

Họ tên: Lê Phúc Hưng

MSSV: 20215276

Mã lớp: 151902

Môn học: Phát triển ứng dụng cho thiết bị di động (IT4785)

Source Code: <https://github.com/lephuchung/HustMobile>

Bài: Lesson 5 – Dice Roller app – Lemonade app

Contents

1. Classes and object instances in Kotlin.....	4
1.1. Roll random numbers	4
1.1.1. Set up your starter code	4
1.1.2. Use the random function	4
1.2. Create a Dice class.....	6
1.2.1. Define a Dice class	6
1.2.2. Create an instance of the Dice class	6
1.2.3. Make the Dice Roll.....	7
1.3. Return your dice roll's value	9
1.4. Change the number of sides on your dice	10
1.5. Customize your dice	12
1.6. Adopt good coding practices	13
1.7. Solution code	13
2. Create an interactive Dice Roller app.....	15
2.1. Set up your app	15
2.1.1. Create an Empty Activity project	15
2.2. Create the layout for the app	16
2.2.1. Open the Layout Editor	16
2.2.2. Add a Button to the layout.....	16
2.2.3. Position the Button	17
2.2.4. Change the Button text	18
2.2.5. Style the TextView.....	20

2.3. Introduction to Activities	21
2.3.1. Open the MainActivity.kt file.....	21
2.3.2. Enable auto imports	22
2.4. Make the Button interactive	22
2.4.1. Display a message when the Button is clicked.....	22
2.4.2. Update the TextView when the Button is clicked	25
2.5. Add the dice roll logic	26
2.5.1. Add the Dice class	26
2.5.2. Create a rollDice() method	27
2.5.3. Create a new Dice object instance.....	27
2.6. Adopt good coding practices	28
2.6.1. Clean up your code	28
3. Add images to the Dice Roller app.....	29
3.1. Update the layout for the app	29
3.1.1. Open Dice Roller app	29
3.1.2. Delete the TextView	29
3.1.3. Add an ImageView to the layout	29
3.1.4. Position the ImageView and Button	29
3.2. Add the dice images.....	31
3.2.1. Download the dice images.....	31
3.2.2. Add dice images to your app	31
3.3. Use the dice images	32
3.3.1. Replace the sample avatar image	32
3.3.2. Change the dice image when the button is clicked	34
3.4. Display the correct dice image based on the dice roll	36
3.4.1. Update the rollDice() moethod.....	36
3.4.2. Optimize your code	37
3.4.3. Set an appropriate content description on the ImageView	38
3.5. Adopt good coding practices	39
3.5.1. Create a more useful launch experience	39

3.5.2. Comment your code	39
4. Project: Lemonade app	40
4.1. App overview.....	40
4.2. Get started.....	40
4.3. Build your user interface	40
4.4. Make your app interactive	41
4.5. Run your app	42
4.6. Testing instructions.....	44

1. Classes and object instances in Kotlin

1.1. Roll random numbers

1.1.1. Set up your starter code



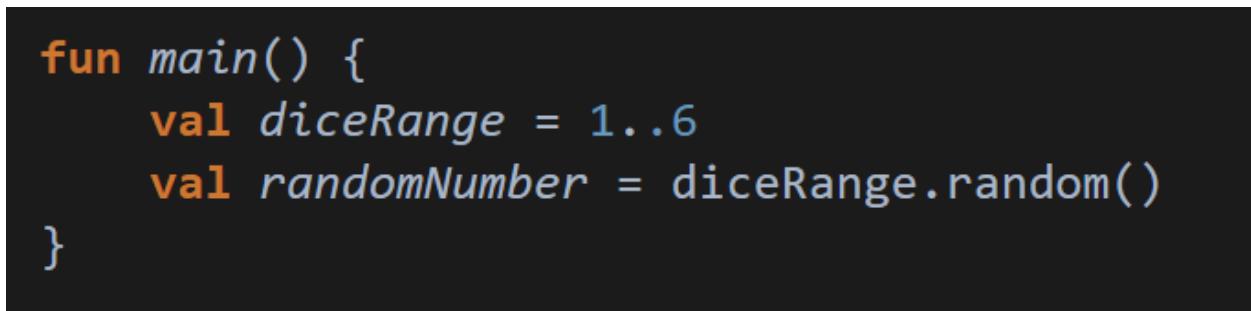
The screenshot shows the Kotlin playground interface. At the top, there's a navigation bar with the Kotlin logo, "Solutions", "Docs", and "Community". Below the navigation bar, there are dropdown menus for "2.0.21" and "JVM". The main area contains the following Kotlin code:

```
fun main() {
    val kotlin = "😊"
    println(kotlin)
}
```

1.1.2. Use the random function

Các bước thực hiện:

1. Trong hàm main(), đặt biến val tên là diceRange, gán nó cho một dãy giá trị nguyên từ 1 đến 6
2. Đặt biến val tên là randomNumber, gọi radom()
3. In ra giá trị random



The screenshot shows the Kotlin playground interface. The code has been updated to use the random function:

```
fun main() {
    val diceRange = 1..6
    val randomNumber = diceRange.random()
}
```

```
fun main() {  
    val diceRange = 1..6  
    val randomNumber = diceRange.random()  
    println("Random number: ${randomNumber}")  
}
```

Random number: 1

```
fun main() {  
    val diceRange = 1..6  
    val randomNumber = diceRange.random()  
    println("Random number: ${randomNumber}")  
}
```

Random number: 3

1.2. Create a Dice class

1.2.1. Define a Dice class

Các bước thực hiện:

1. Xóa code ở hàm main
2. Dưới hàm main, viết lớp Dice
3. Trong lớp Dice, thêm biến val tên là sides, đặt giá trị là 6

```
fun main() {  
}  
  
class Dice {  
}  
|
```

```
class Dice {  
    var sides = 6  
}
```

1.2.2. Create an instance of the Dice class

Các bước thực hiện:

1. Để tạo một instance của Dice, trong hàm main, tạo một biến val tên là myFirstDice
2. In ra số sides của myFirstDice

```
fun main() {  
    val myFirstDice = Dice()  
}
```

```
fun main() {  
    val myFirstDice = Dice()  
    println(myFirstDice.sides)  
}  
  
class Dice {  
    var sides = 6  
}
```

6

1.2.3. Make the Dice Roll

Các bước thực hiện:

1. Trong lớp Dice, thêm hàm roll() để xoay xúc xắc
2. Trong phương thức roll(), tạo một val randomNumber, gán cho nó giá trị random trong khoảng từ 1 đến 6, gọi random()
3. Trong hàm main, gọi phương thức roll() trong myFirstDice
4. Chạy chương trình

```
class Dice {  
    var sides = 6  
  
    fun roll(){  
        |  
    }  
}
```

```
fun roll(){  
    val randomNumber = (1..6).random()  
    println(randomNumber)  
}
```

```
fun main() {  
    val myFirstDice = Dice()  
    println(myFirstDice.sides)  
    myFirstDice.roll()  
}
```

```
fun main() {
    val myFirstDice = Dice()
    println(myFirstDice.sides)
    myFirstDice.roll()
}

class Dice {
    var sides = 6

    fun roll(){
        val randomNumber = (1..6).random()
        println(randomNumber)
    }
}
```

6
4

1.3. Return your dice roll's value

Các bước thực hiện:

1. Trong hàm main, tạo một biến val tên diceRoll
2. Thay đổi hàm roll() trả về Int
3. Trong hàm roll(), bỏ lệnh println() và thay nó bằng lệnh return randomNumber
4. Trong hàm main(), bỏ lệnh in số mặt của xúc xắc
5. Thêm lệnh để in giá trị của sides và diceRoll vào hàm main()
6. Chạy chương trình

```
fun roll(): Int{
    val randomNumber = (1..6).random()
    return randomNumber
}
```

```
fun main() {
    val myFirstDice = Dice()
    val diceRoll = myFirstDice.roll()
    println("Your ${myFirstDice.sides} sided dice rolled ${diceRoll}")
}
```

```
fun main() {
    val myFirstDice = Dice()
    val diceRoll = myFirstDice.roll()
    println("Your ${myFirstDice.sides} sided dice rolled ${diceRoll}")
}

class Dice {
    var sides = 6

    fun roll(): Int{
        val randomNumber = (1..6).random()
        return randomNumber
    }
}
```

```
Your 6 sided dice rolled 2!
```

1.4. Change the number of sides on your dice

Các bước thực hiện:

1. Trong lớp Dice, trong phương thức roll(), thay đổi 1..6 để dùng sides
2. Trong hàm main(), sau khi in dice roll, thay đổi sides của myFirstDice thành 20

3. Viết hàm main() sao cho đầy đủ

4. Chạy chương trình

```
fun main() {
    val myFirstDice = Dice()
    val diceRoll = myFirstDice.roll()
    println("Your ${myFirstDice.sides} sided dice rolled ${diceRoll}")
    myFirstDice.sides = 20
    println("Your ${myFirstDice.sides} sided dice rolled ${diceRoll}")
}
class Dice {
    var sides = 6
    fun roll(): Int {
        val randomNumber = (1..sides).random()
        return randomNumber
    }
}
```

```
fun main() {
    val myFirstDice = Dice()
    val diceRoll = myFirstDice.roll()
    println("Your ${myFirstDice.sides} sided dice rolled ${diceRoll}")
    myFirstDice.sides = 20
    println("Your ${myFirstDice.sides} sided dice rolled ${diceRoll}")
}
class Dice {
    var sides = 6
    fun roll(): Int {
        val randomNumber = (1..sides).random()
        return randomNumber
    }
}
```

```
Your 6 sided dice rolled 2!
Your 20 sided dice rolled 4!
```

```
fun main() {
    val myFirstDice = Dice()
    val diceRoll = myFirstDice.roll()
    println("Your ${myFirstDice.sides} sided dice rolled $diceRoll")
    myFirstDice.sides = 20
    println("Your ${myFirstDice.sides} sided dice rolled $diceRoll")
}

class Dice {
    var sides = 6
    fun roll(): Int {
        val randomNumber = (1..sides).random()
        return randomNumber
    }
}
```

Your 6 sided dice rolled 3!
Your 20 sided dice rolled 10!

1.5. Customize your dice

Các bước thực hiện:

1. Điều chỉnh lớp Dice để nhận vào một biến nguyên tên là numSides
2. Trong lớp Dice, xóa biến sides, từ bây giờ sẽ dùng biến numSides và thay đổi range để sử dụng numSides
3. Trong hàm main(), để tạo myFirstDice với 6 mặt, phải thay đổi số sides như một tham số cho lớp Dice. Trong lệnh in, thay đổi sides thành numSides và xóa phần code sau đó
4. Thêm code để tạo và in ra đối tượng Dice thứ 2 tên là mySecondDice với 20 mặt, thêm lệnh in để xoay và in ra giá trị
5. Chạy chương trình

```

fun main() {
    val myFirstDice = Dice(6)
    val diceRoll = myFirstDice.roll()
    println("Your ${myFirstDice.numSides} sided dice rolled $diceRoll")
    val mySecondDice = Dice(20)
    println("Your ${mySecondDice.numSides} sided dice rolled $diceRoll")
}
class Dice (val numSides: Int) {
    fun roll(): Int {
        val randomNumber = (1..numSides).random()
        return randomNumber
    }
}

```

```

Your 6 sided dice rolled 4!
Your 20 sided dice rolled 16!

```

1.6. Adopt good coding practices

Các bước thực hiện:

1. Thay đổi lệnh return để return một số bất kỳ
2. Gọi myFirstDice.roll() trong chuỗi và xóa biến diceRoll

```

fun main() {
    val myFirstDice = Dice(6)
    println("Your ${myFirstDice.numSides} sided dice rolled ${myFirstDice.roll()}")
    val mySecondDice = Dice(20)
    println("Your ${mySecondDice.numSides} sided dice rolled ${mySecondDice.roll()}")
}
class Dice (val numSides: Int) {
    fun roll(): Int {
        return (1..numSides).random()
    }
}

```

1.7. Solution code

```
fun main() {
    val myFirstDice = Dice(6)
    println("Your ${myFirstDice.numSides} sided dice rolled ${myFirstDice.roll()}")
    val mySecondDice = Dice(20)
    println("Your ${mySecondDice.numSides} sided dice rolled ${mySecondDice.roll()}")
}
class Dice (val numSides: Int) {
    fun roll(): Int {
        return (1..numSides).random()
    }
}
```

?

