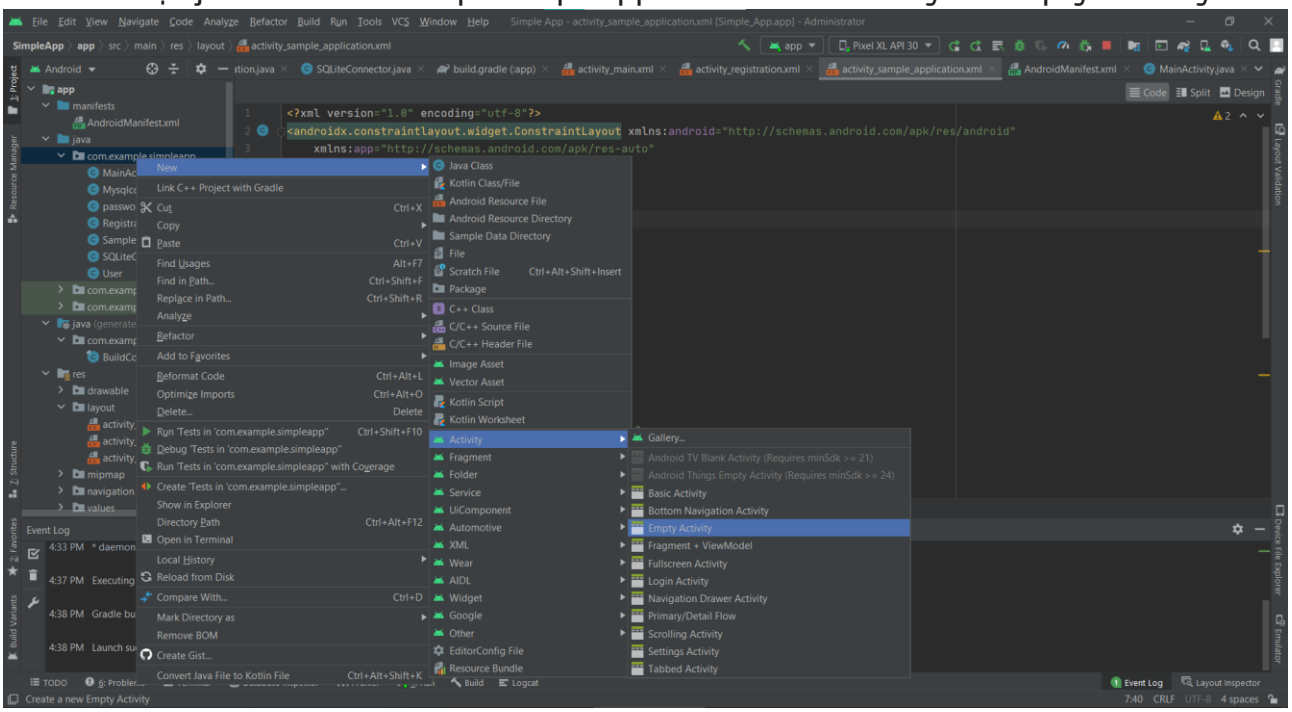


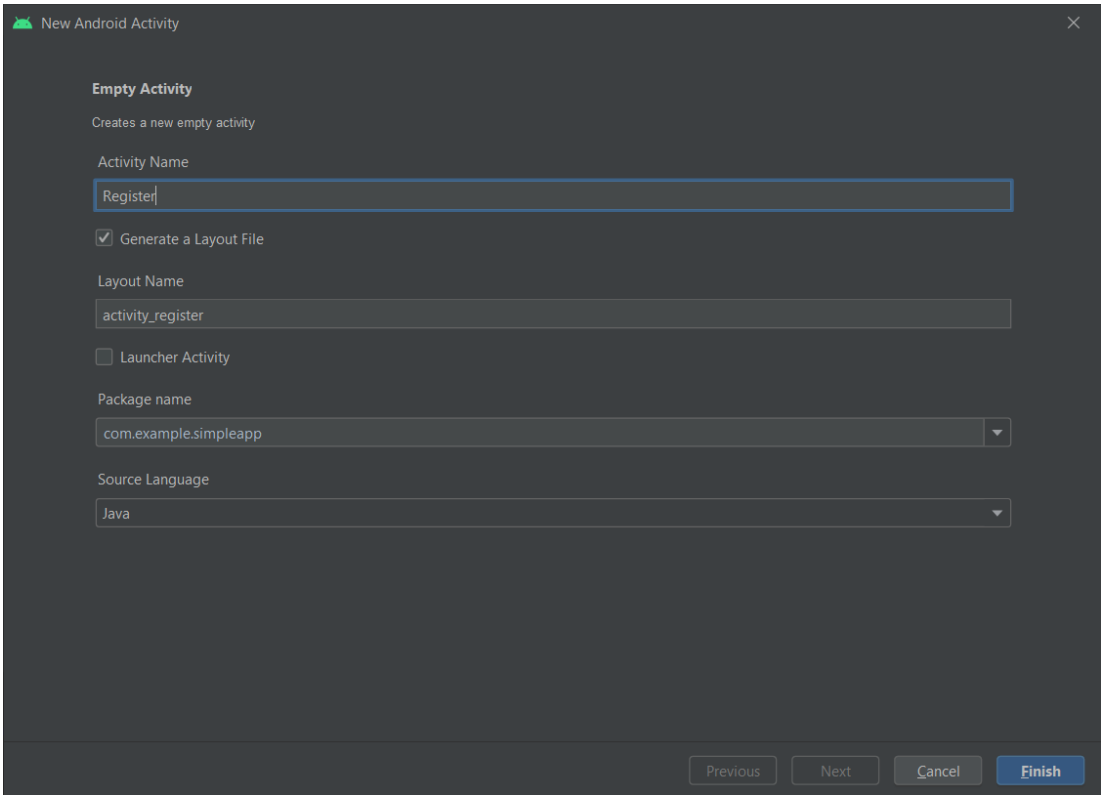


Bước 2: Tạo các activity

- Sử dụng các main_activity để làm trang login
- Tạo thêm 2 activity tương ứng với 2 chức năng Register và Welcome

