

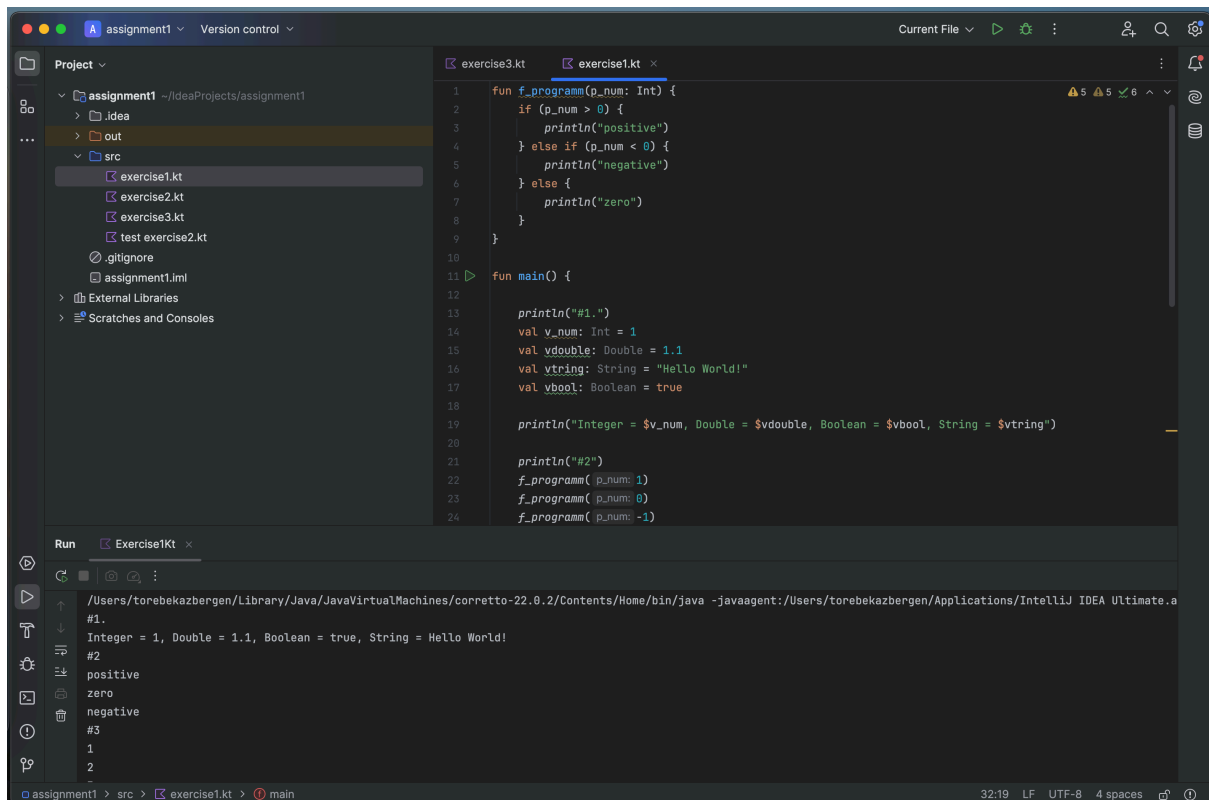
Exercise 1: Kotlin Syntax Basics

1. Variables and Data Types:

- Create variables of different data types: `Int`, `Double`, `String`, `Boolean`.
- Print the variables using `println`.

Conditional Statements:

- Create a simple program that checks if a number is positive, negative, or zero.