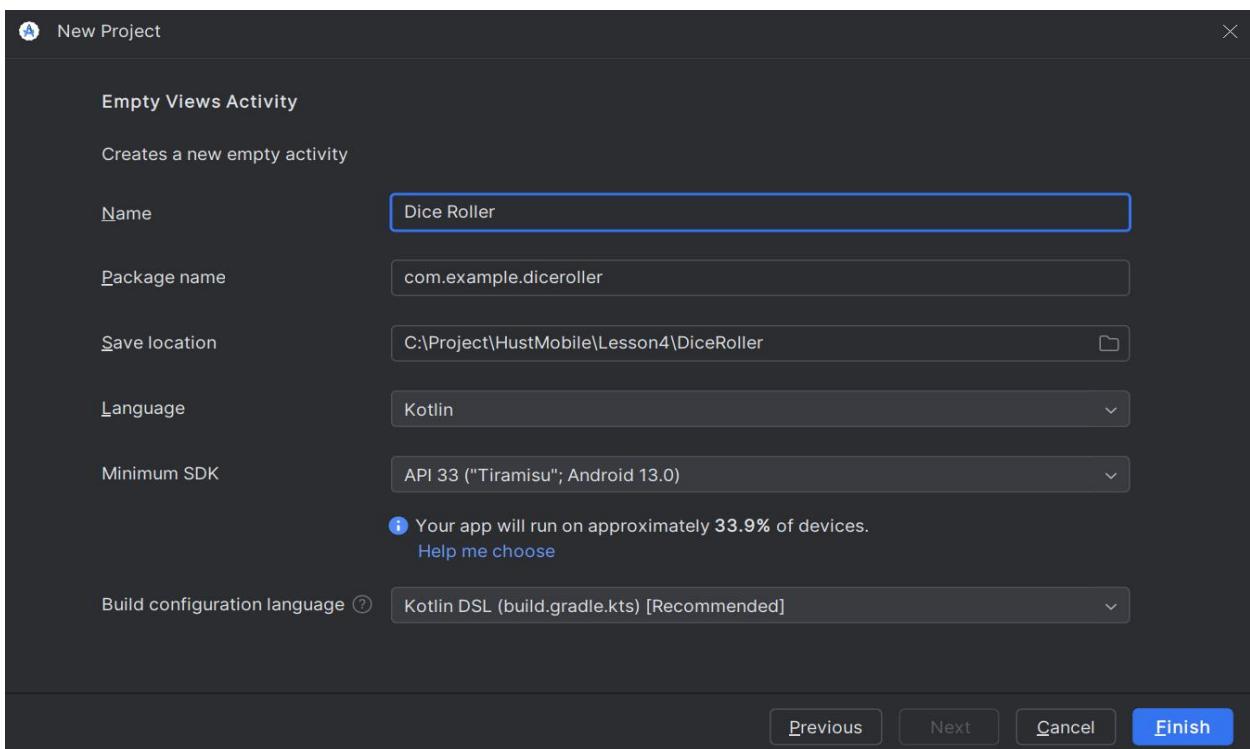
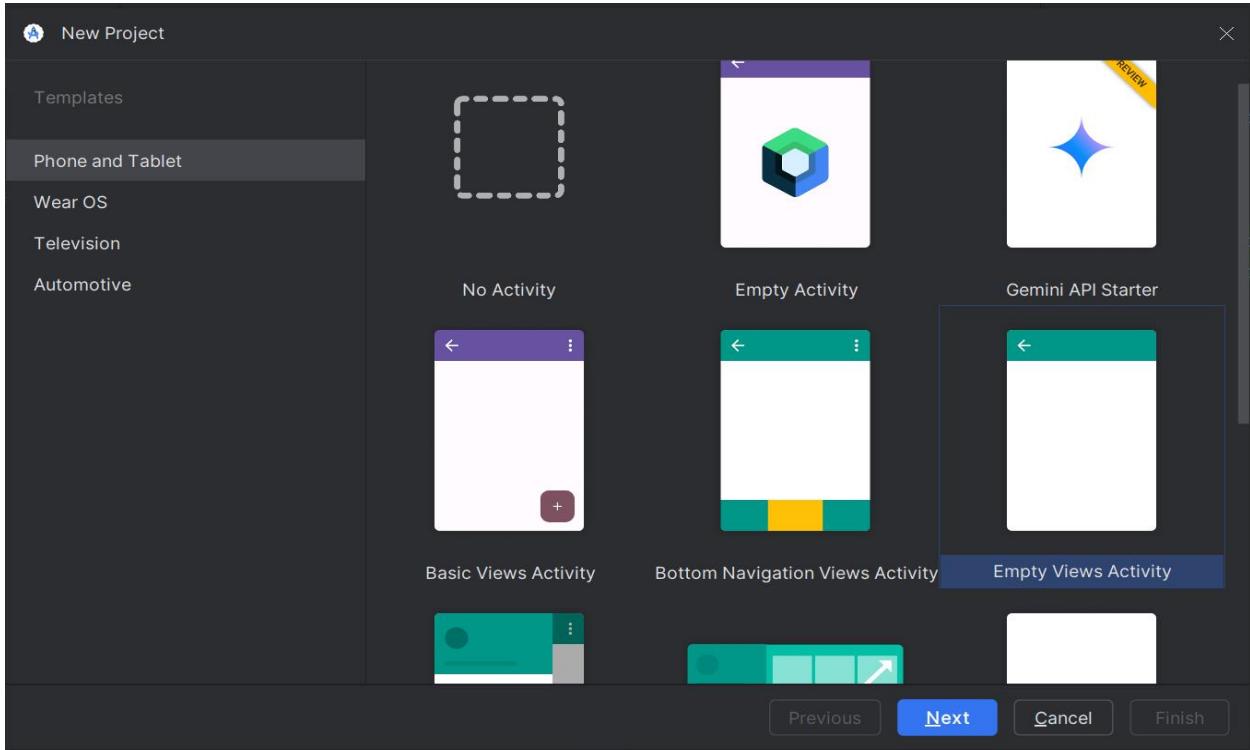
```
Your 6 sided dice rolled 6!
Your 20 sided dice rolled 7!
```

X

2. Create an interactive Dice Roller app

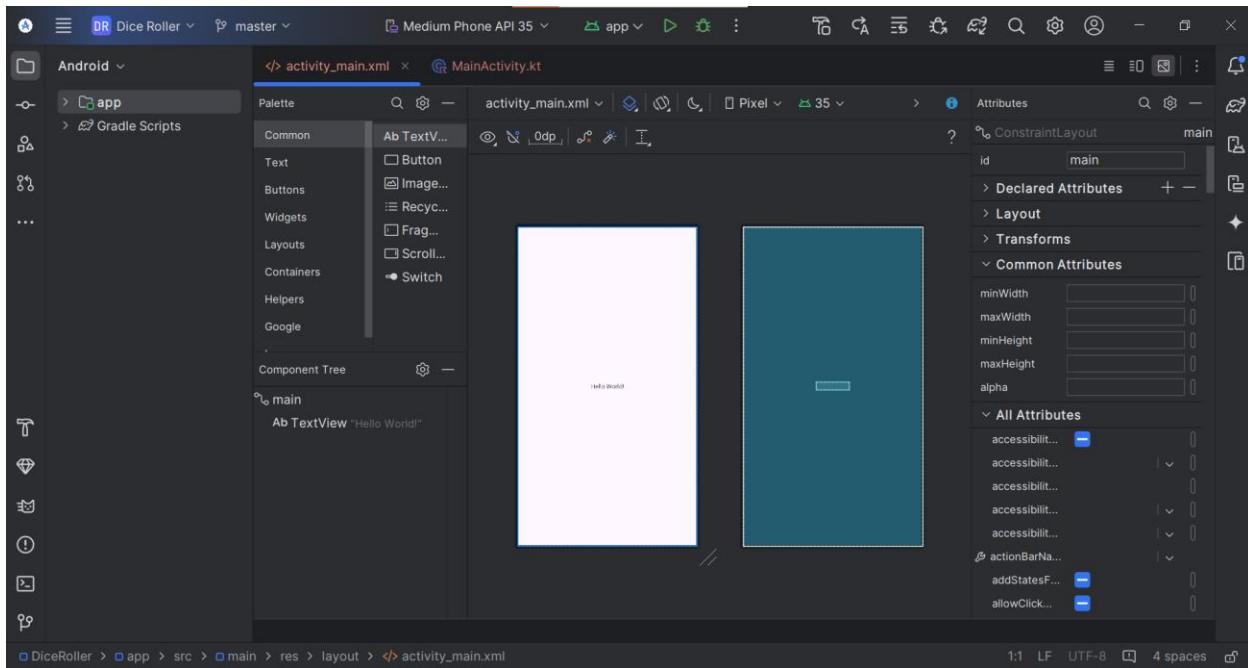
2.1. Set up your app

2.1.1. Create an Empty Activity project



2.2. Create the layout for the app

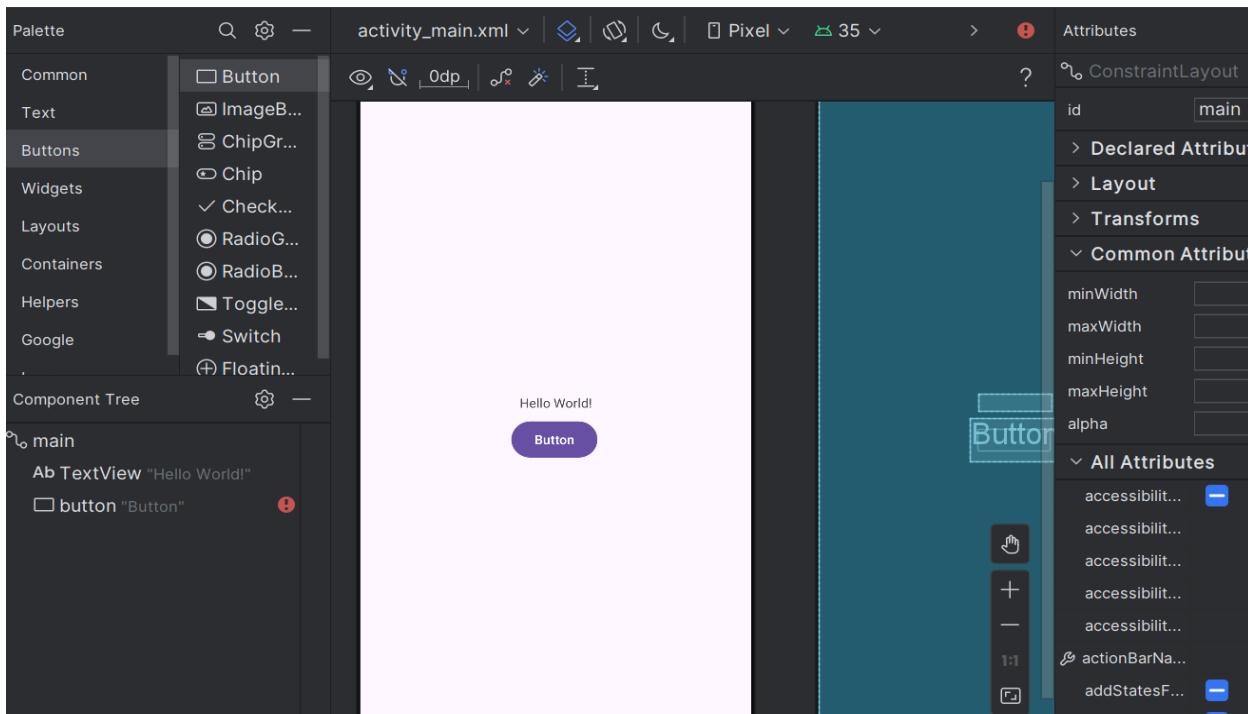
2.2.1. Open the Layout Editor



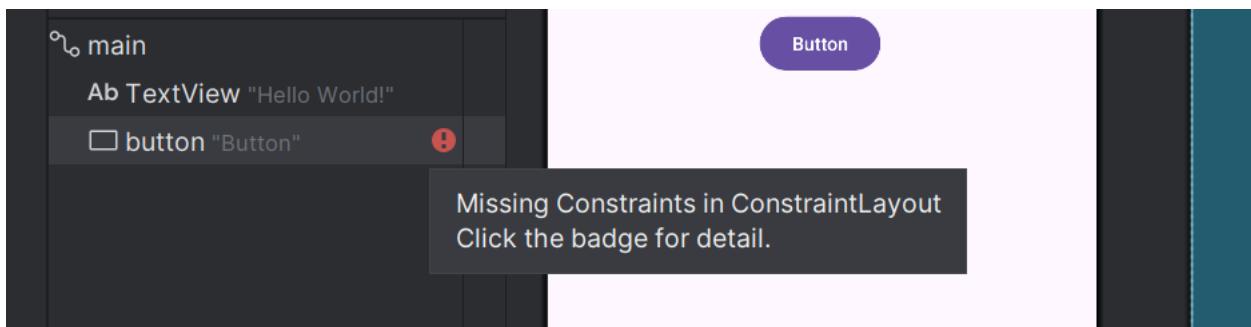
2.2.2. Add a Button to the layout

Các bước thực hiện:

