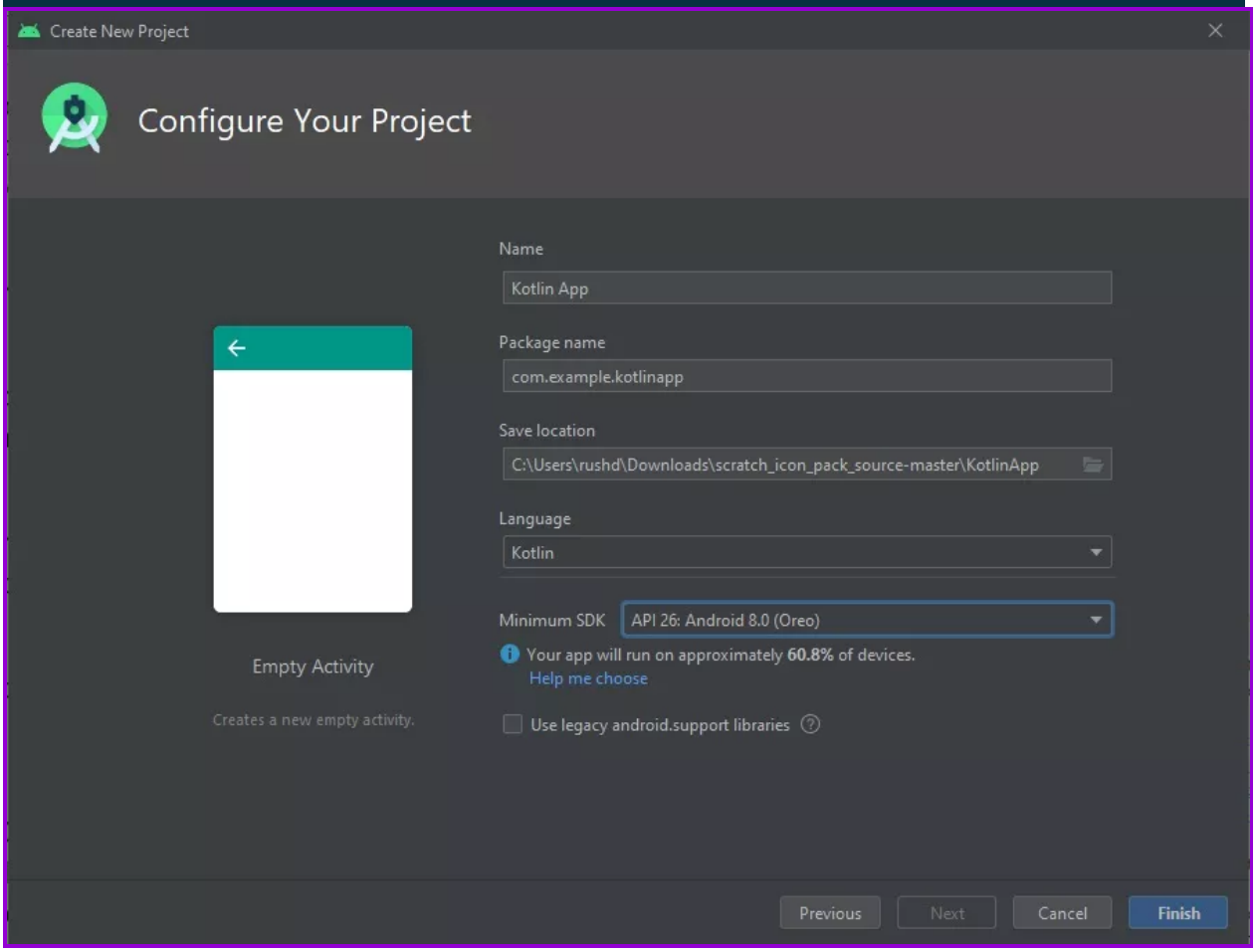
ANDROID BASICS IN KOTLIN

Unit 1: Kotlin basics

Build your first Android apps with the Kotlin programming language. Add images and text to your Android apps, and learn how to use classes, objects, and conditionals to create an interactive app for your users.



short program in Kotlin using functions and loops to print a happy birthday message.

```
fun main() {
    val age = 24
    val layers = 5
    printCakeCandles(age)
    printCakeTop(age)
    printCakeBottom(age, layers)
}
```