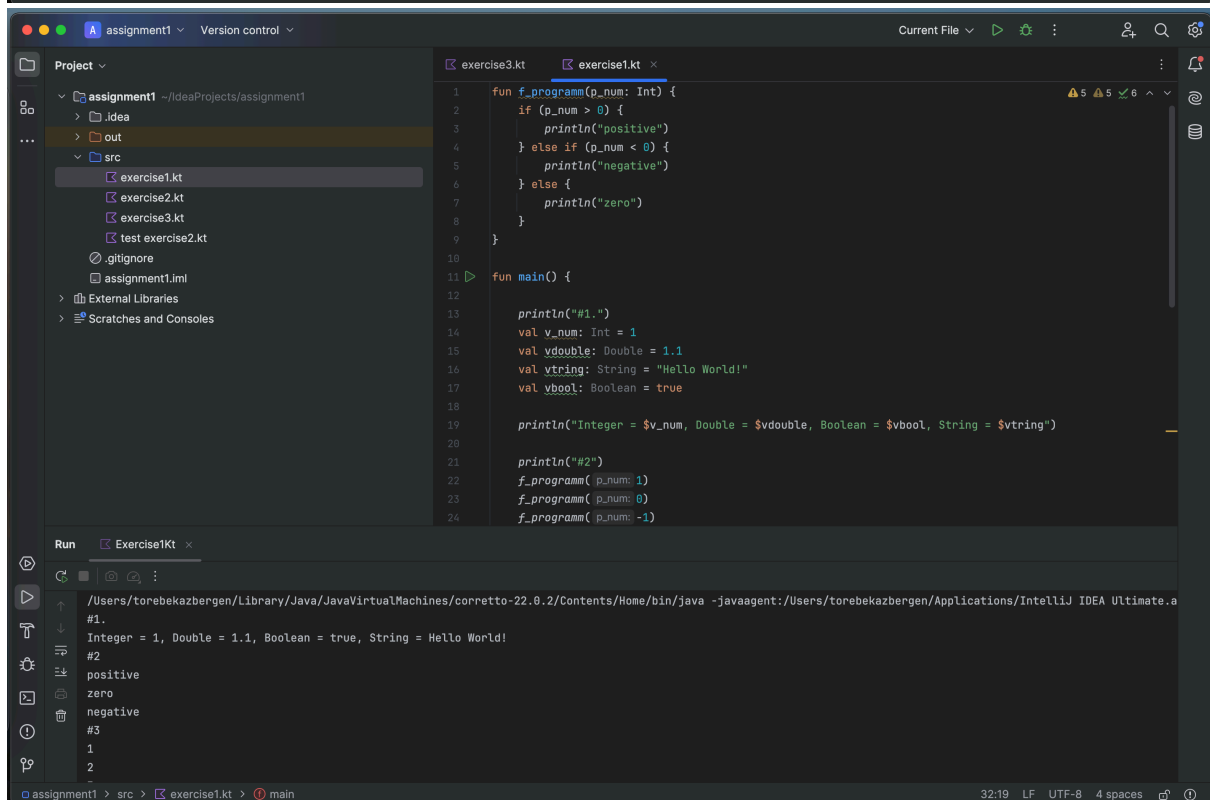
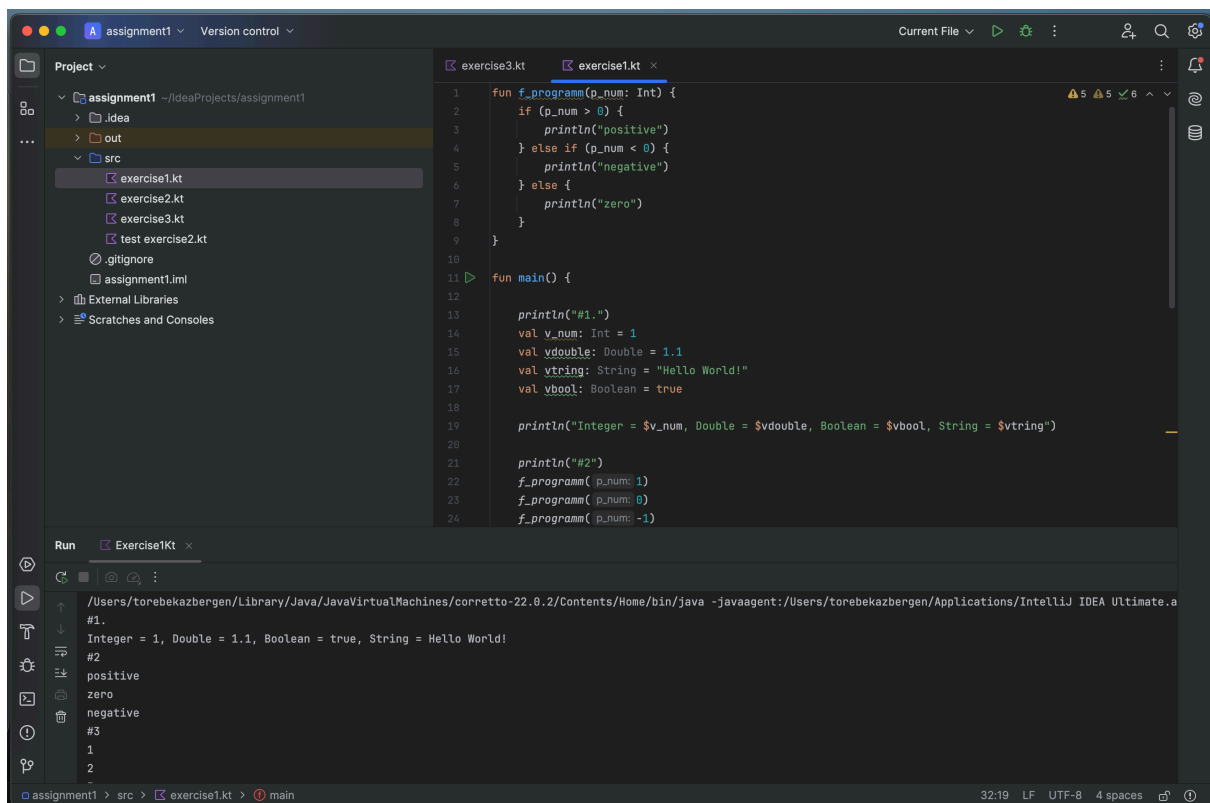