1. Kéo Button từ Palette vào Design view, đặt ở dưới TextView “Hello World”

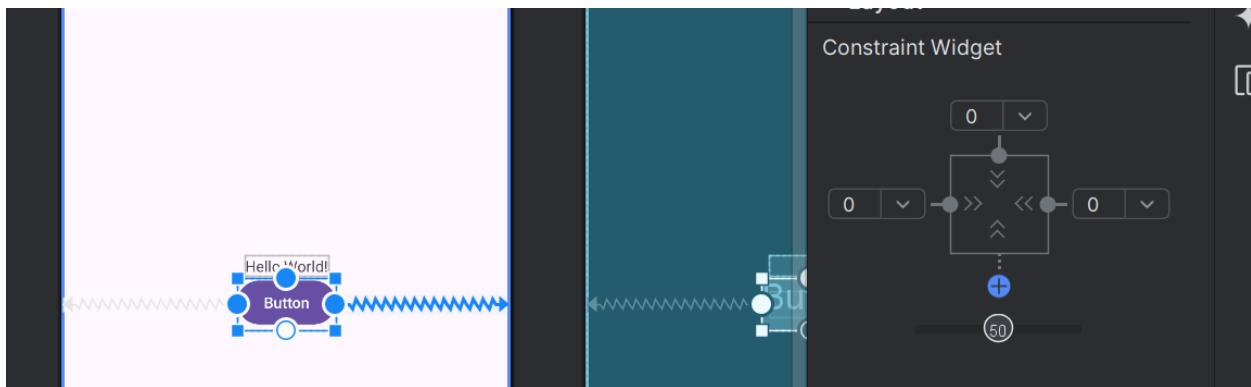


2. Kiểm tra Button và TextView được liệt kê bên dưới ConstraintLayout

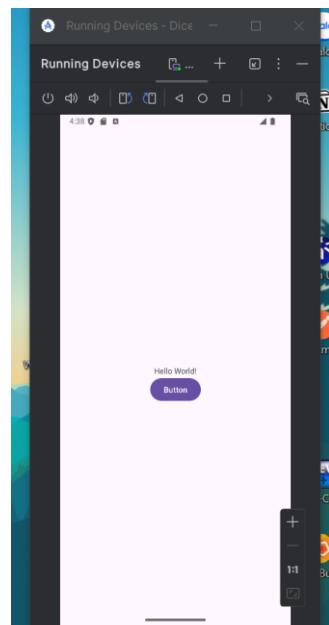


2.2.3. Position the Button

1. Trong Constraint Layout, sửa lề trên thành 0. Kéo hình tròn bên phải vào lề trái và hình tròn bên phải vào lề phải

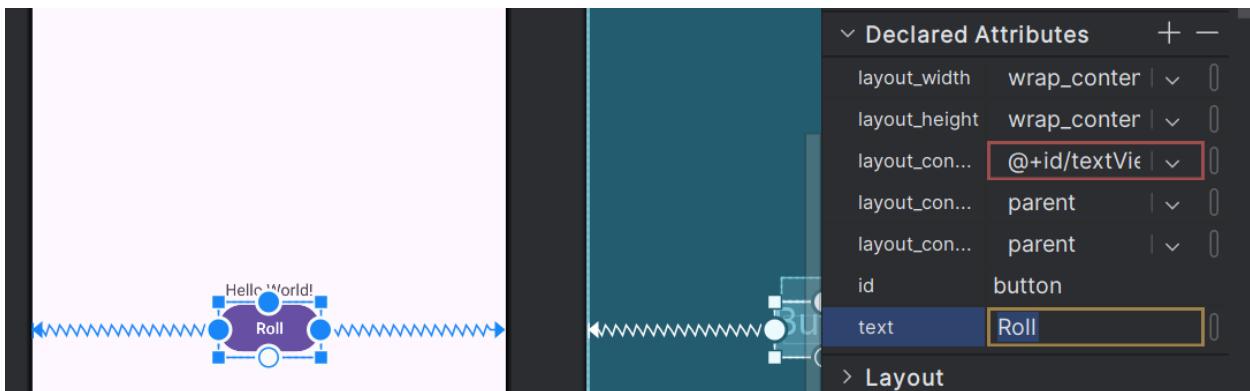


2. Chạy chương trình

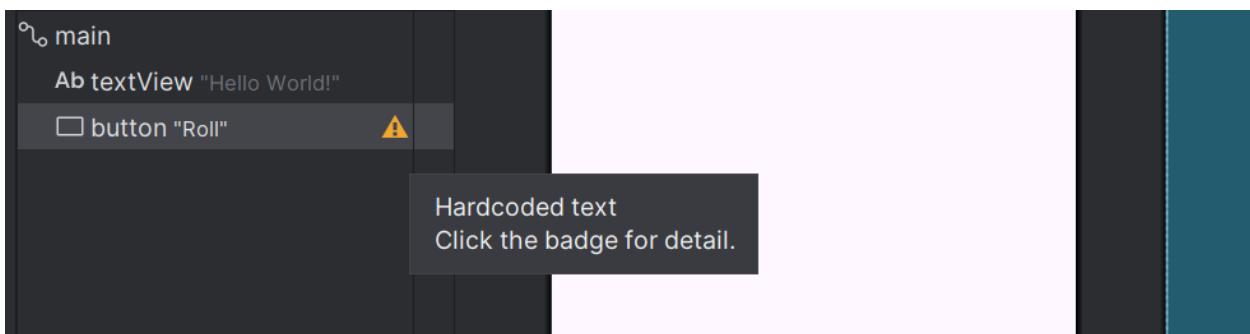


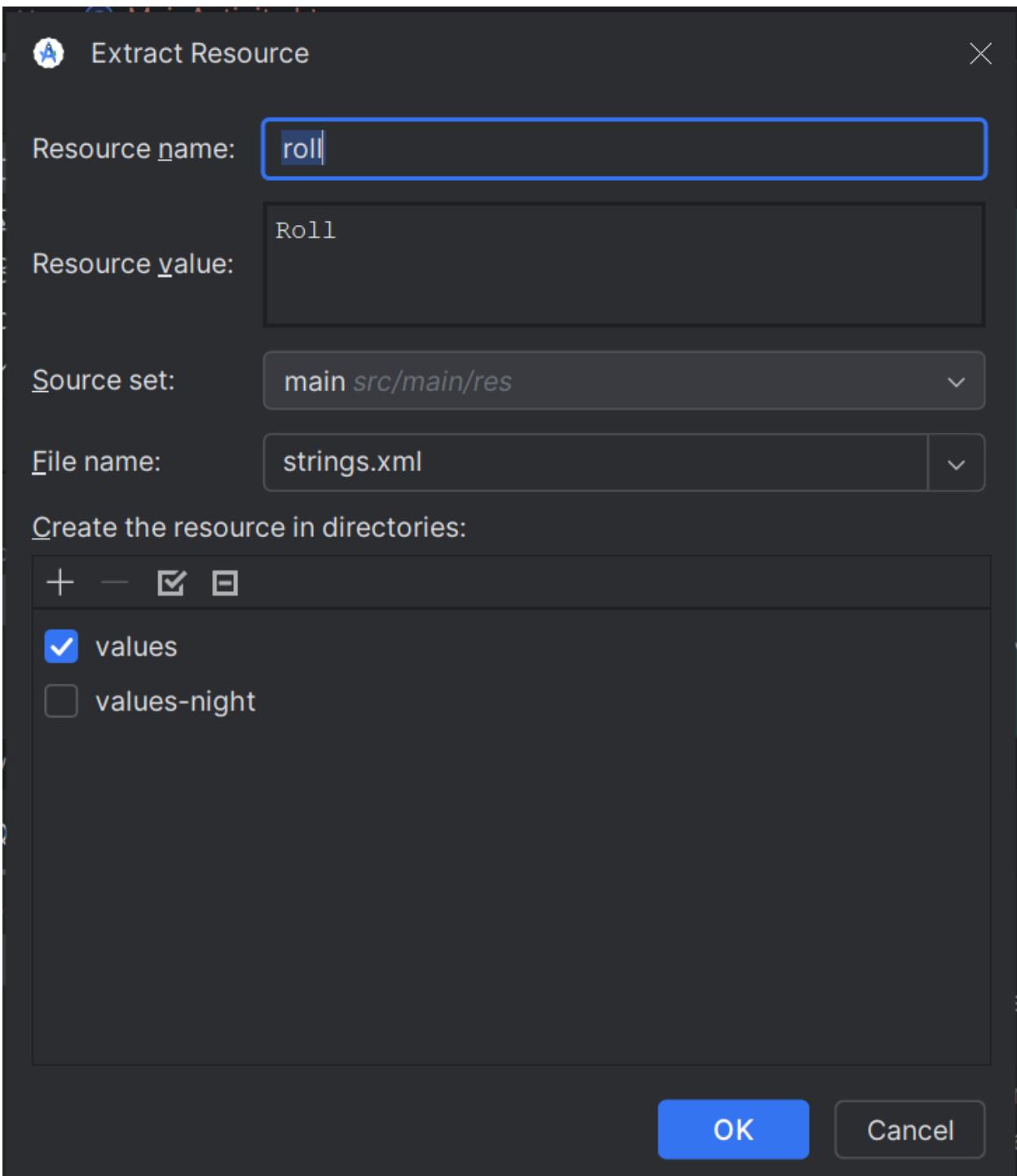
2.2.4. Change the Button text

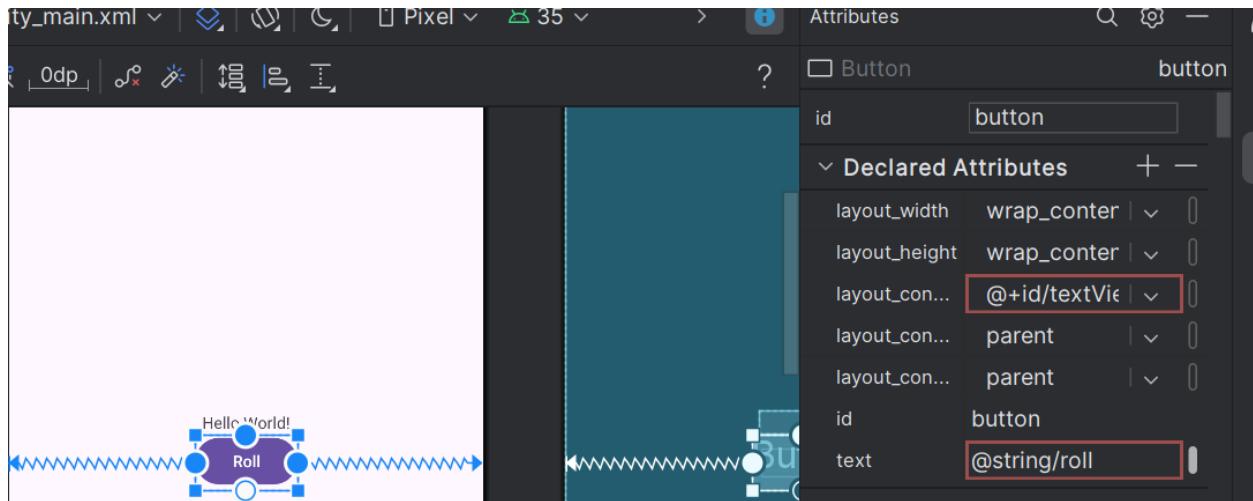
- Trong Layout Editor, thay đổi text thành Roll



- Sửa lỗi ở Button



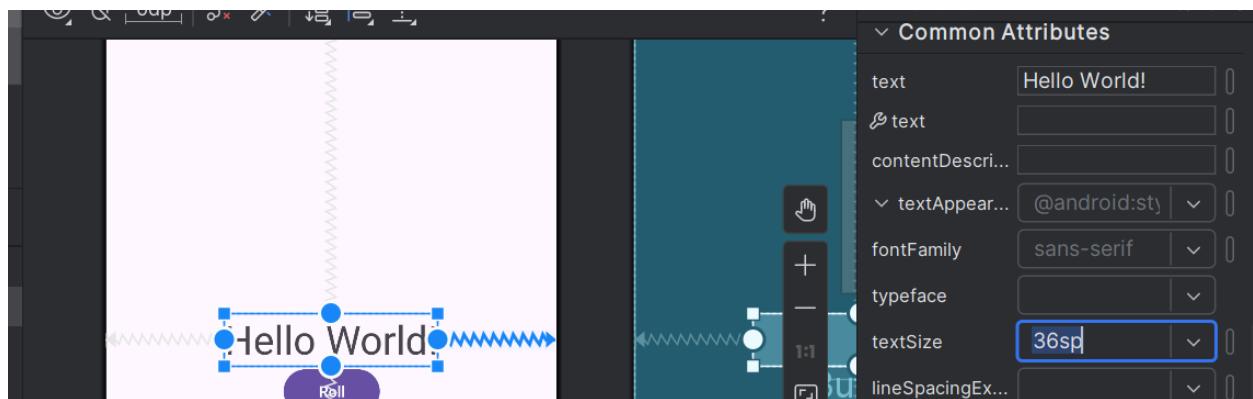




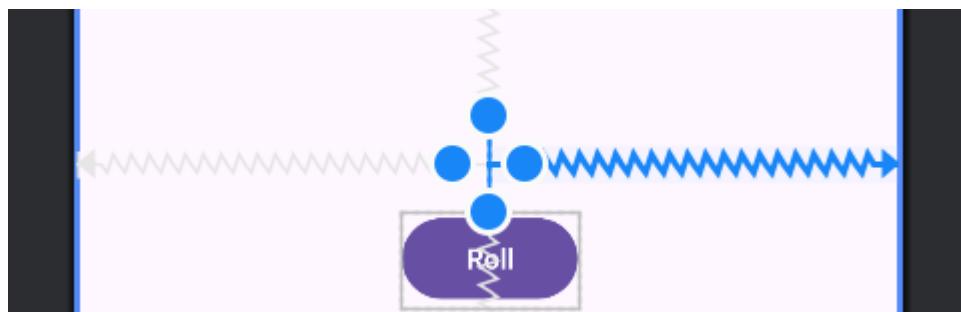
2.2.5. Style the TextView

Các bước thực hiện:

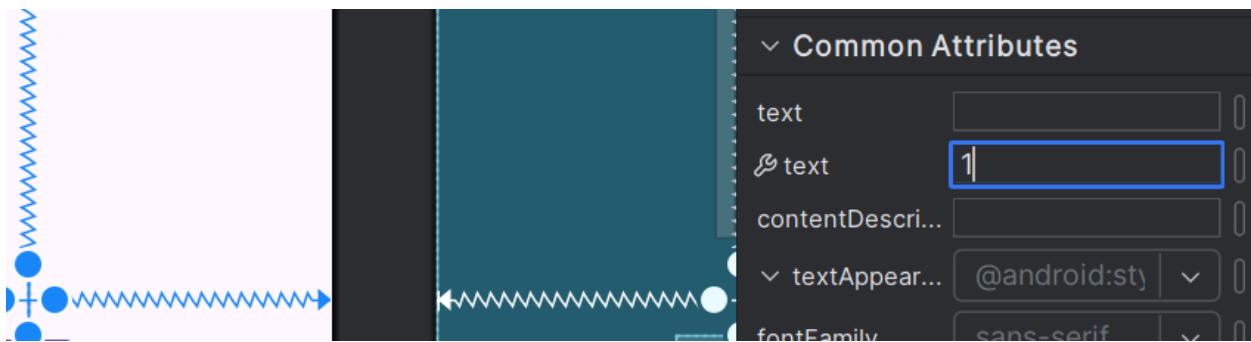
1. Trong Design Editor, chọn TextView và thay đổi textSize thành 36sp



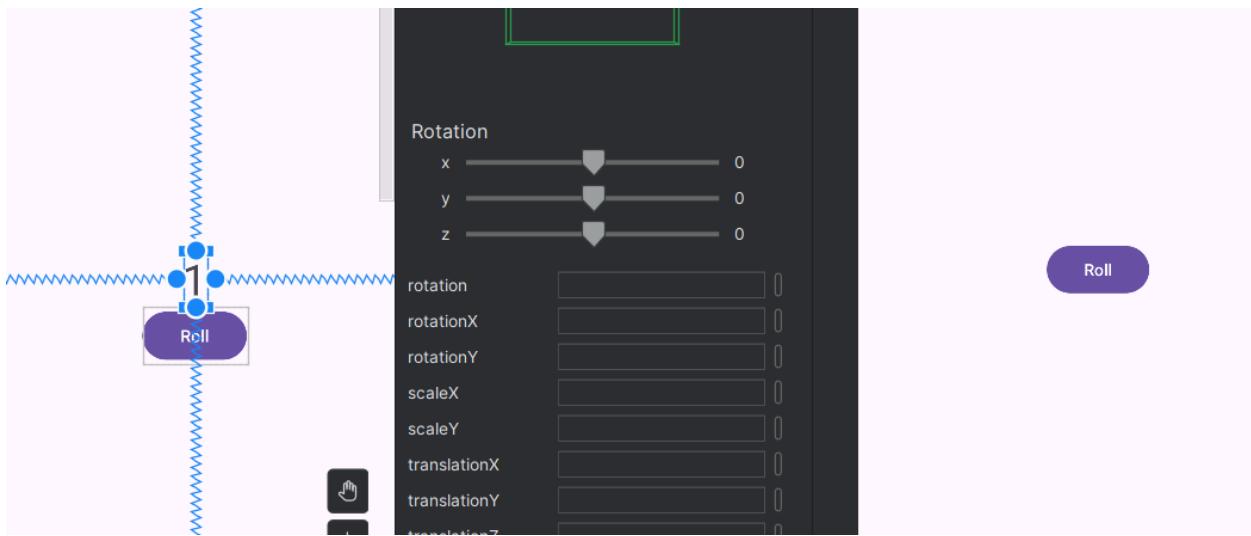
2. Xóa text của TextView vì không cần hiện text gì khi xúc xắc chưa được xoay



3. Chọn TextView trong Component Tree, dưới Common Attributes, tìm text với tool icon, đặt thành 1



4. Chạy chương trình



2.3. Introduction to Activities

2.3.1. Open the MainActivity.kt file

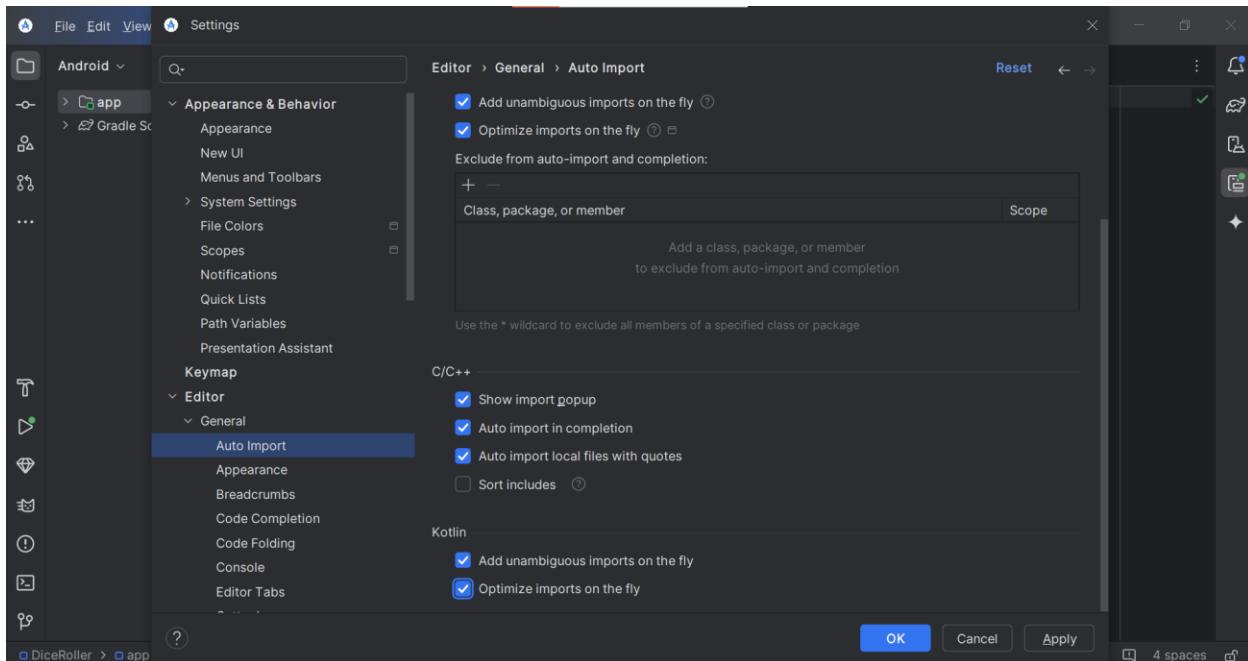
1. Chuyển sang app > java > com.example.diceroller > MainActivity.kt