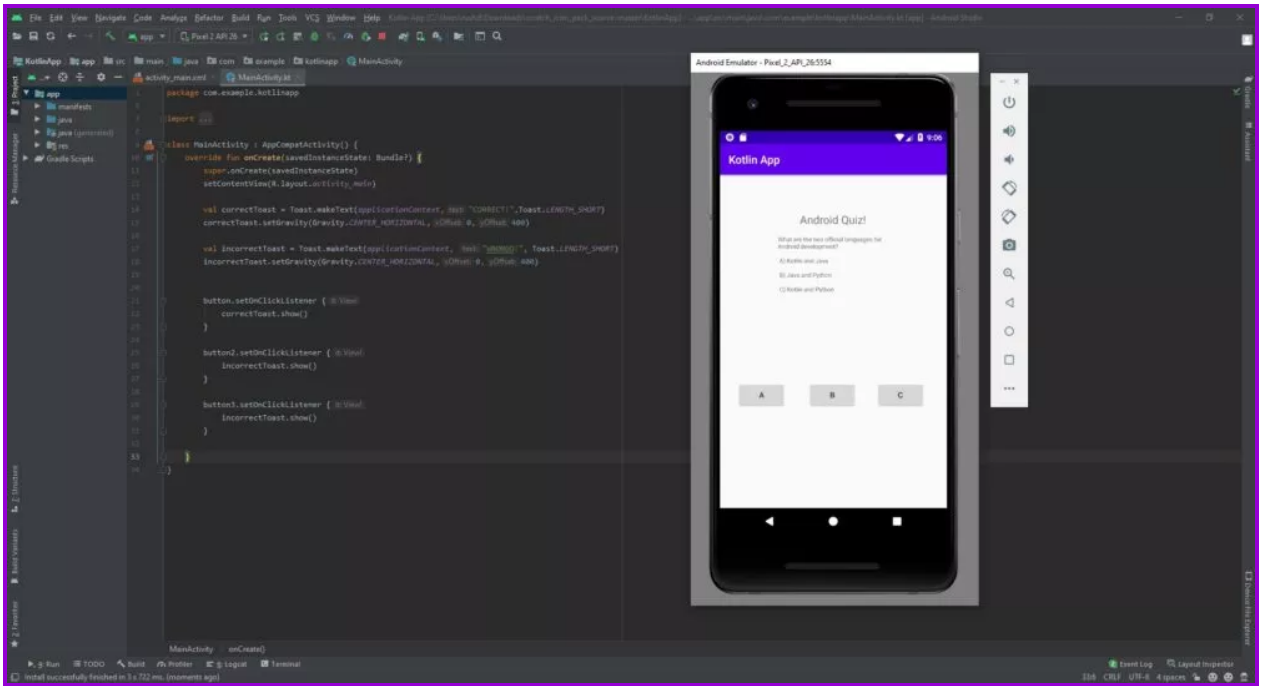
```
fun printCakeCandles(age: Int) {
    print (" ")
    repeat(age) {
         print(",")
    }
    println() // Print an empty line

    print(" ") // Print the inset of the candles on the cake
    repeat(age) {
        print("|")
    }
    println()
}

fun printCakeTop(age: Int) {
    repeat(age + 2) {
        print("=")
    }
    println()
}

fun printCakeBottom(age: Int, layers: Int) {
    repeat(layers) {
        repeat(age + 2) {
            print("@")
        }
        println()
    }
}
```

- **Explain:** Use `${}` to surround variables and calculations in the text of print statements. For example: `${age}` where `age` is a variable.
- Create a variable using the `val` keyword and a name. Once set, this value cannot be changed. Assign a value to a variable using the equal sign. Examples of values are text and numbers.
- A `String` is text surrounded by quotes, such as `"Hello"`.

- An `Int` is a whole positive or negative number, such as 0, 23, or -1024.
- You can pass one or more arguments into a function for the function to use, for example: `fun printCakeBottom(age:Int, layers:Int) {}`
- Use a `repeat() {}` statement to repeat a set of instructions several times. For example: `repeat (23) { print("%") }` or `repeat (layers) { print("@@@@@@@@@@") }`
- A *loop* is an instruction to repeat instructions multiple times. A `repeat()` statement is an example of a loop.
- You can nest loops, that is, put loops within loops. For example, you can create a `repeat()` statement within a `repeat()` statement to print a symbol a number of times for a number of rows, like you did for the cake layers.

Summary of using function arguments: To use arguments with a function, you need to do three things:

- Add the argument and type to the function definition: `printBorder(border: String)`
- Use the argument inside the function: `println(border)`
- Supply the argument when you call the function: `printBorder(border)`