Loops:

- Write a program that prints numbers from 1 to 10 using `for` and `while` loops

Collections:

- Create a list of numbers, iterate through the list, and print the sum of all numbers.



Exercise 2: Kotlin OOP (Object-Oriented Programming)

1. Create a **Person** class:

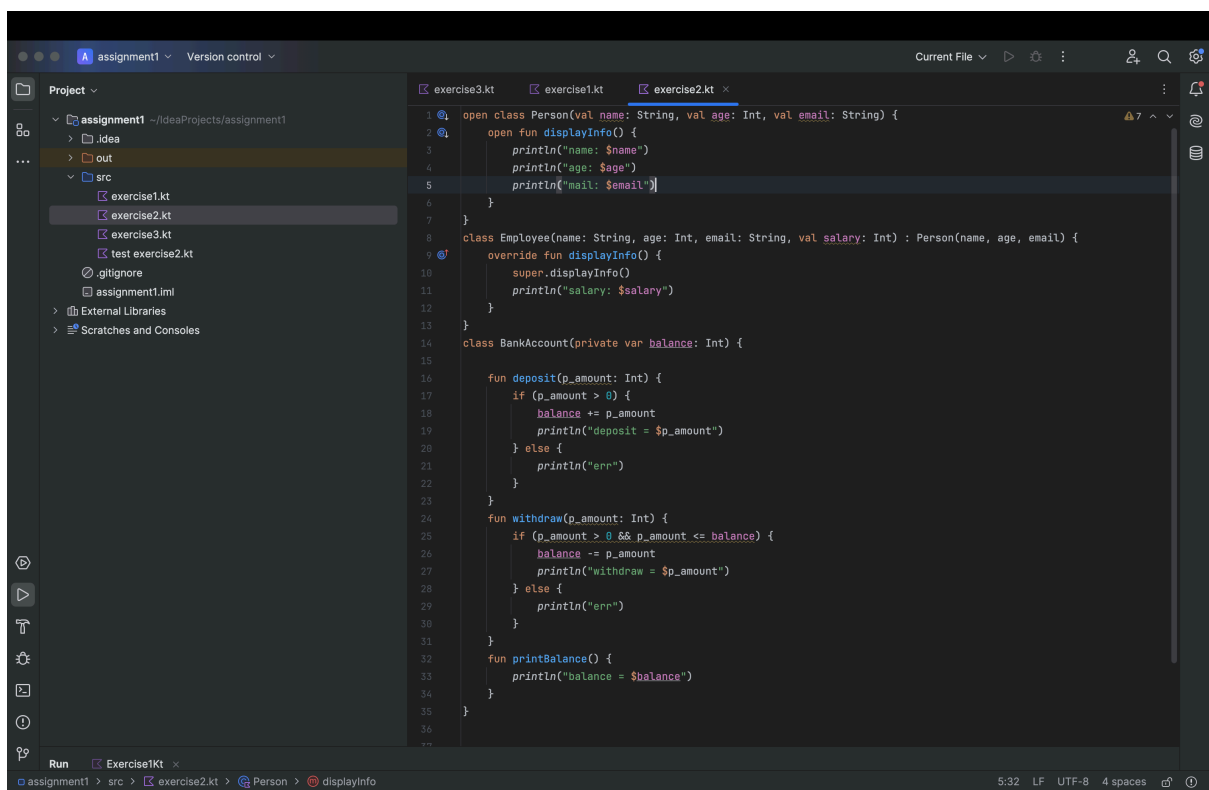
- Define properties for **name**, **age**, and **email**.
- Create a method to display the person's details.