The screenshot shows the Android Studio interface with the code editor open to the `MainActivity.kt` file. The code implements a `MainActivity` that extends `AppCompatActivity`. It overrides the `onCreate` method to enable edge-to-edge display and set the content view to `activity_main`. Inside the `onCreate` method, it calls `enableEdgeToEdge()` and sets padding for the system bars. The code editor has syntax highlighting for Java/Kotlin and includes standard tools like copy, paste, and search.

```
1 package com.example.diceroller
2
3 import android.os.Bundle
4 import androidx.activity.enableEdgeToEdge
5 import androidx.appcompat.app.AppCompatActivity
6 import android.core.view.ViewCompat
7 import android.core.view.WindowInsetsCompat
8
9 class MainActivity : AppCompatActivity() {
10     override fun onCreate(savedInstanceState: Bundle?) {
11         super.onCreate(savedInstanceState)
12         enableEdgeToEdge()
13         setContentView(R.layout.activity_main)
14         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
15             val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
16             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
17             insets
18         }
19     }
20 }
```

2.3.2. Enable auto imports

File > Settings > Editor > General > Auto Import. Trong mục Java và Kotlin, tích chọn cả 2 mục Add unambiguous imports on the fly và Optimize imports on the fly (for current project)

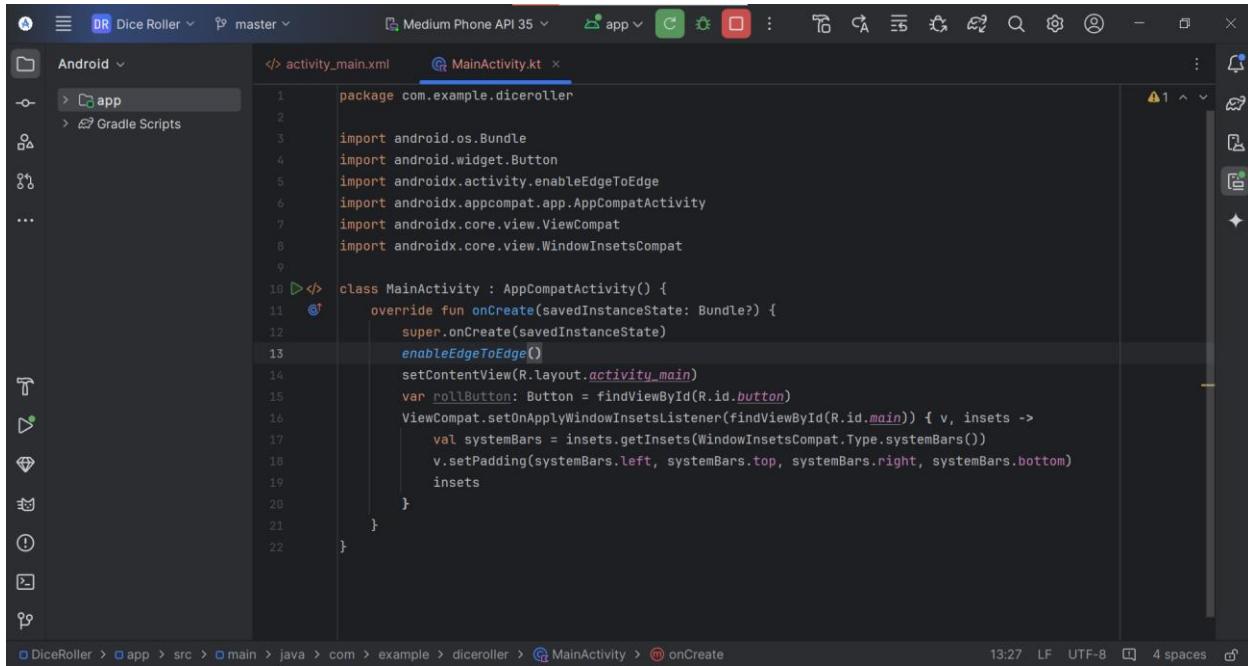


2.4. Make the Button interactive

2.4.1. Display a message when the Button is clicked

Các bước thực hiện

1. Thêm đoạn code màu vào phương thức onCreate() sau khi gọi setContentView()



```
DR Dice Roller master Medium Phone API 35 app MainActivity.kt

1 package com.example.diceroller
2
3 import android.os.Bundle
4 import android.widget.Button
5 import androidx.activity.enableEdgeToEdge
6 import androidx.appcompat.app.AppCompatActivity
7 import androidx.core.view.ViewCompat
8 import androidx.core.view.WindowInsetsCompat
9
10 class MainActivity : AppCompatActivity() {
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         enableEdgeToEdge()
14         setContentView(R.layout.activity_main)
15         var rollButton: Button = findViewById(R.id.button)
16         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
17             val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
18             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
19         }
20     }
21 }
22 }
```

2. Kiểm tra import

```
3 import android.os.Bundle
4 import android.widget.Button
5 import androidx.activity.enableEdgeToEdge
6 import androidx.appcompat.app.AppCompatActivity
7 import androidx.core.view.ViewCompat
8 import androidx.core.view.WindowInsetsCompat
```

3. Dùng đối tượng rollButton và đặt click listener bằng cách gọi phương thức setOnClickListener()

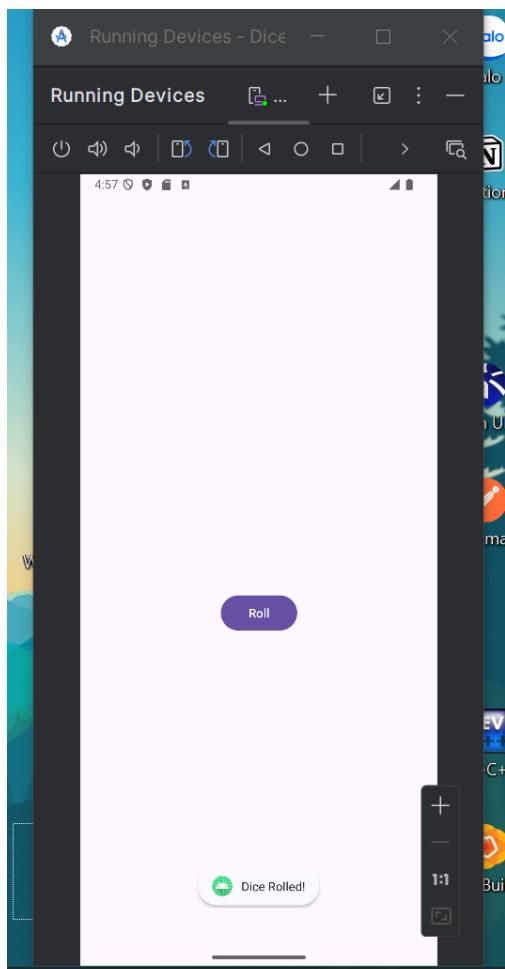
4. Thêm code màu

The screenshot shows the Android Studio interface with the code editor open. The file being edited is `MainActivity.kt`. The code implements a `MainActivity` that extends `AppCompatActivity`. It overrides the `onCreate` method to set up the activity's content view and handle window insets. It also sets an `onClickListener` for a button named `rollButton` to display a `Toast` message "Dice Rolled!" when clicked.

```
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        var rollButton: Button = findViewById(R.id.button)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }
        rollButton.setOnClickListener{
            var toast = Toast.makeText(context: this, text: "Dice Rolled!", Toast.LENGTH_SHORT)
            toast.show()
        }
    }
}
```

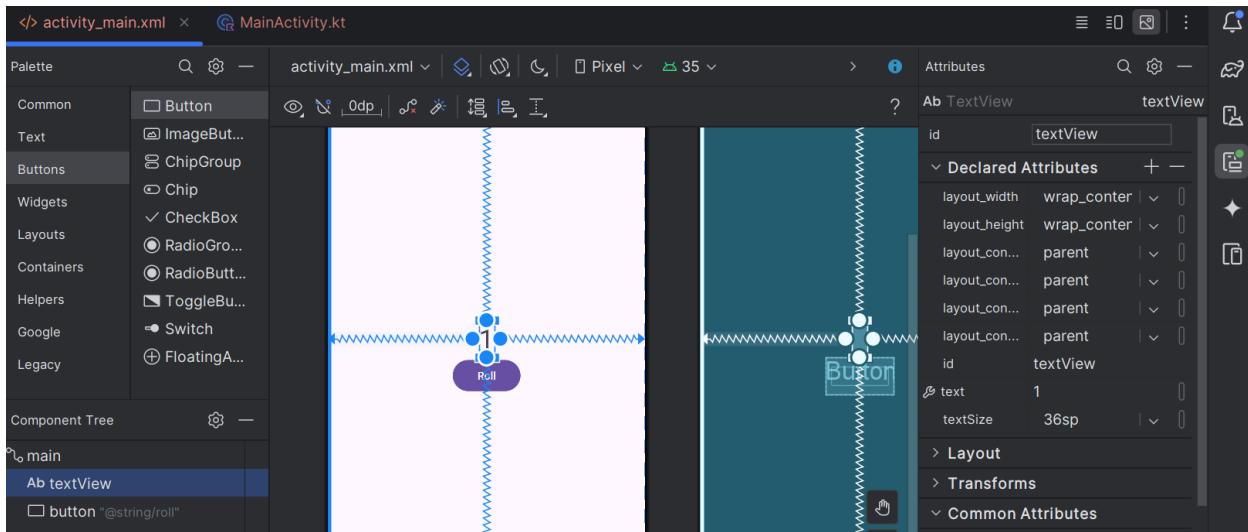
5. Chạy chương trình vào nhấn nút Roll



2.4.2. Update the TextView when the Button is clicked

Các bước thực hiện:

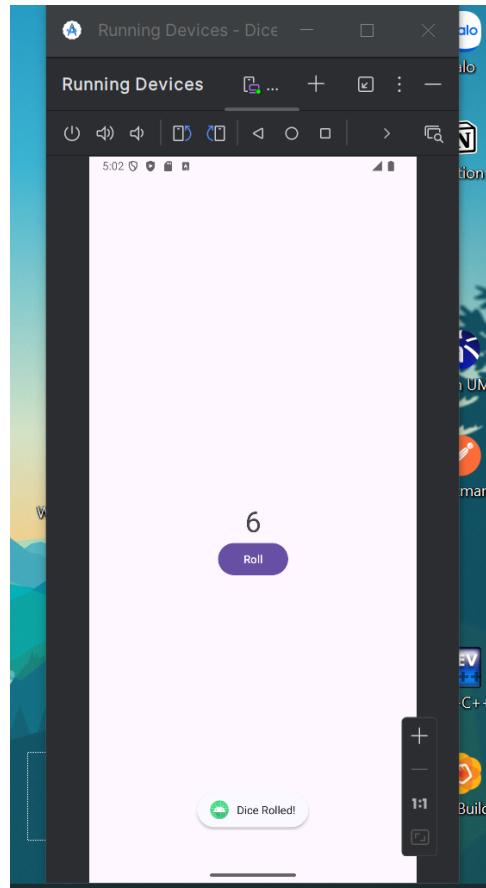
1. Chuyển về activity_main.xml
2. Chọn TextView, kiểm tra id là textView



3. Mở MainActivity.kt, tạo biến tên resultsTextView để lưu TextView
4. Dùng findViewById() để tìm textView, đặt text cho resultTextView là 6

```
23
24     rollButton.setOnClickListener{
25         var toast = Toast.makeText(context: this, text: "Dice Rolled!", Toast.LENGTH_SHORT)
26         toast.show()
27         val resultsTextView: TextView = findViewById(R.id.textView)
28         resultsTextView.text = "6"
29     }
}
```

5. Chạy chương trình



2.5. Add the dice roll logic

2.5.1. Add the Dice class

1. Tạo lớp Dice với phương thức roll()

The screenshot shows the Android Studio code editor with the following details:

- Project Structure:** The left sidebar shows the project structure under "Android". It includes the "app" module with "manifests", "kotlin+java", and "com.example.diceroller" package. Inside "com.example.diceroller", the "Dice" class is selected and highlighted in blue.
- Code Editor:** The main area shows the "Dice.kt" file. The code defines a class "Dice" with a constructor taking "numSides" and a "roll()" method returning a random integer between 1 and "numSides".

```
1 package com.example.diceroller
2
3 class Dice(val numSides: Int) {
4     fun roll(): Int {
5         return (1..numSides).random()
6     }
7 }
```
- Toolbars and Status Bar:** The top of the screen shows the Android Studio toolbar with icons for file operations, navigation, and search. The status bar indicates the device is a "Medium Phone API 35" and the branch is "master".

2.5.2. Create a rollDice() method

1. Bỏ phần code trong click listener bằng việc gọi tới rollDice()

```
21
22     rollButton.setOnClickListener{
23         // var toast = Toast.makeText(this, "Dice Rolled!", Toast.LENGTH_SHORT)
24         // toast.show()
25         // val resultsTextView: TextView = findViewById(R.id.textView)
26         // resultsTextView.text = "6";
27         rollDice()
28     }
29
30     // var toast = Toast.makeText(this, "Dice Rolled!", Toast.LENGTH_SHORT)
31     // toast.show()
32     // val resultsTextView: TextView = findViewById(R.id.textView)
33     // resultsTextView.text = "6";
34     rollDice()
35
36     ? Create function 'rollDice'
37     ? Rename reference
```

The screenshot shows a portion of Java code within an Android Studio editor. Line 27 contains the call to 'rollDice()' which is underlined with a red squiggle, indicating it's undefined. A context menu is open at this position with two options: 'Create function 'rollDice'' and 'Rename reference'.

2. Chọn Create function 'rollDice'

```
31
32     private fun rollDice() {
33         TODO(reason: "Not yet implemented")
34     }
35 }
```

The screenshot shows the Java code after creating the 'rollDice()' function. Line 33 now contains a 'TODO' comment with the reason 'Not yet implemented'.