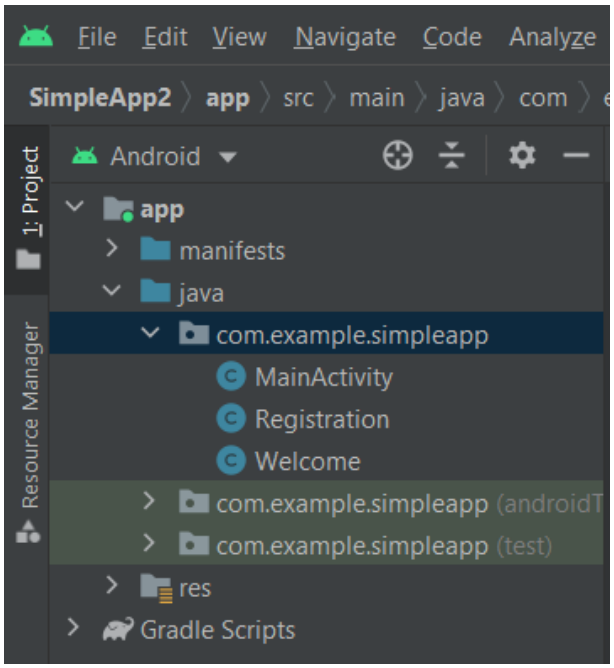
Vào thư mục java ➔ com.example.simpleapp ➔ New ➔ Activity ➔ Empty Activity





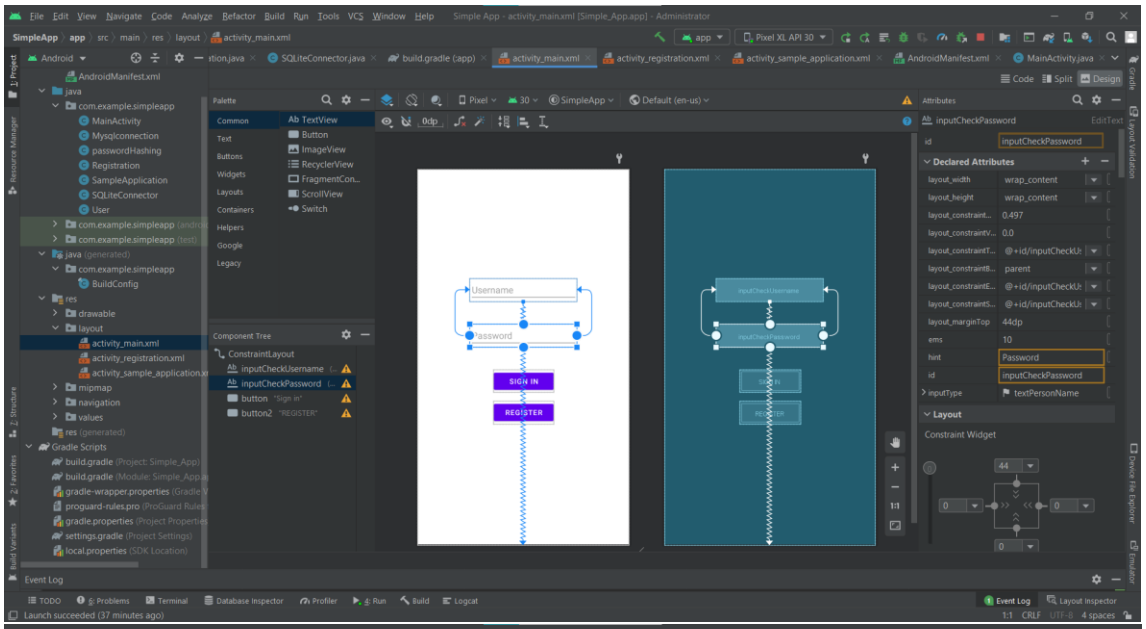
- Đặt tên cho Activity sau đó Finish
- Tương tự sẽ tạo thêm cho Welcome

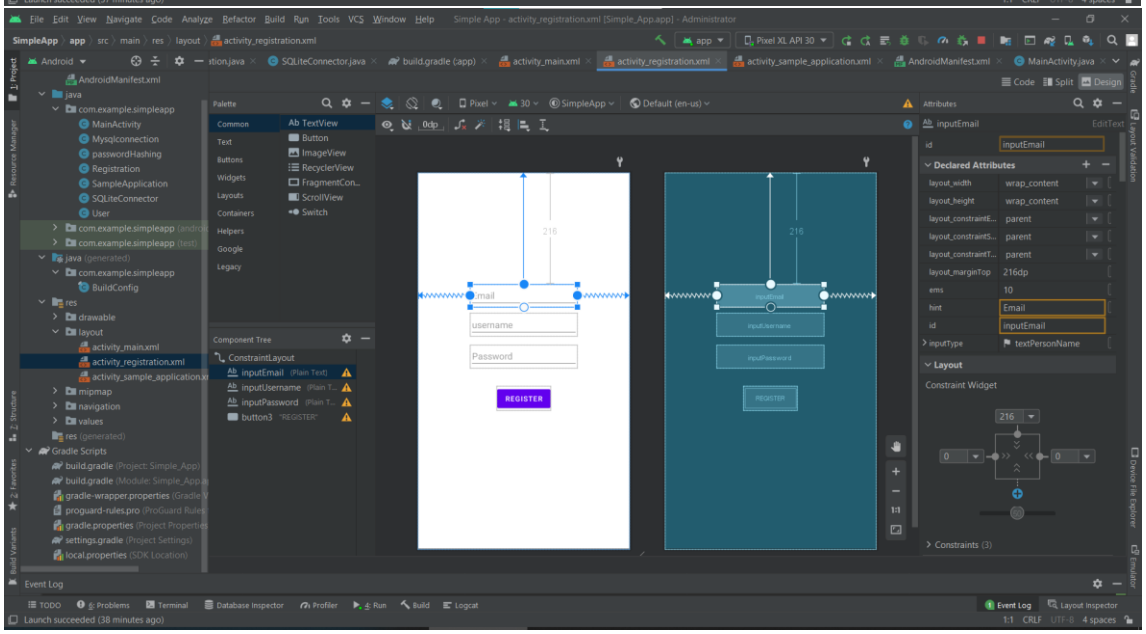
Kết quả sau khi tạo các Activity cần cho ứng dụng