Inheritance:

- Create a class **Employee** that inherits from the **Person** class.
- Add a property for **salary**.
- Override the **displayInfo** method to include the salary.

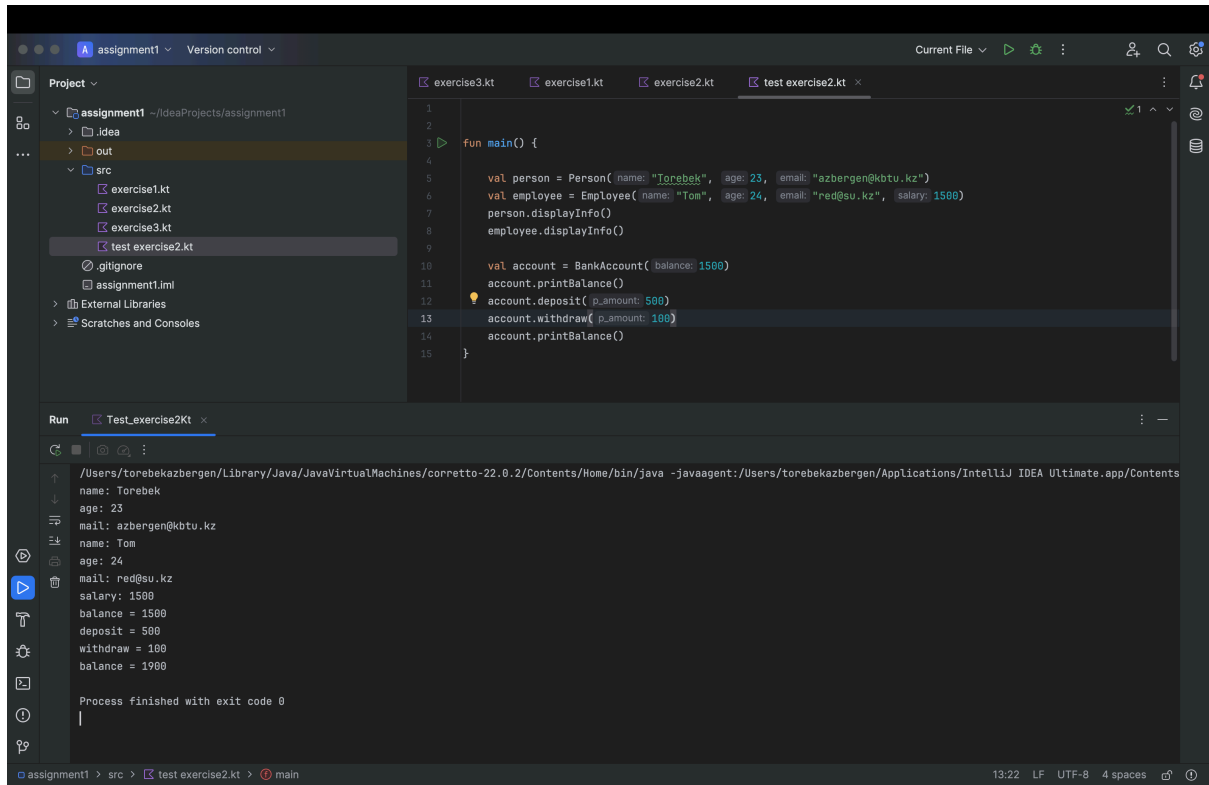
Encapsulation:

- Create a **BankAccount** class with a private property **balance**.
- Provide methods to **deposit** and **withdraw** money, ensuring the balance never goes negative.



```
1 open class Person(val name: String, val age: Int, val email: String) {
2     open fun displayInfo() {
3         println("name: $name")
4         println("age: $age")
5         println("mail: $email")
6     }
7 }
8 class Employee(name: String, age: Int, email: String, val salary: Int) : Person(name, age, email) {
9     override fun displayInfo() {
10         super.displayInfo()
11         println("salary: $salary")
12     }
13 }
14 class BankAccount(private var balance: Int) {
15
16     fun deposit(p_amount: Int) {
17         if (p_amount > 0) {
18             balance += p_amount
19             println("deposit = $p_amount")
20         } else {
21             println("err")
22         }
23     }
24
25     fun withdraw(p_amount: Int) {
26         if (p_amount > 0 && p_amount <= balance) {
27             balance -= p_amount
28             println("withdraw = $p_amount")
29         } else {
30             println("err")
31         }
32     }
33
34     fun printBalance() {
35         println("balance = $balance")
36     }
37 }
```

Test:



Exercise 3: Kotlin Functions

1. Basic Function:

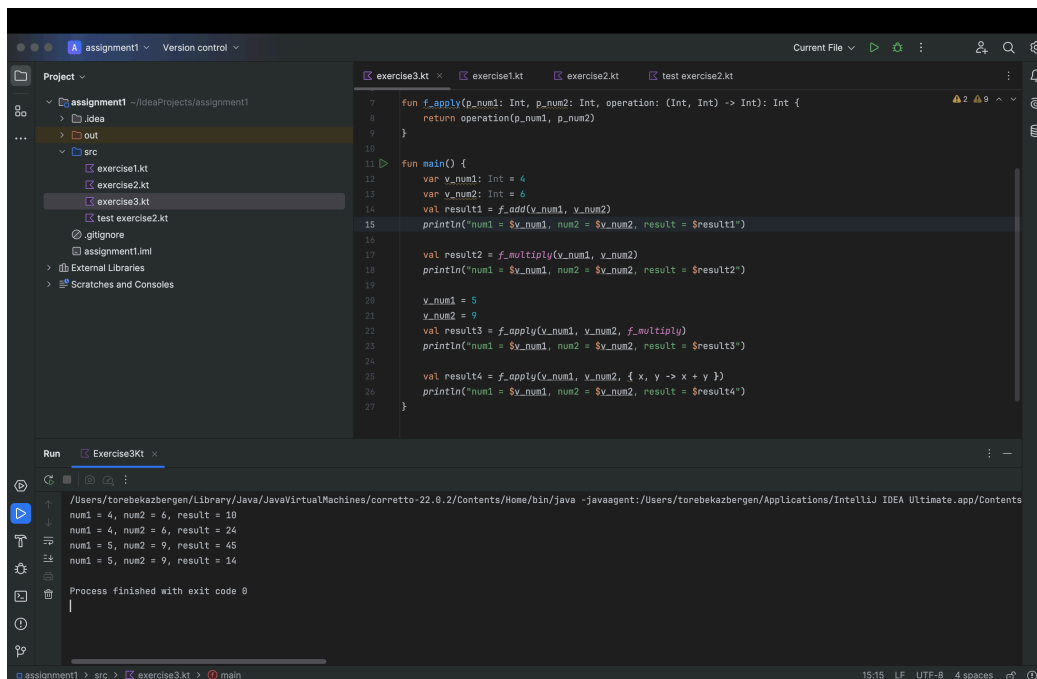
- Write a function that takes two integers as arguments and returns their sum