2.5.3. Create a new Dice object instance

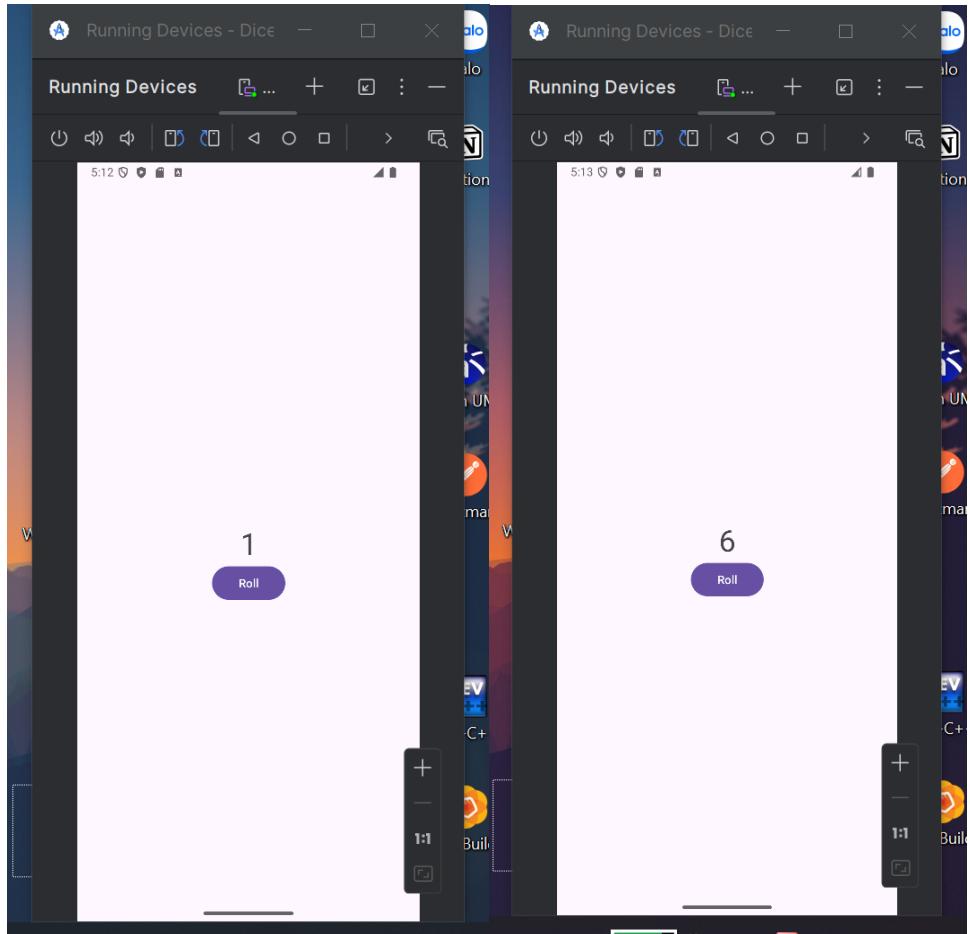
Các bước thực hiện

1. Trong rollDice(), xóa TODO()
2. Thêm code để tạo ra dice với 6 mặt, tìm TextView với findViewById(), chuyển diceRoll thành một chuỗi bằng toString()

```
31
32
33     private fun rollDice() {
34         val dice = Dice( numSides: 6)
35         val diceRoll = dice.roll();
36         val resultsTextView: TextView = findViewById(R.id.textView)
37         resultsTextView.text = diceRoll.toString();
38     }
39 }
```

The screenshot shows the Java code with the 'rollDice()' function fully implemented. It creates a new 'Dice' object with 6 sides, rolls it, and then sets the result as a string to the TextView.

3. Chạy chương trình



2.6. Adopt good coding practices

2.6.1. Clean up your code

Sửa lại code phần rollButton

```
24     /////
25     /////
26     /////
27     /////
28     //    var toast = Toast.makeText(this, "Dice Rolled!", Toast.LENGTH_SHORT)
29     //    toast.show()
30     //    val resultsTextView: TextView = findViewById(R.id.textView)
31     //    resultsTextView.text = "6";
32     //    rollDice()
33     //}
34     //    rollButton.setOnClickListener { rollDice() }
```

3. Add images to the Dice Roller app

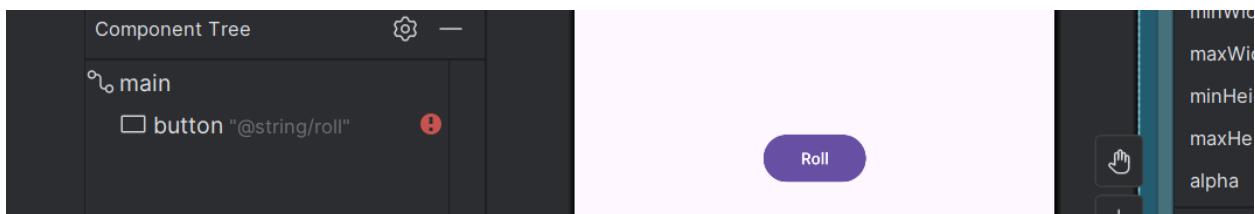
3.1. Update the layout for the app

3.1.1. Open Dice Roller app

1. Mở app Dice Roller từ lab 4.3
2. Mở activity_main.xml

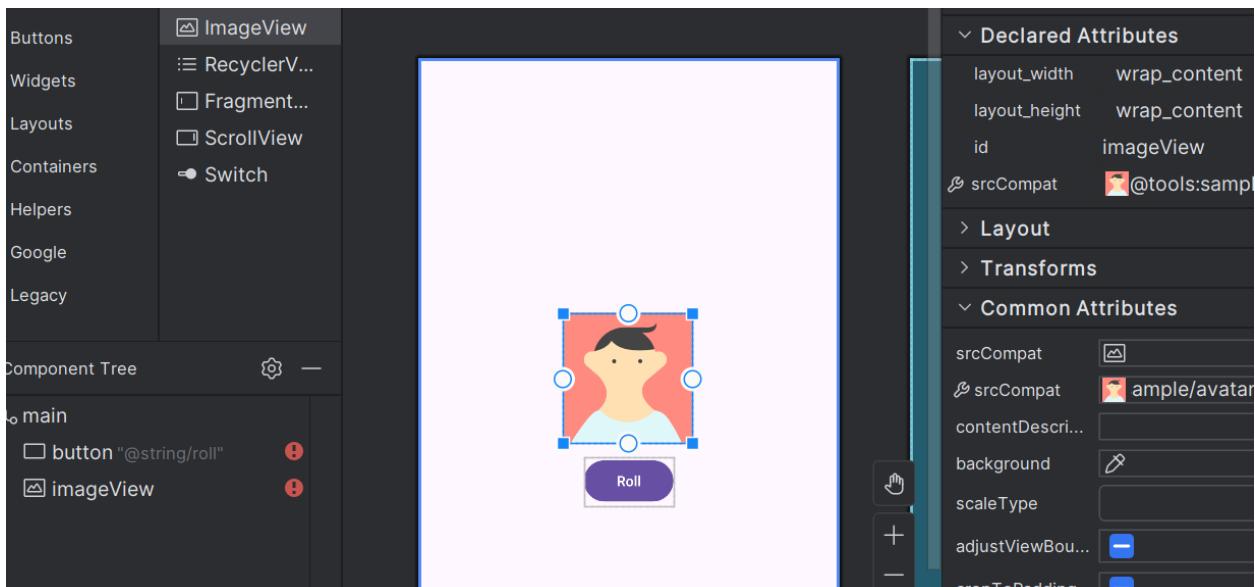
3.1.2. Delete the TextView

1. Trong Layout Editor, chọn TextView trong Component Tree
2. Chọn Delete để xóa TextView



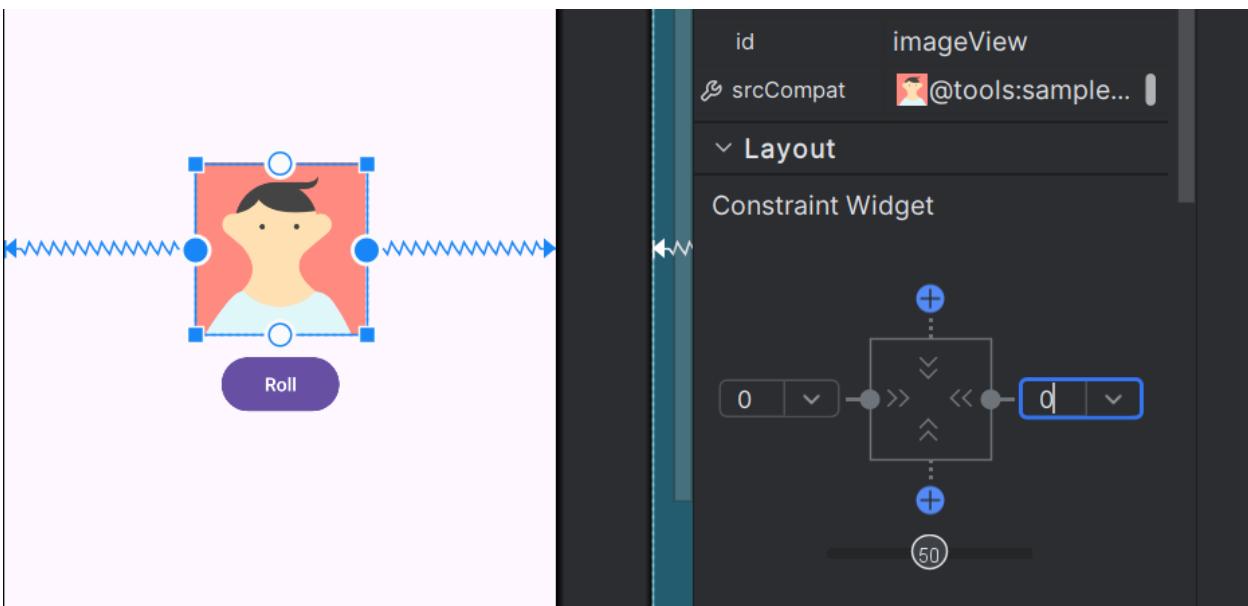
3.1.3. Add an ImageView to the layout

1. Kéo một ImageView từ Palette vào Design view, để nó ở trên Button

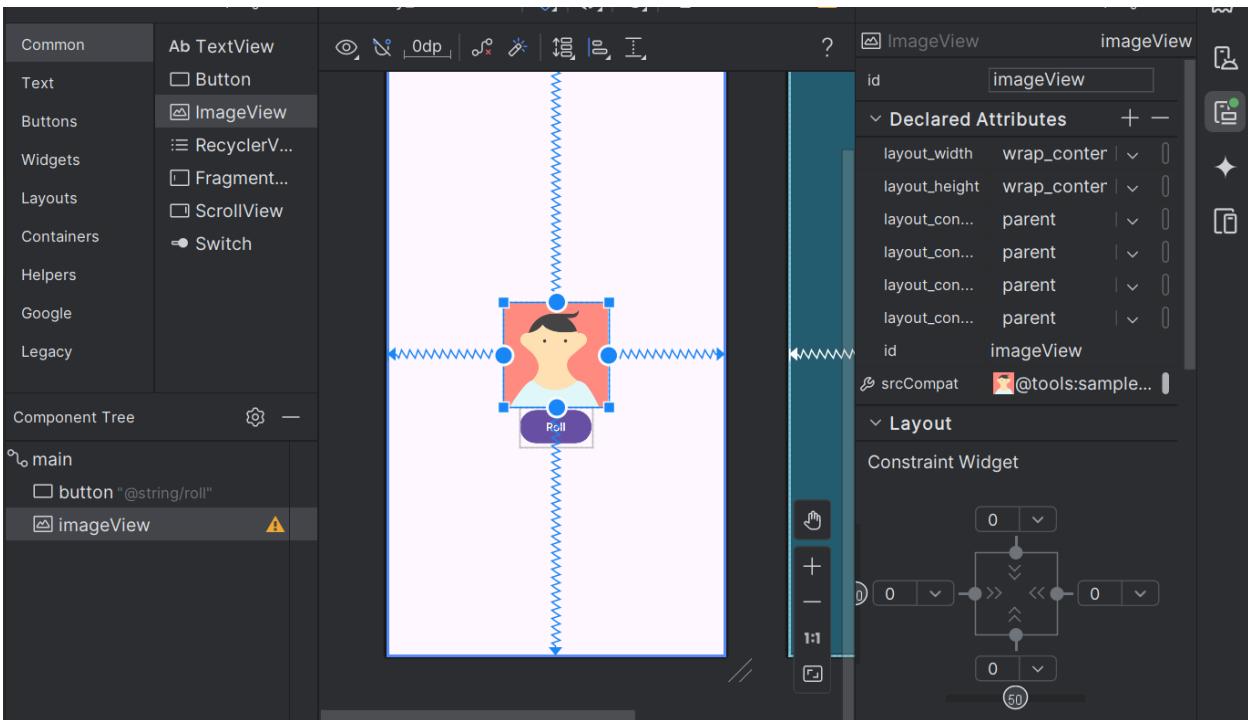


3.1.4. Position the ImageView and Button

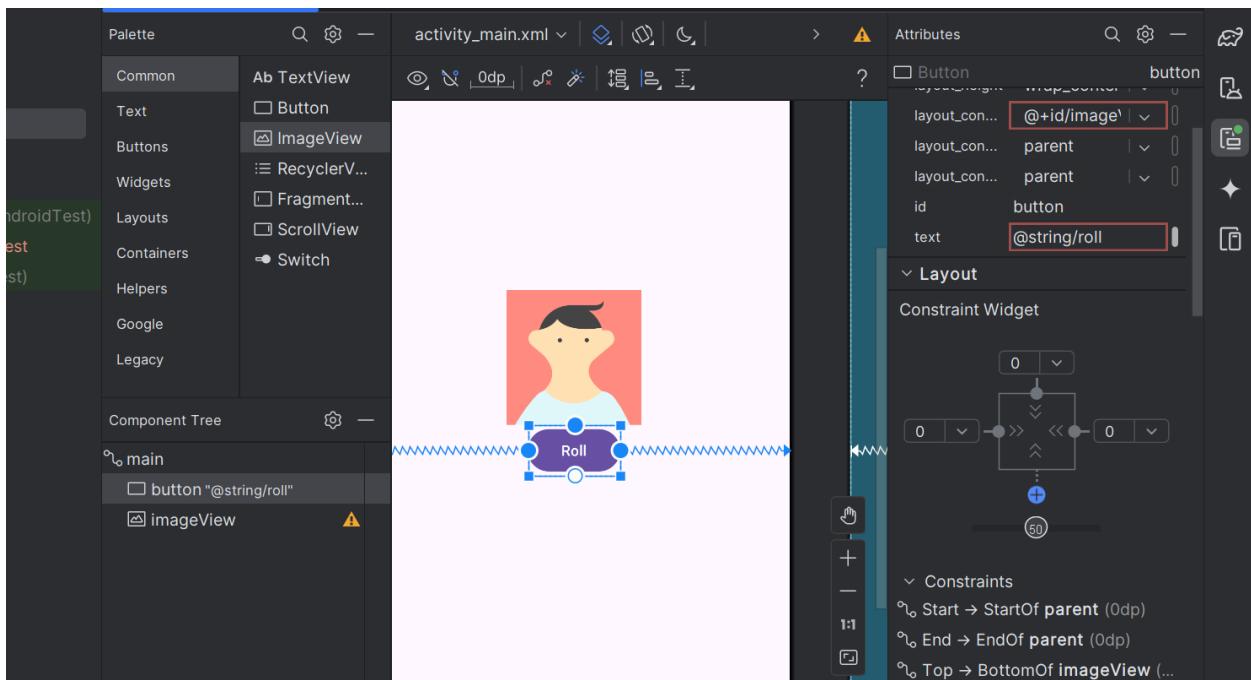
1. Nối viền trái của ImageView với lề trái, tương tự với lề phải