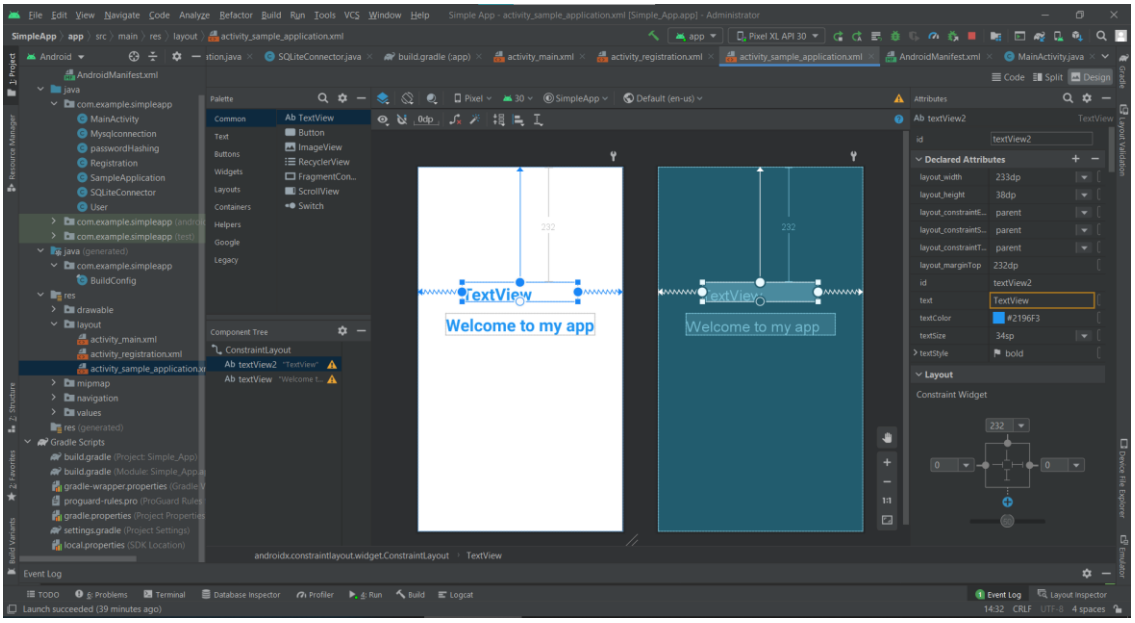


Bước 3: Tạo layout và giao diện cho Activity

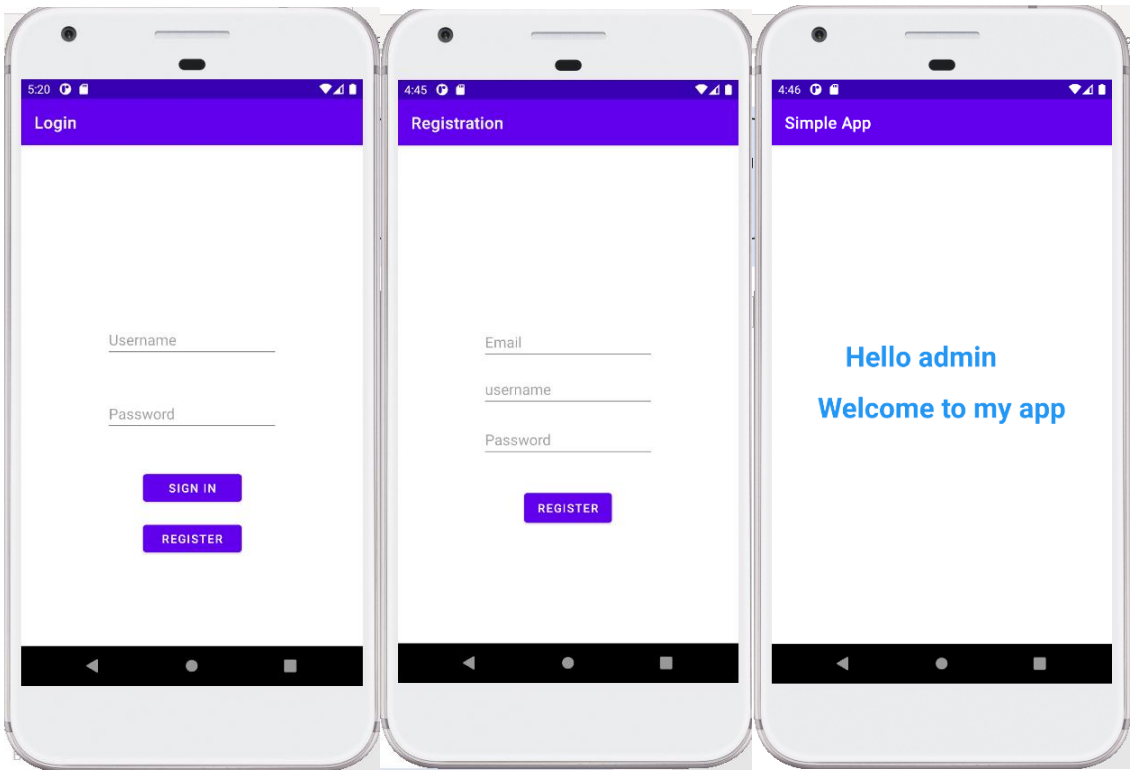
Chuyển sang chế độ Design để sử dụng kéo thả, thuận tiện hơn cho việc thiết kế UI và tập trung vào phần backend.