Unit 2: Layouts

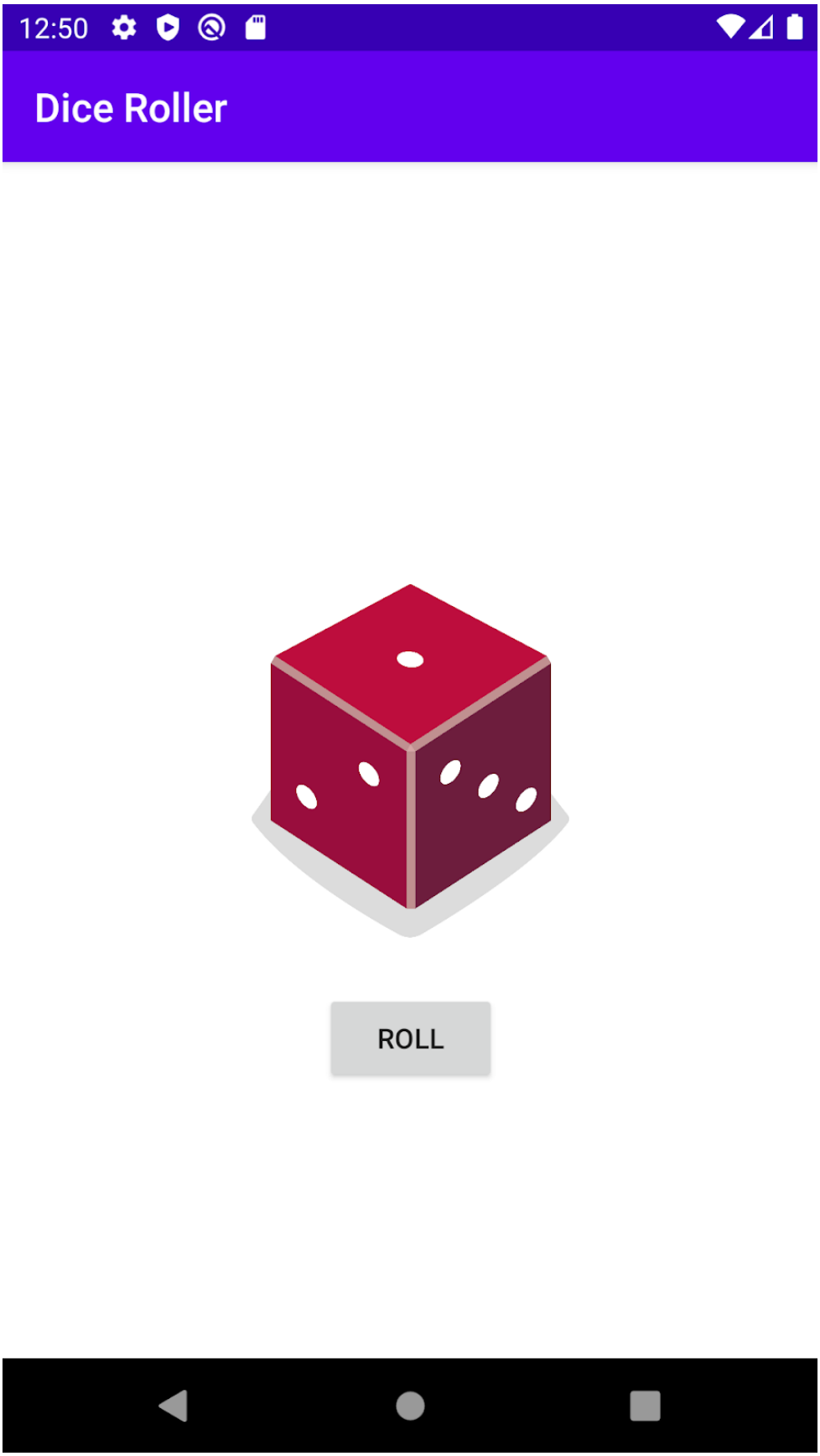
Improve the user interface of your app by learning about layouts, Material Design

guidelines, and best practices for UI development.

Add images to the Dice Roller app

```
package
com.example.diceroller

import android.os.Bundle
import android.widget.Button
import android.widget.ImageView
import androidx.appcompat.app.AppCompatActivity

/**
 * This activity allows the user to roll a dice and view the result
 * on the screen.
 */
class MainActivity : AppCompatActivity() {

    /**
     * This method is called when the Activity is created.
     */
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Find the Button in the layout
        val rollButton: Button = findViewById(R.id.button)

        // Set a click listener on the button to roll the dice when t
the button
        rollButton.setOnClickListener { rollDice() }

        // Do a dice roll when the app starts
        rollDice()
    }

    /**
     * Roll the dice and update the screen with the result.
     */
    private fun rollDice() {
        // Create new Dice object with 6 sides and roll it
        val dice = Dice(6)
        val diceRoll = dice.roll()
```

```
        // Find the ImageView in the layout
        val diceImage: ImageView = findViewById(R.id.imageView)

        // Determine which drawable resource ID to use based on the d
        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
        }

        // Update the ImageView with the correct drawable resource ID
        diceImage.setImageResource(drawableResource)

        // Update the content description
        diceImage.contentDescription = diceRoll.toString()
    }
}

/**
 * Dice with a fixed number of sides.
 */
class Dice(private val numSides: Int) {

    /**
     * Do a random dice roll and return the result.
     */
    fun roll(): Int {
        return (1..numSides).random()
    }
}
```


12:50
Dice Roller
ROLL

Project: Lemonade App - Starter Code

Starter code for the first independent project for Android Basics in Kotlin

Introduction

This is the starter code for the Lemonade app project in the final pathway of Android Basics Unit 1. This project is an opportunity for you to demonstrate the concepts you learned in the unit.

Getting Started

1. Download the starter code
2. Open the project in Android Studio
3. Complete the project in accordance with the project instructions

Tips

- Use the provided tests to ensure your app is running as expected
- DO NOT ALTER THE PROVIDED TESTS