2. Nối viền trên với lề trên, nối viền trên của Button với viền dưới của ImageView



3. Nối viền dưới của ImageView với lề dưới

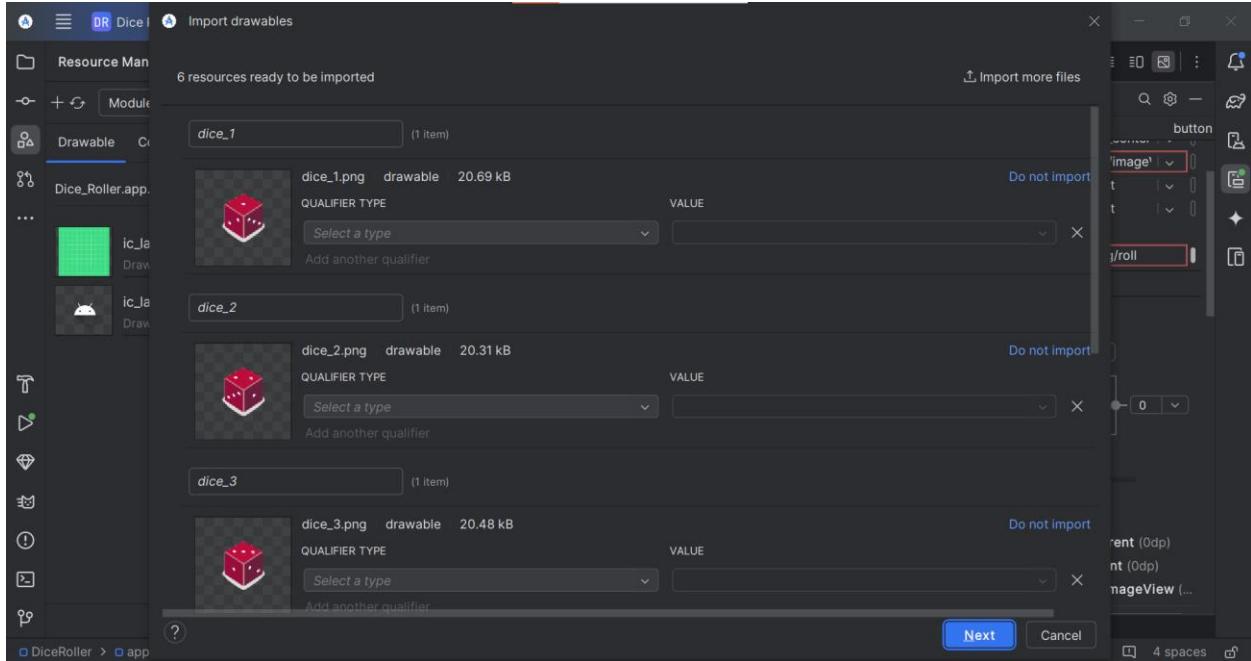


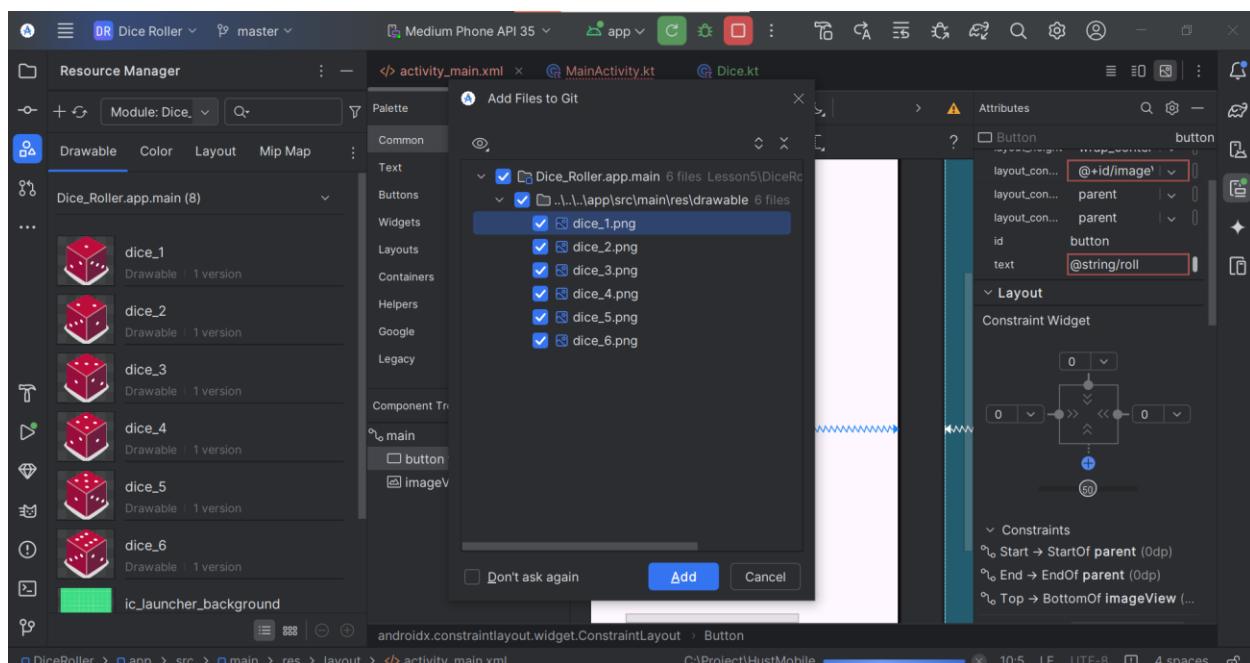
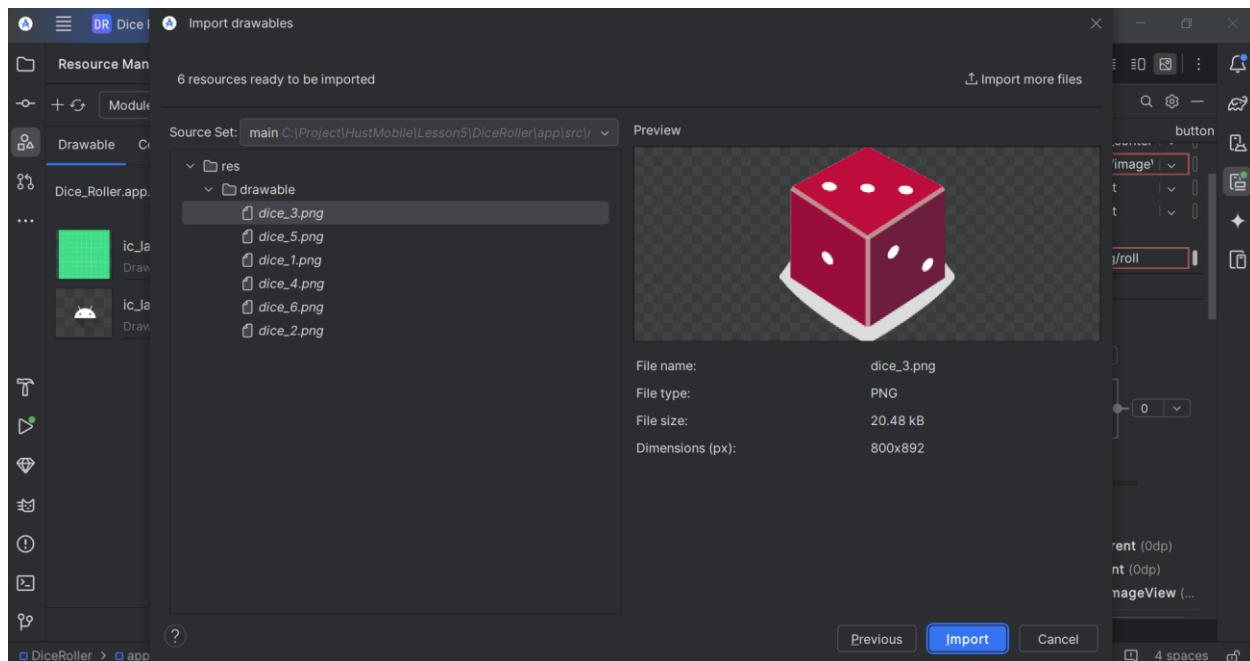
3.2. Add the dice images

3.2.1. Download the dice images

3.2.2. Add dice images to your app

1. Chọn View > Tool Windows > Resource Manager. Chọn Import Drawables
2. Import 6 hình đã tải về

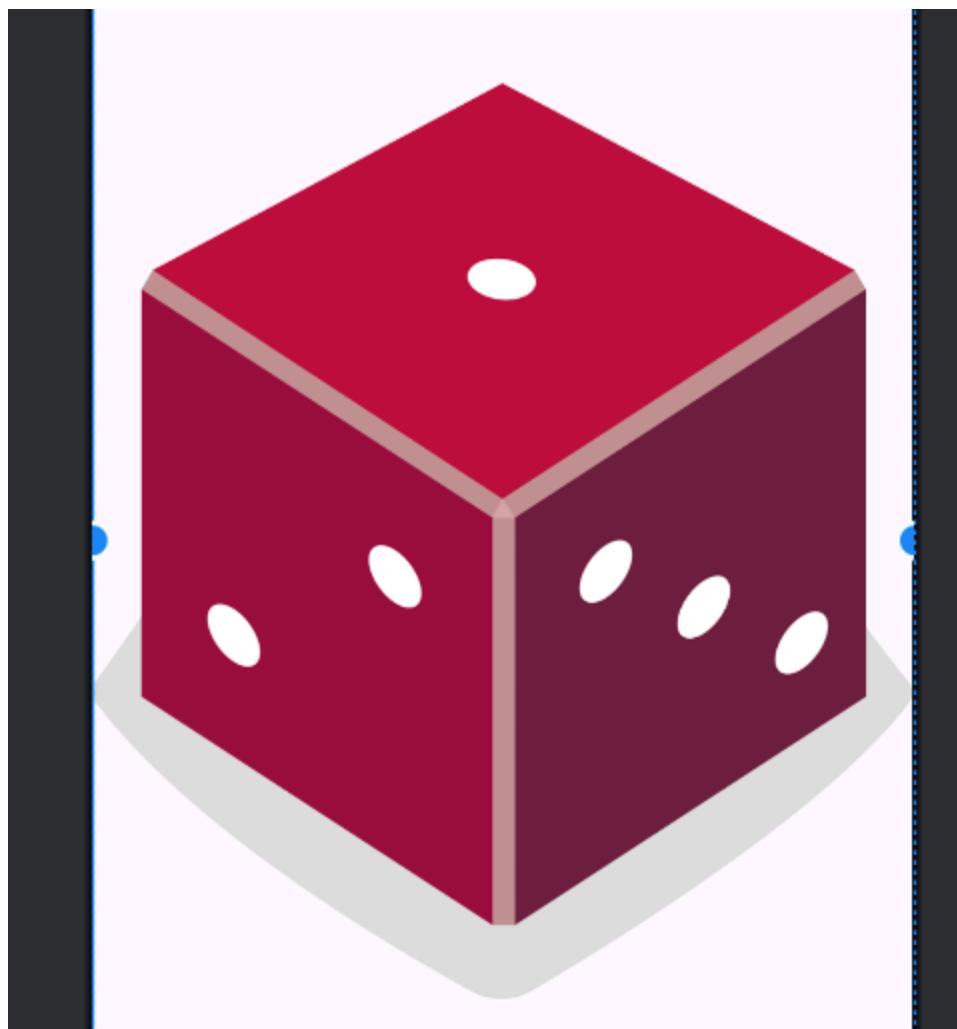




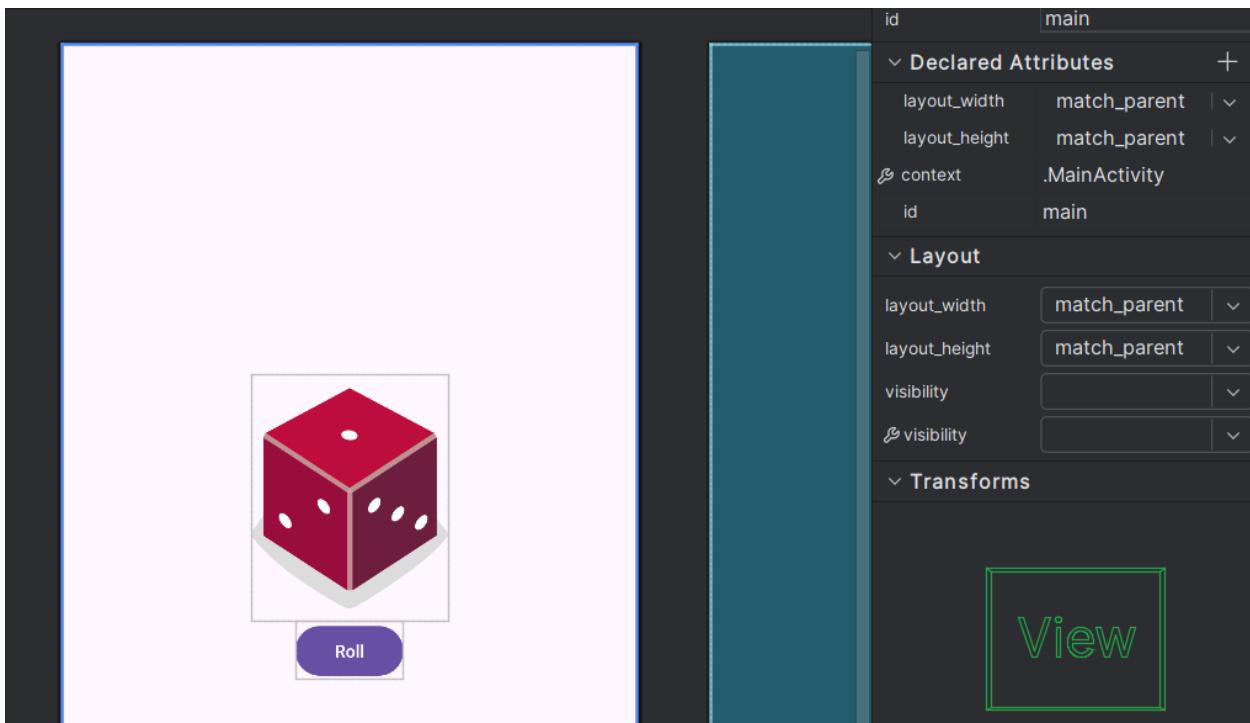
3.3. Use the dice images

3.3.1. Replace the sample avatar image

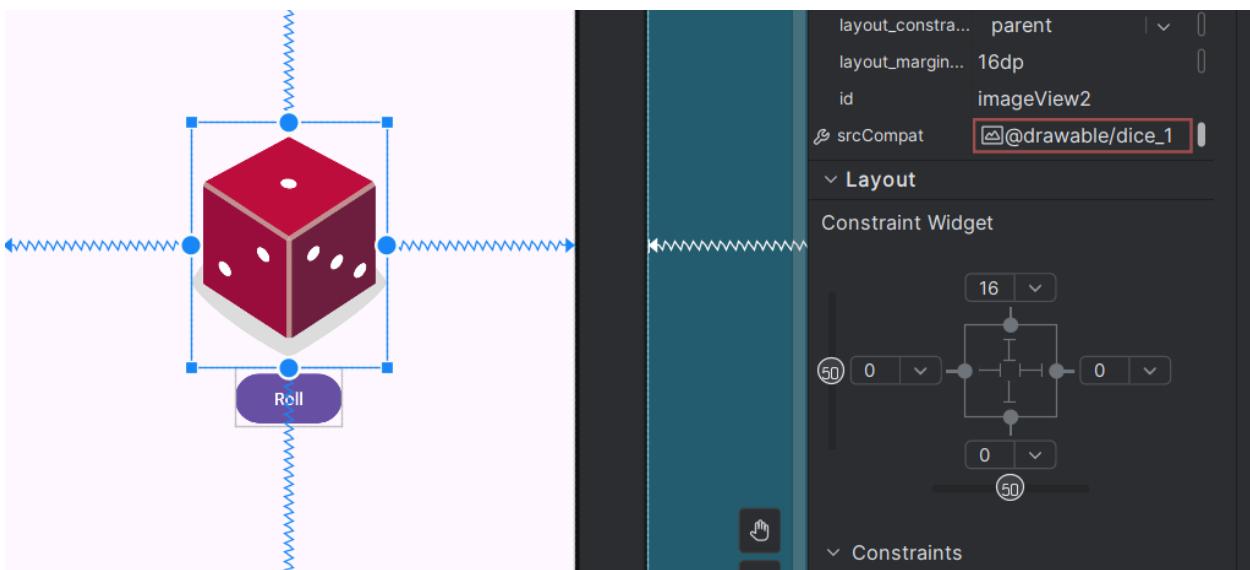
1. Trong Design Editor, chọn ImageView
2. Trong Declares Attributes, tìm srcCompat
3. Chọn preview của avatar, chọn dice_1