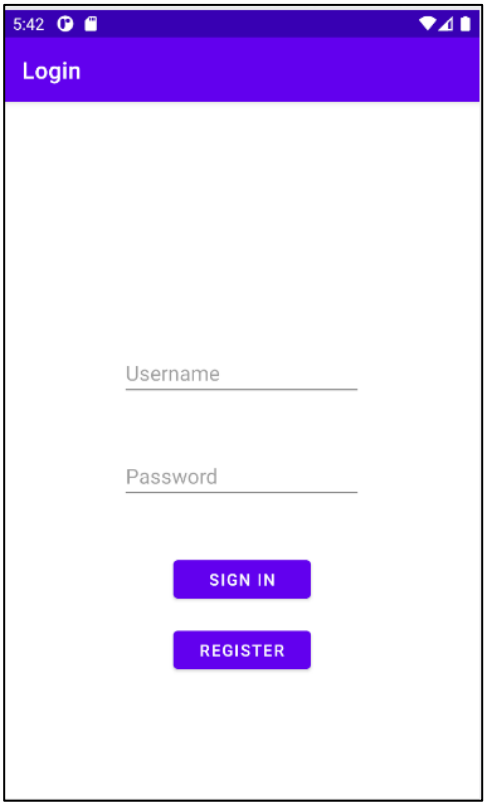


Sau khi hoàn thành kết quả giao diện các activity như bên dưới:



Bước 4: Tạo sự kiện cho button và navigation giữa các activity

❖ Main Activity




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


    Button regisActivity = (Button) findViewById(R.id.button2);
    regisActivity.setOnClickListener(new View.OnClickListener() {
        public void onClick(View view) {
            Intent myIntent = new Intent( packageContext: MainActivity.this, Registration.class);
            startActivity(myIntent);
        }
    });
```

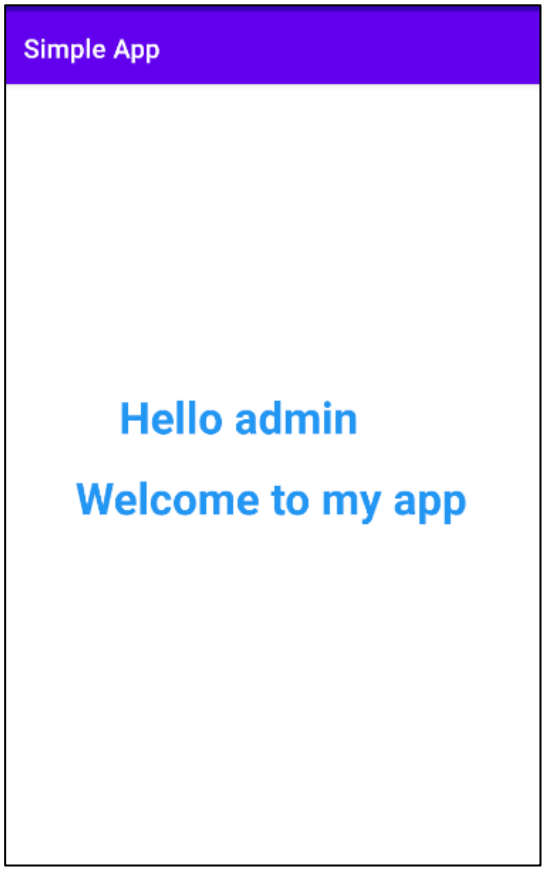
Sử dụng một intent để gọi Registration activity sau khi nhấn button Register trong Main activity.

Tiếp theo tạo sự kiện cho button Login trong main activity, khi nhấn nút nó sẽ lấy thông tin input username và password kiểm tra trong database nếu hợp lệ sẽ chuyển sang welcome, đồng thời gửi một intent kèm message là username sang Welcome activity. Ngược lại hiện một Toast với message là "Login failed! Try again".

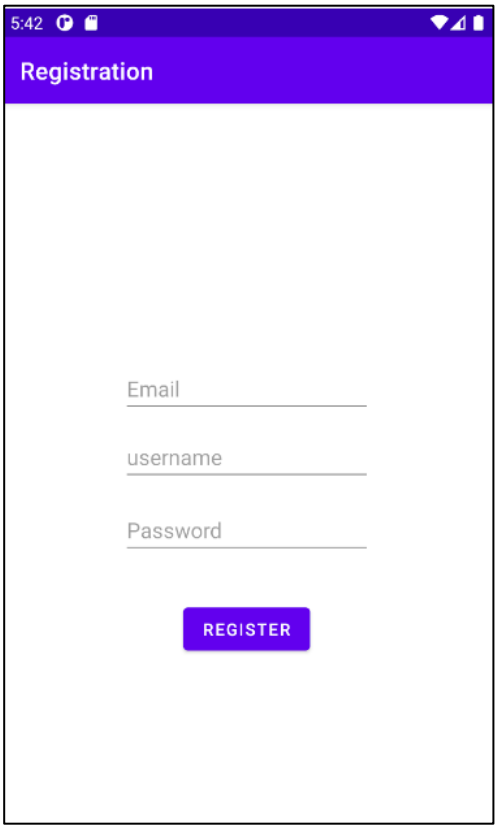
Tại điểm này password của người dùng khi nhập và sẽ được hash bằng MD5 sau đó đưa kết quả để kiểm tra trong database.

```
Button signupActivity = (Button)findViewById(R.id.button);
signupActivity.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view) {

        EditText inputUsername = (EditText)findViewById(R.id.inputCheckUsername);
        EditText inputPassword = (EditText)findViewById(R.id.inputCheckPassword);

        String username = inputUsername.getText().toString();
        String password = inputPassword.getText().toString();
        String passHash = passwordHashing.doHashing(password);

        if (db.checkUser(username, passHash)) {
            Intent intent = new Intent( packageContext: MainActivity.this, SampleApplication.class);
            intent.putExtra( name: "username",username);
            startActivity(intent);
        }
        else {
            String status = "Login failed! Try again!";
            Toast.makeText( context: MainActivity.this, status, Toast.LENGTH_SHORT).show();
        }
    }
});
```

Kết quả sau khi login thành công với admin/admin



❖ **Registration Activity**



Khi nhấn button Register ở Registration activity, lấy thông tin người dùng nhập vào sau đó kiểm tra điều kiện nhập đủ hay chưa, nếu chưa sẽ thông báo là "Please fill all information". Ngược lại sẽ thực hiện xử lí công việc chính.