4. Chuyển `layout_width` thành 160dp và `layout_height` thành 200dp



5. Chính Constraint Widget



3.3.2. Change the dice image when the button is clicked

1. Mở MainActivity.kt
2. Trong phương thức rollDice(), chọn bất kỳ phần code nào liên quan đến TextView và xóa nó

```
33     private fun rollDice() {
34         val dice = Dice( numSides: 6)
35         val diceRoll = dice.roll();
36     }
37 }
```

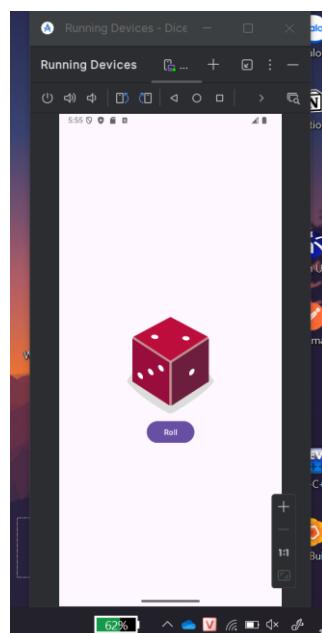
3. Trong rollDice(), tạo một biến mới tên diceImage có kiểu là ImageView, đặt giá trị của nó bằng ImageView từ layout. Dùng findViewById() và truyền vào ID nguồn cho ImageView

```
33
34     private fun rollDice() {
35         val dice = Dice( numSides: 6)
36         val diceRoll = dice.roll();
37         val diceImage: ImageView = findViewById(R.id.imageView)
38     }
39 }
```

4. Thêm code mẫu vào để kiểm tra đã cập nhật đúng ImageView khi button được click

```
34     private fun rollDice() {
35         val dice = Dice( numSides: 6)
36         val diceRoll = dice.roll();
37         // val resultsTextView: TextView = findViewById(R.id.textView)
38         // resultsTextView.text = diceRoll.toString();
39         val diceImage: ImageView = findViewById(R.id.imageView)
40         diceImage.setImageResource(R.drawable.dice_2)
41 }
```

5. Chạy chương trình



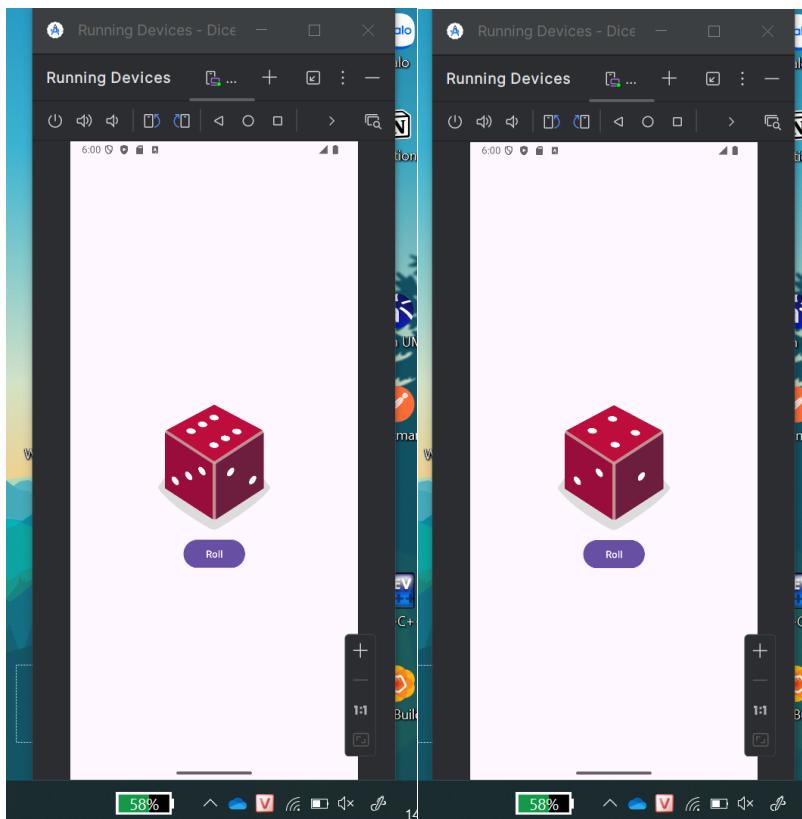
3.4. Display the correct dice image based on the dice roll

3.4.1. Update the rollDice() method

- Trong phương thức rollDice(), xóa dòng code đặt nguồn ID ảnh
- Thay dòng code đó bằng câu lệnh when để cập nhật ImageView dựa trên giá trị diceRoll

```
34     private fun rollDice() {  
35         val dice = Dice( numSides: 6)  
36         val diceRoll = dice.roll();  
37         //  
38         //  
39         val resultsTextView: TextView = findViewById(R.id.textView)  
40         resultsTextView.text = diceRoll.toString();  
41         val diceImage: ImageView = findViewById(R.id.imageView)  
42         diceImage.setImageResource(R.drawable.dice_2)  
43         when (diceRoll){  
44             1 -> diceImage.setImageResource(R.drawable.dice_1)  
45             2 -> diceImage.setImageResource(R.drawable.dice_2)  
46             3 -> diceImage.setImageResource(R.drawable.dice_3)  
47             4 -> diceImage.setImageResource(R.drawable.dice_4)  
48             5 -> diceImage.setImageResource(R.drawable.dice_5)  
49             6 -> diceImage.setImageResource(R.drawable.dice_6)  
50         }  
51     }
```

3. Chạy chương trình

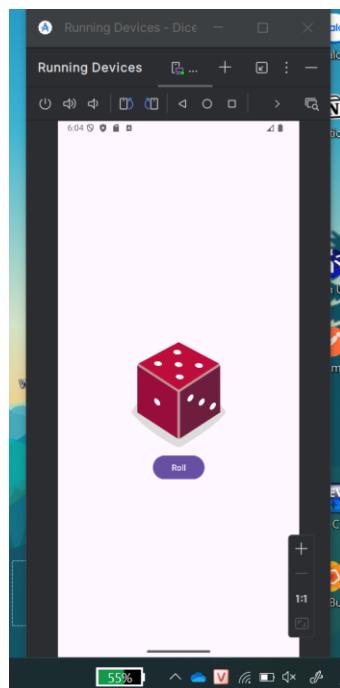


3.4.2. Optimize your code

- Thay đổi đoạn code với lệnh when bằng code mẫu

```
private fun rollDice() {  
    // Create new Dice object with 6 sides and roll it  
    val dice = Dice( numSides: 6)  
    val diceRoll = dice.roll()  
    val diceImage: ImageView = findViewById(R.id.imageView2)  
    val drawableResource = when (diceRoll) {  
        1 -> R.drawable.dice_1  
        2 -> R.drawable.dice_2  
        3 -> R.drawable.dice_3  
        4 -> R.drawable.dice_4  
        5 -> R.drawable.dice_5  
        else-> R.drawable.dice_6  
    }  
    diceImage.setImageResource(drawableResource)
```

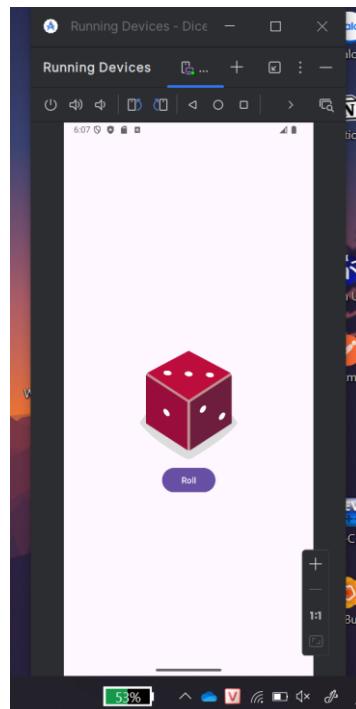
- Chạy chương trình



3.4.3. Set an appropriate content description on the ImageView

Thêm đoạn code mẫu và chạy chương trình

```
34     private fun rollDice() {  
37         //      val resultsTextView: TextView = findViewById(R.id.textView)  
38         //      resultsTextView.text = diceRoll.toString();  
39         val diceImage: ImageView = findViewById(R.id.imageView)  
40         //      diceImage.setImageResource(R.drawable.dice_2)  
41         //      when (diceRoll){  
42             //          1 -> diceImage.setImageResource(R.drawable.dice_1)  
43             //          2 -> diceImage.setImageResource(R.drawable.dice_2)  
44             //          3 -> diceImage.setImageResource(R.drawable.dice_3)  
45             //          4 -> diceImage.setImageResource(R.drawable.dice_4)  
46             //          5 -> diceImage.setImageResource(R.drawable.dice_5)  
47             //          6 -> diceImage.setImageResource(R.drawable.dice_6)  
48         //      }  
49         val drawableResource = when (diceRoll) {  
50             //          1 -> R.drawable.dice_1  
51             //          2 -> R.drawable.dice_2  
52             //          3 -> R.drawable.dice_3  
53             //          4 -> R.drawable.dice_4  
54             //          5 -> R.drawable.dice_5  
55             //          else -> R.drawable.dice_6  
56         }  
57         diceImage.setImageResource(drawableResource)  
58         diceImage.contentDescription = diceRoll.toString()  
59     }  
60 }
```



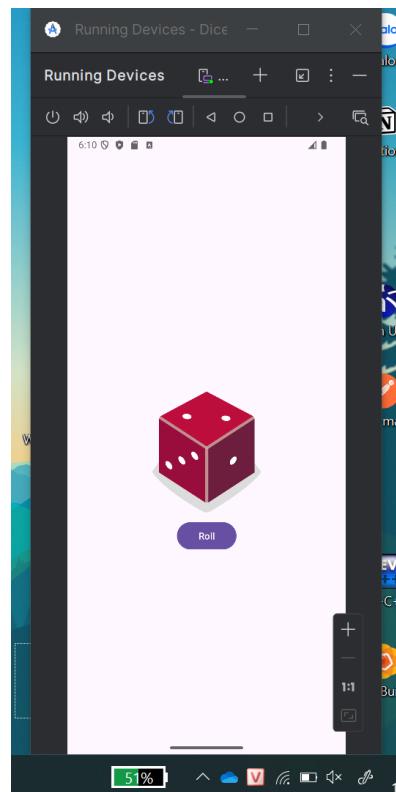
3.5. Adopt good coding practices

3.5.1. Create a more useful launch experience

Thêm hình ảnh xúc sắc khi vừa mở app

```
11 >< class MainActivity : AppCompatActivity() {
12     override fun onCreate(savedInstanceState: Bundle?) {
13         super.onCreate(savedInstanceState)
14         enableEdgeToEdge()
15         setContentView(R.layout.activity_main)
16         var rollButton: Button = findViewById(R.id.button)
17         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
18             val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
19             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
20             insets
21         }
22
23         // rollButton.setOnClickListener{
24         //     var toast = Toast.makeText(this, "Dice Rolled!", Toast.LENGTH_SHORT)
25         //     toast.show()
26         //     val resultsTextView: TextView = findViewById(R.id.textView)
27         //     resultsTextView.text = "6";
28         //     rollDice()
29         //}
30         rollButton.setOnClickListener { rollDice() }
31     }

```



3.5.2. Comment your code

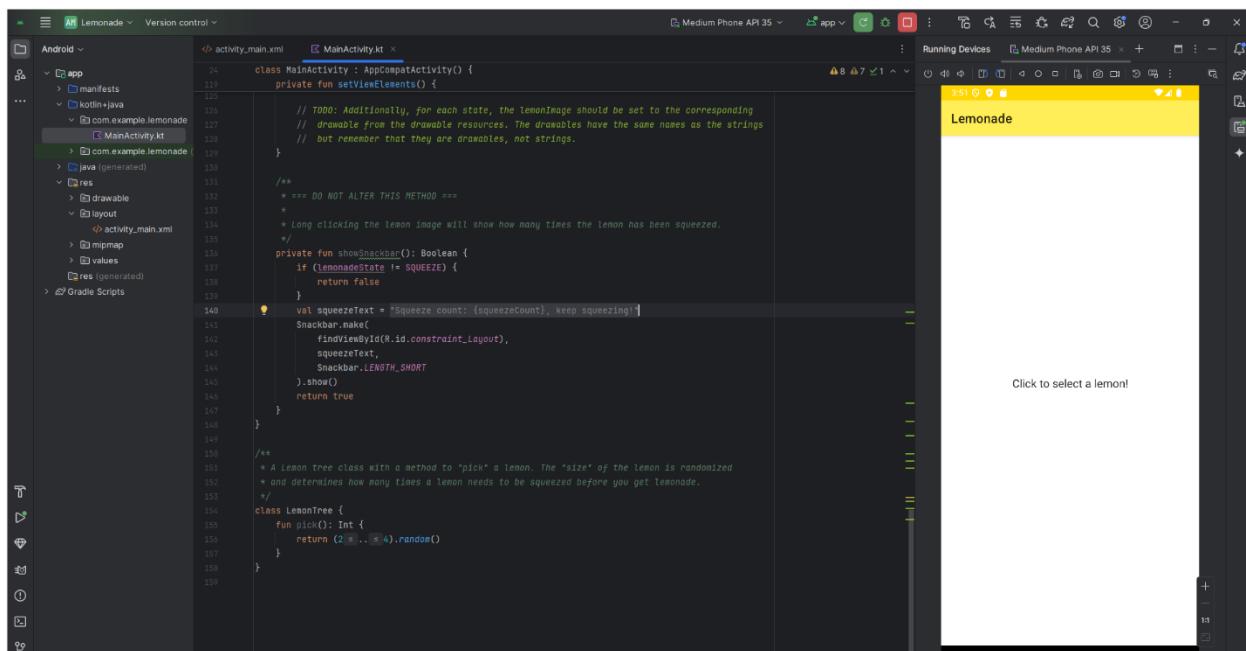
4. Project: Lemonade app

4.1. App overview

- Ứng dụng Lemonade sau khi hoàn thiện sẽ có một màn hình duy nhất. Trong lần đầu người dùng mở ứng dụng, ứng dụng sẽ chào đón bằng lời nháy nháy vào hình ảnh cây chanh để chọn một quả chanh.
- Khi nháy vào cây chanh, người dùng sẽ thấy một quả chanh. Họ có thể nháy để "vắt" quả chanh này với số lần không xác định trước (số lần vắt cần thiết sẽ được tạo ngẫu nhiên) trước khi chuyển sang màn hình tiếp theo.
- Khi người dùng đã nháy vắt quả chanh đủ số lần, họ sẽ thấy hình ảnh một chiếc cốc để "uống" nước chanh.
- Sau khi người dùng nhấp vào hình ảnh để quay lại màn hình đầu tiên rồi chọn một quả chanh khác trên cây.