```
//Click register button
Button regis = (Button)findViewById(R.id.button3);
regis.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view) {
        // get information
        EditText email = (EditText) findViewById(R.id.inputEmail);
        EditText username = (EditText) findViewById(R.id.inputUsername);
        EditText password = (EditText) findViewById(R.id.inputPassword);

        String emailString = email.getText().toString();
        String usernameString = username.getText().toString();
        String passwordString = password.getText().toString();


        if (emailString.isEmpty() || usernameString.isEmpty() || passwordString.isEmpty()){
            String message = "Please fill all information!";
            Toast.makeText( context: Registration.this, message, Toast.LENGTH_SHORT).show();
        } else {
            // Processing....
```

Ngược lại, nếu user nhập đủ thông tin email, username, password thì kiểm tra xem trong database có tồn tại user hay chưa sử dụng hàm checkUser trong

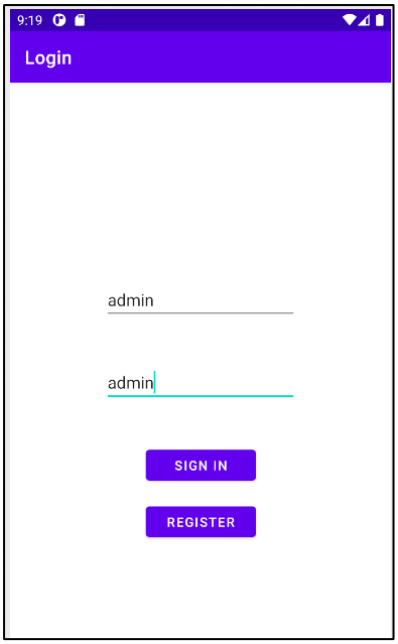
SQLiteConnector bài lab đã cung cấp. Nếu tồn tại sẽ thông báo "User exist!" và cần nhập lại.

Ngược lại nếu chưa tồn tại sẽ tạo đối tượng User từ class đã tạo, thiết lập các thông tin cho user sau đó sử dụng hàm addUser được cung cấp để thêm user mới. Sau khi add thành công sẽ báo "create successfully!" chuyển sang main activity ngược lại sẽ báo lỗi ""Something went wrong! Try again!"

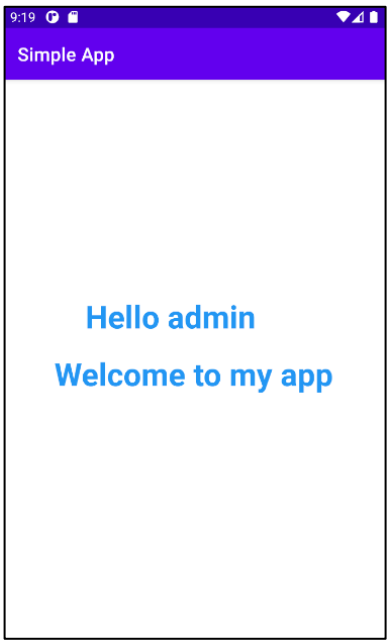
```
        } else {
            // Processing....
            String result;
            String passHash = passwordHashing.doHashing(passwordString);
            Log.i( tag: "hashhhhhhhh: ", passHash);
            User user = new User();

            Mysqlconnection mysql = new Mysqlconnection();
            // ------------------------------------------------------
            if (db.checkUser(usernameString, passHash)) { //--SQLite
            //if (mysql.checkUser(usernameString, passHash)) {
                result = "User exist!";
            }
            else {
                user.setId(1);
                user.setEmail(emailString);
                user.setName(usernameString);
                user.setPassword(passHash);

               db.addUser(user);  //SQLite

                //mysql.addUser(usernameString, passHash, emailString);

//                if (mysql.checkUser(usernameString, passHash)){
                if (db.checkUser(usernameString, passHash)){
                    result = "Create Successfully!";
                    email.setText("");
                    username.setText("");
                    password.setText("");
                    Intent myIntent = new Intent( packageContext: Registration.this, MainActivity.class);
                    startActivity(myIntent);
```

```
package com.example.simpleapp;

public class User {
    private int id;
    private String name;
    private String email;
    private String password;

    public int getId() { return id; }

    public void setId(int id) { this.id = id; }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getEmail() { return email; }

    public  void setEmail(String email) { this.email = email; }

    public String getPassword() { return password; }

    public void setPassword(String password) { this.password = password; }
}
```

❖ Demo và minh chứng



Sign up



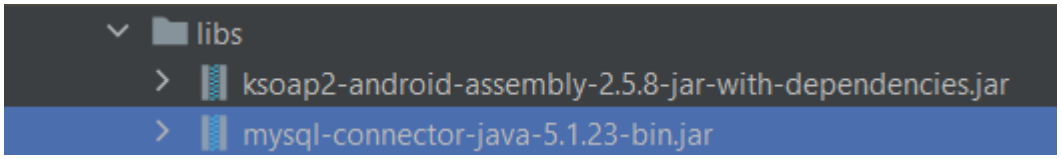
Chi tiết xem ở phần link video đính kèm

2. Yêu cầu 4

> Tạo một cơ sở dữ liệu tương tự bên ngoài thiết bị, viết mã nguồn thực hiện kết nối đến CSDL này để truy vấn thay vì sử dụng SQLite.

Để thực hiện yêu cầu này nhóm sử dụng:

- MySql server 5.7 (UI phpmyadmin)
- Sử dụng mysql-connector-java



Trong code thêm 1 class Mysqlconnection với 3 function:

- Hàm tạo connection
- Hàm addUser
- Hàm checkUser

Khai báo các thông số để thực hiện kết nối đến MySQL server

```
String classs = "com.mysql.jdbc.Driver";
String url = "jdbc:mysql://192.168.1.3:3306/simpleapp";
String un = "root";
String password = "123456";
```

```
@SuppressLint("NewApi")
public Connection CONN() {
    StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder()
            .permitAll().build();
    StrictMode.setThreadPolicy(policy);
    Connection conn = null;
    //String ConnURL = null;
    try {
        Class.forName(classs);
        conn = DriverManager.getConnection(url, un, password);
        if (conn != null) {
            Log.d( tag: "MYSQL:",  msg: "CONNECTED!" );
        }
        //conn = DriverManager.getConnection(ConnURL);

    } catch (SQLException se) {
        Log.e( tag: "ERRO", se.getMessage());
    } catch (ClassNotFoundException e) {
        Log.e( tag: "ERRO", e.getMessage());
    } catch (Exception e) {
        Log.e( tag: "ERRO", e.getMessage());
    }
    return conn;
}
```

```
public boolean checkUser(String username, String password){
    Connection conn = new Mysqlconnection().CONN();
    String query = "SELECT username,password from users WHERE username=" + "'" + username + "' and " + "password=" + "'" + password + "'";
    Log.d( tag: "QUERY: ", query);
    Statement statement = null;
    try {
        statement = conn.createStatement();
        ResultSet result = statement.executeQuery(query);
        if(result.next()) {
            while (result.next()){
                String re = result.getString( columnIndex: 1) + "/" + result.getString( columnIndex: 2);
                Log.d( tag: "RESULT: ", re);
            }
            statement.close();
            try { conn.close(); Log.d( tag: "MYSQL:",  msg: "DISCONNECTED!");} catch (Exception e) { /* Ignored */ }
            return true;
        } else {
            conn.close();
            return false;
        }
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
    return false;
}
```

```
public boolean addUser(String username, String password, String email) {
    Connection conn = new Mysqlconnection().CONN();
    String query = "INSERT INTO users (username, password, email) values(" + "'" + username + "','" + password + "','" + email + "')";
    Log.d( tag: "QUERY: ", query);
    Statement statement = null;
    try {
        statement = conn.createStatement();
        statement.executeUpdate(query);
        statement.close();
        try { conn.close(); Log.d( tag: "MYSQL:", msg: "DISCONNECTED!");} catch (Exception e) { /* Ignored */ }
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
```

Một chút thay đổi so với SQLite: sau khi tạo class, tạo đối tượng và gọi các hàm checkUser và addUser tương tự như SQLite.

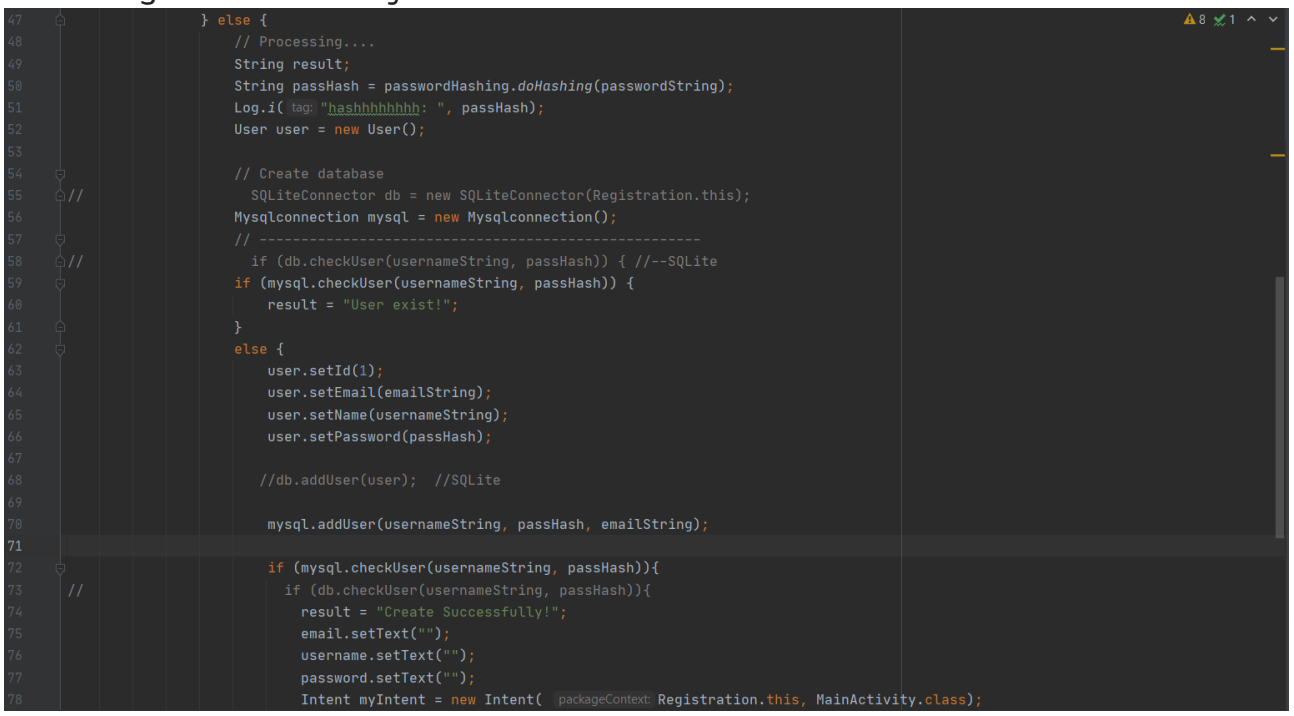
❖ Main acitivity

```
//---------------------------MYSQL---------------------------
ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.INTERNET}, PackageManager.PERMISSION_GRANTED);
StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
StrictMode.setThreadPolicy(policy);
Button signupActivity = (Button) findViewById(R.id.button);
signupActivity.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view) {

        EditText inputUsername = (EditText) findViewById(R.id.inputCheckUsername);
        EditText inputPassword = (EditText) findViewById(R.id.inputCheckPassword);