4.2. Get started

Download mã nguồn trên git và open project trong Android Studio:



4.3. Build your user interface

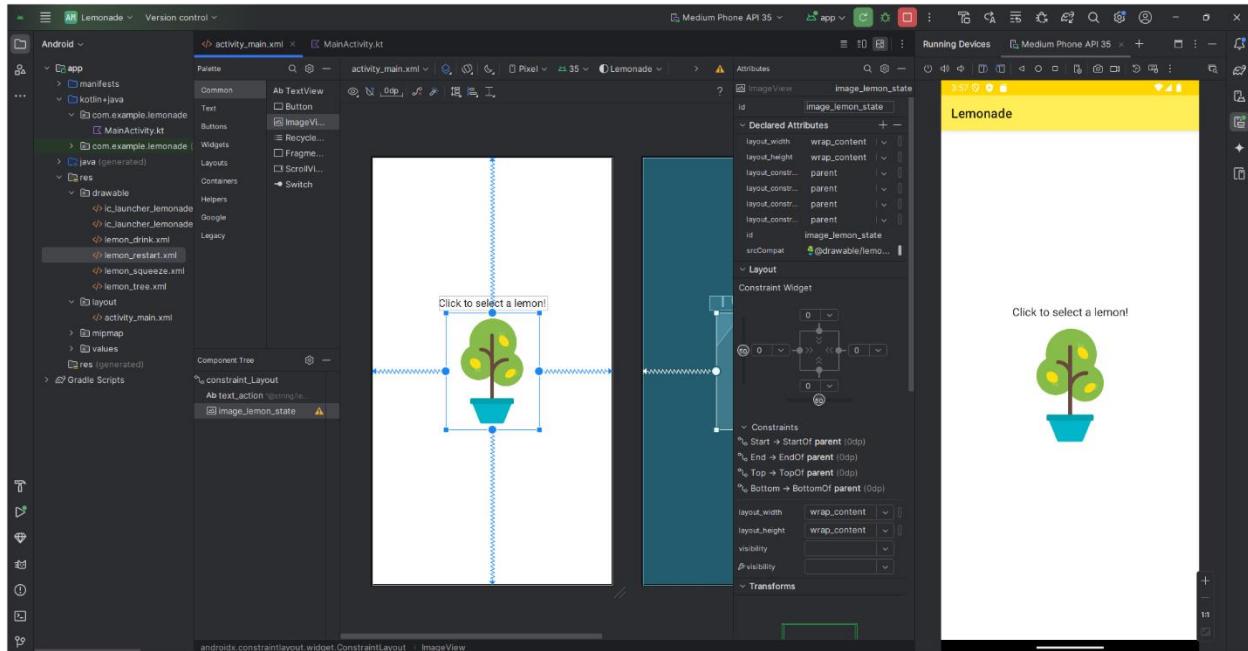
Ứng dụng Lemonade chỉ yêu cầu bộ cục cơ bản; bạn chỉ cần hai chế độ xem để triển khai tất cả chức năng của ứng dụng này.

- Một TextView đưa ra hướng dẫn cho người dùng.

2. Một ImageView hiển thị hình ảnh đồ họa theo trạng thái hiện tại của ứng dụng (ví dụ: một quả chanh để vắt).

Tạo ra một bộ cục tương tự như hình ảnh dưới đây, có cả hai khung hiển thị ở giữa màn hình và TextView ở trên ImageView.

Kết quả:



4.4. Make your app interactive

Cập nhật hàm clickLemonImage để cập nhật trạng thái của hình ảnh khi select

```

92     private fun clickLemonImage() {
93         when (lemonadeState) {
94             "select" -> {
95                 // Chọn chanh từ cây
96                 lemonSize = LemonTree().pick()
97                 squeezeCount = 0
98                 lemonadeState = "squeeze"
99             }
100            "squeeze" -> {
101                // Ép chanh cho đến khi không còn nước
102                squeezeCount++
103                lemonSize--
104                if (lemonSize == 0) {
105                    lemonadeState = "drink"
106                }
107            }
108            "drink" -> {
109                // Uống nước chanh
110                lemonadeState = "restart"
111            }
112            "restart" -> {
113                // Bắt đầu lại
114                lemonadeState = "select"
115            }
116        }
117        setViewElements() // Cập nhật giao diện sau khi thay đổi trạng thái
118    }
...

```

Cập nhật hàm setViewElements() để cập nhật hình ảnh và text phù hợp với trạng thái

```

123     private fun setViewElements() {
124         val textAction: TextView = findViewById(R.id.text_action)
125
126         // Cập nhật nội dung text dựa vào trạng thái hiện tại
127         val actionText = when (lemonadeState) {
128             "select" -> getString(R.string.lemon_select)
129             "squeeze" -> getString(R.string.lemon_squeeze)
130             "drink" -> getString(R.string.lemon_drink)
131             "restart" -> getString(R.string.lemon_empty_glass)
132             else -> ""
133         }
134         textAction.text = actionText
135
136         // Cập nhật ảnh của chanh dựa trên trạng thái hiện tại
137         val lemonImageRes = when (lemonadeState) {
138             "select" -> R.drawable.lemon_tree
139             "squeeze" -> R.drawable.lemon_squeeze
140             "drink" -> R.drawable.lemon_drink
141             "restart" -> R.drawable.lemon_restart
142             else -> R.drawable.lemon_tree
143         }
144         lemonImage!!.setImageResource(lemonImageRes)
145     }
...

```

4.5. Run your app

Kết quả khi chạy chương trình:

Android Studio screenshot showing the code for the Lemonade app's MainActivity.kt and its corresponding UI on a virtual device.

MainActivity.kt

```
class MainActivity : AppCompatActivity() {
    private var lemonadeState = "select"
    // Default lemonsSize to -1
    private var lemonSize = -1
    // Default the squeezeCount to -1
    private var squeezeCount = -1

    private var lemonTree = LemonTree()
    private var lemonImage: ImageView? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // *** DO NOT ALTER THE CODE IN THE FOLLOWING IF STATEMENT ***
        if (savedInstanceState != null) {
            lemonadeState = savedInstanceState.getString(LEMONADE_STATE, defaultValue: "select")
            lemonSize = savedInstanceState.getInt(LEMON_SIZE, defaultValue: -1)
            squeezeCount = savedInstanceState.getInt(SQUEEZE_COUNT, defaultValue: -1)
        }
        // *** END IF STATEMENT ***

        lemonImage = findViewById(R.id.image_lemon_state)
        setViewElements()
        lemonImage!!.setOnClickListener {
            clickLemonImage() // Löt hàn xử lý click
        }
        lemonImage!!.setOnLongClickListener {
            showSnackbar() // Hiển thị số lần ép chanh nếu hợp lệ
        }
    }

    /**
     * *** DO NOT ALTER THIS METHOD ***
     *
     * This method saves the state of the app if it is put in the background.
     */
    override fun onSaveInstanceState(outState: Bundle) {
        outState.putString(LEMONADE_STATE, lemonadeState)
        outState.putInt(LEMON_SIZE, lemonSize)
        outState.putInt(SQUEEZE_COUNT, squeezeCount)
    }
}
```

Running Devices shows a virtual device with the app running. The UI displays a lemon tree in a pot with the text "Click to select a lemon!" above it.

Android Studio screenshot showing the code for the Lemonade app's MainActivity.kt and its corresponding UI on a virtual device.

MainActivity.kt

```
class MainActivity : AppCompatActivity() {
    private var lemonadeState = "select"
    // Default lemonsSize to -1
    private var lemonSize = -1
    // Default the squeezeCount to -1
    private var squeezeCount = -1

    private var lemonTree = LemonTree()
    private var lemonImage: ImageView? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // *** DO NOT ALTER THE CODE IN THE FOLLOWING IF STATEMENT ***
        if (savedInstanceState != null) {
            lemonadeState = savedInstanceState.getString(LEMONADE_STATE, defaultValue: "select")
            lemonSize = savedInstanceState.getInt(LEMON_SIZE, defaultValue: -1)
            squeezeCount = savedInstanceState.getInt(SQUEEZE_COUNT, defaultValue: -1)
        }
        // *** END IF STATEMENT ***

        lemonImage = findViewById(R.id.image_lemon_state)
        setViewElements()
        lemonImage!!.setOnClickListener {
            clickLemonImage() // Löt hàn xử lý click
        }
        lemonImage!!.setOnLongClickListener {
            showSnackbar() // Hiển thị số lần ép chanh nếu hợp lệ
        }
    }

    /**
     * *** DO NOT ALTER THIS METHOD ***
     *
     * This method saves the state of the app if it is put in the background.
     */
    override fun onSaveInstanceState(outState: Bundle) {
        outState.putString(LEMONADE_STATE, lemonadeState)
        outState.putInt(LEMON_SIZE, lemonSize)
        outState.putInt(SQUEEZE_COUNT, squeezeCount)
    }
}
```

Running Devices shows a virtual device with the app running. The UI displays a lemon slice icon with the text "Click to juice the lemon!" above it.

Activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<layout>
    <activity android:name=".MainActivity" android:label="Lemonade">
        <activity_main>
            <include layout="@+id/ic_launcher_lemonade_foreground.xml" />
        </activity_main>
    </activity>
</layout>

```

MainActivity.kt

```

class MainActivity : AppCompatActivity() {
    private var lemonadeState = "select"
    // Default var lemonSize to -1
    private var lemonSize = -1
    // Default the squeezeCount to -1
    private var squeezeCount = -1

    private var lemonTree = LemonTree()
    private var lemonImage: ImageView? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // *** DO NOT ALTER THE CODE IN THE FOLLOWING IF STATEMENT ***
        if (savedInstanceState != null) {
            lemonadeState = savedInstanceState.getString(LEMONADE_STATE, defaultValue = "select")
            lemonSize = savedInstanceState.getInt(LEMON_SIZE, defaultValue = -1)
            squeezeCount = savedInstanceState.getInt(SQUEEZE_COUNT, defaultValue = -1)
        }
        // *** END IF STATEMENT ***

        lemonImage = findViewById(R.id.image_lemon_state)
        setViewElements()
        lemonImage!!.setOnClickListener {
            clickLemonImage() // gọi hàm xử lý click
        }
        lemonImage!!.setOnLongClickListener {
            showSnackbar() // hiển thị số lần ép chanh nếu hợp lệ
        }
    }

    /**
     * *** DO NOT ALTER THIS METHOD ***
     *
     * This method saves the state of the app if it is put in the background.
     */
    override fun onSaveInstanceState(outState: Bundle) {
        outState.putString(LEMONADE_STATE, lemonadeState)
        outState.putInt(LEMON_SIZE, lemonSize)
        outState.putInt(SQUEEZE_COUNT, squeezeCount)
    }
}

```

Activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<layout>
    <activity android:name=".MainActivity" android:label="Lemonade">
        <activity_main>
            <include layout="@+id/ic_launcher_lemonade_foreground.xml" />
        </activity_main>
    </activity>
</layout>

```

MainActivity.kt

```

class MainActivity : AppCompatActivity() {
    private var lemonadeState = "select"
    // Default var lemonSize to -1
    private var lemonSize = -1
    // Default the squeezeCount to -1
    private var squeezeCount = -1

    private var lemonTree = LemonTree()
    private var lemonImage: ImageView? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // *** DO NOT ALTER THE CODE IN THE FOLLOWING IF STATEMENT ***
        if (savedInstanceState != null) {
            lemonadeState = savedInstanceState.getString(LEMONADE_STATE, defaultValue = "select")
            lemonSize = savedInstanceState.getInt(LEMON_SIZE, defaultValue = -1)
            squeezeCount = savedInstanceState.getInt(SQUEEZE_COUNT, defaultValue = -1)
        }
        // *** END IF STATEMENT ***

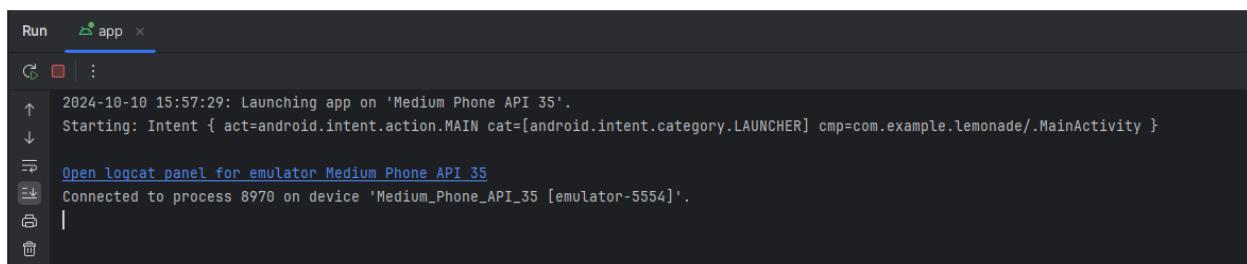
        lemonImage = findViewById(R.id.image_lemon_state)
        setViewElements()
        lemonImage!!.setOnClickListener {
            clickLemonImage() // gọi hàm xử lý click
        }
        lemonImage!!.setOnLongClickListener {
            showSnackbar() // hiển thị số lần ép chanh nếu hợp lệ
        }
    }

    /**
     * *** DO NOT ALTER THIS METHOD ***
     *
     * This method saves the state of the app if it is put in the background.
     */
    override fun onSaveInstanceState(outState: Bundle) {
        outState.putString(LEMONADE_STATE, lemonadeState)
        outState.putInt(LEMON_SIZE, lemonSize)
        outState.putInt(SQUEEZE_COUNT, squeezeCount)
    }
}

```

4.6. Testing instructions

- Hoàn thành các bước kiểm thử không bao lối:



The screenshot shows the 'Run' tab in the Android Studio interface, specifically the log window for the 'app' configuration. The window displays the following log output:

```
2024-10-10 15:57:29: Launching app on 'Medium Phone API 35'.
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.example.lemonade/.MainActivity }
Open logcat panel for emulator Medium Phone API 35
Connected to process 8970 on device 'Medium_Phone_API_35 [emulator-5554]'.
```