        String username = inputUsername.getText().toString();
        String password = inputPassword.getText().toString();
        String passHash = passwordHashing.doHashing(password);

        Mysqlconnection mysql = new Mysqlconnection();

        if (mysql.checkUser(username, passHash)) {
            inputUsername.setText("");
            inputPassword.setText("");
            Intent intent = new Intent( packageContext: MainActivity.this, SampleApplication.class);
            intent.putExtra( name: "username",username);
            startActivity(intent);
        } else {
            String status = "Login failed! Try again!";
            Toast.makeText( context: MainActivity.this, status, Toast.LENGTH_SHORT).show();
        }
    }
});
```

```
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"></uses-permission>
```

Lưu ý: để sử dụng MySQL server cần cấp thêm quyền sử dụng internet và thiết lập policy ở mức thấp cho phép tất nếu không sẽ không thể kết nối đến MySQL.

❖ Registration acitivity



```
} else {
    // Processing....
    String result;
    String passHash = passwordHashing.doHashing(passwordString);
    Log.i( tag: "hashhhhhhhh: ", passHash);
    User user = new User();

    // Create database
//    SQLiteConnector db = new SQLiteConnector(Registration.this);
    Mysqlconnection mysql = new Mysqlconnection();
    // ------------------------------------------------------
//    if (db.checkUser(usernameString, passHash)) { //--SQLite
    if (mysql.checkUser(usernameString, passHash)) {
        result = "User exist!";
    }
    else {
        user.setId(1);
        user.setEmail(emailString);
        user.setName(usernameString);
        user.setPassword(passHash);

        //db.addUser(user);  //SQLite

        mysql.addUser(usernameString, passHash, emailString);

        if (mysql.checkUser(usernameString, passHash)){
//        if (db.checkUser(usernameString, passHash)){
            result = "Create Successfully!";
            email.setText("");
            username.setText("");
            password.setText("");
            Intent myIntent = new Intent( packageContext: Registration.this, MainActivity.class);
```

❖ **Demo và minh chứng**

*Chi tiết trong video đính kèm

3. Yêu cầu 6

> Với ứng dụng đã xây dựng, tìm hiểu và sử dụng công cụ ProGuard để tối ưu hóa mã nguồn. Trình bày khác biệt trước và sau khi sử dụng?

ProGuard là công cụ tích hợp sẵn trong Android Studio

- Tính năng:
  - Thu gọn mã nguồn ứng → để dễ phân phối
  - Làm rối → để chống dịch ngược, vì tên các hàm, biến ... bị đổi tên khó đọc
  - Tối ưu để ứng dụng chạy nhanh hơn.
- ProGuard là một công cụ rút gọn (shrink), tối ưu hoá (optimize) và làm mờ (obfuscate) code.
- Sử dụng ProGuard để bảo vệ và tối ưu ứng dụng Android.

Cách sử dụng ProGuard trong android studio

```
android {
    //..
    buildTypes {
        release { //Thêm một khối debug nếu muốn
            minifyEnabled true //Thu gọn code, false nếu không dùng
            useProguard true
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
                'proguard-rules.pro'
        }
    }
}
```

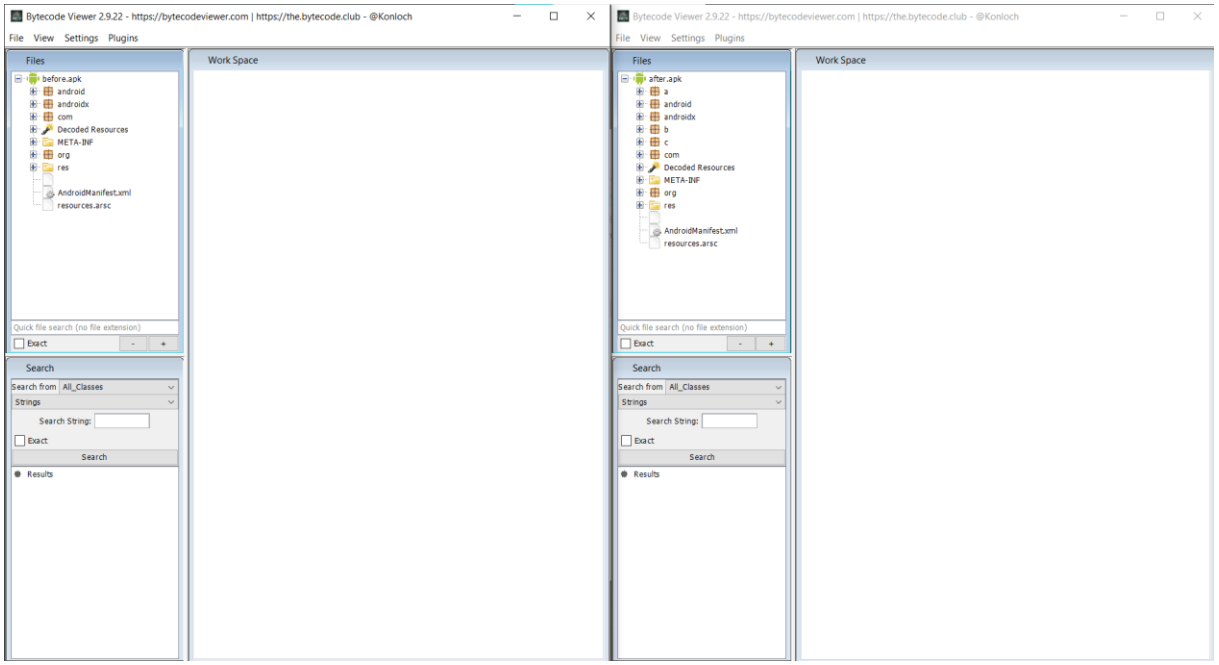
- MinifyEnabled: true bật tính năng ProGuard
- UseProguard: làm rối code

```
android {
    //..
    buildTypes {
        release { //Thêm một khối debug nếu muốn
            minifyEnabled true //Thu gọn code, false nếu không dùng
            useProguard true   //Làm rối code
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
                'proguard-rules.pro'
        }

        debug {
            minifyEnabled true
            useProguard false //Không làm rối code
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
                'proguard-rules.pro'
        }

    }
}
```

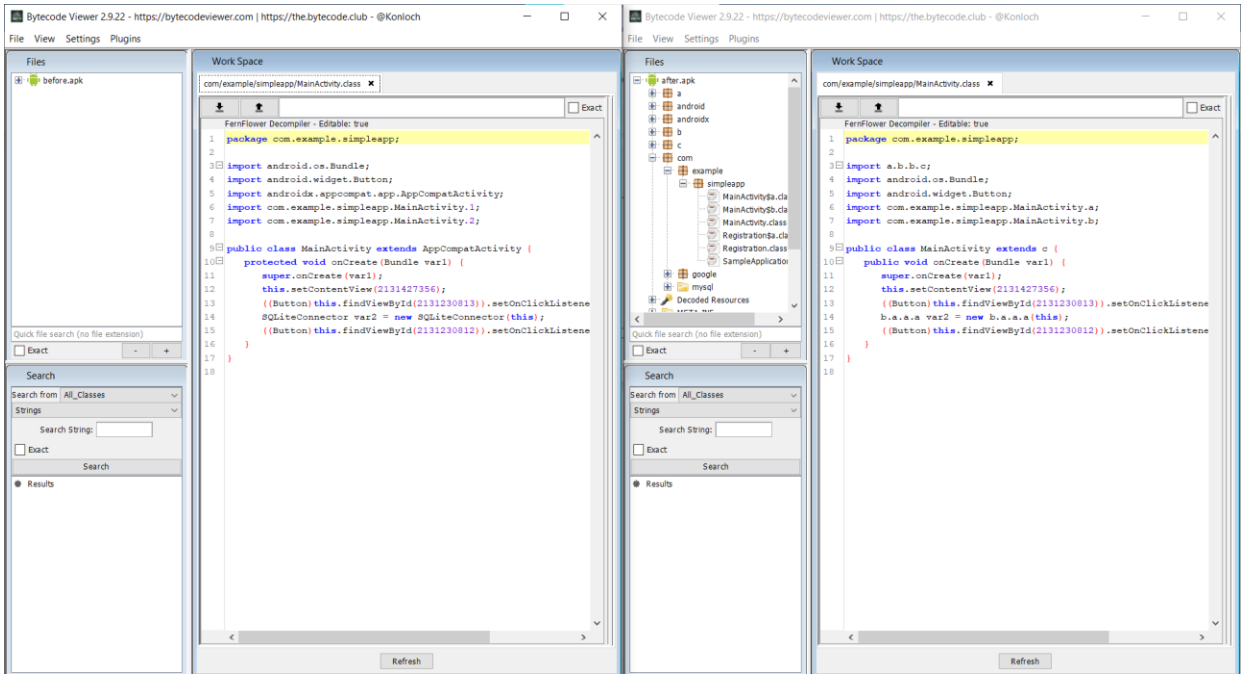
Để sử dụng ProGuard thêm một block debug tương tự như release bởi vì khi build ra file .apk sẽ nằm ở Debug.

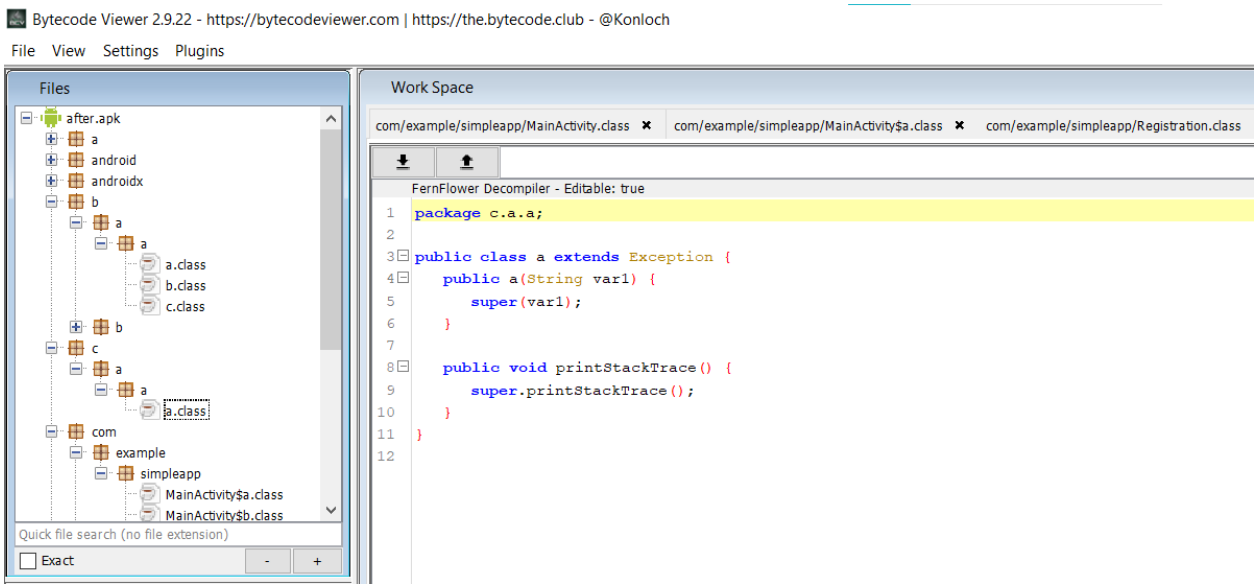
So sánh trước và sau khi sử dụng ProGuard

- Sau khi sử dụng proguard cấu trước sau khi extract từ file .apk số lượng các thư mục file tăng lên



- Các thư viện, class đều bị đổi thành các kí tự a,b,c,... ➔ khiến cho attacker dịch ngược đọc hiểu sẽ rất khó.





Các thư mục file, class đều bị xáo trộn gây khó hiểu nếu dịch ngược để tấn công.

-- Hết --