Lambda Functions:

- Create a lambda function that multiplies two numbers and returns the result

Higher-Order Functions:

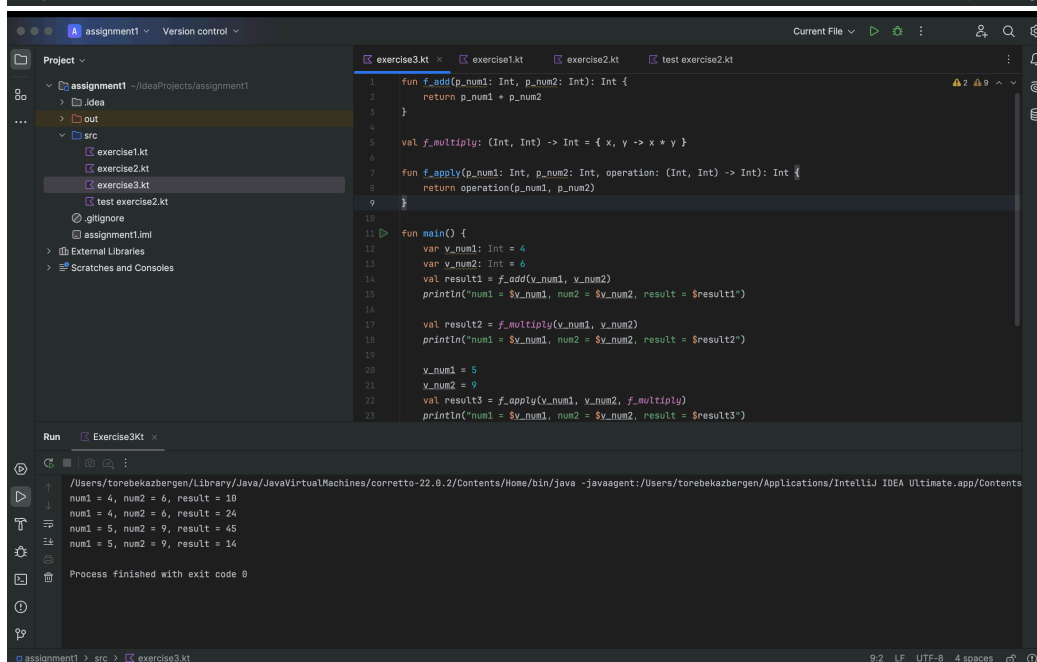
- Write a function that takes a lambda function as a parameter and applies it to two integers.



```
7 fun f_apply(p_num1: Int, p_num2: Int, operation: (Int, Int) -> Int): Int {
8     return operation(p_num1, p_num2)
9 }
10
11 fun main() {
12     var v_num1: Int = 4
13     var v_num2: Int = 6
14     val result1 = f_add(v_num1, v_num2)
15     println("num1 = $v_num1, num2 = $v_num2, result = $result1")
16
17     val result2 = f_multiply(v_num1, v_num2)
18     println("num1 = $v_num1, num2 = $v_num2, result = $result2")
19
20     v_num1 = 5
21     v_num2 = 9
22     val result3 = f_apply(v_num1, v_num2, f_multiply)
23     println("num1 = $v_num1, num2 = $v_num2, result = $result3")
24
25     val result4 = f_apply(v_num1, v_num2, { x, y -> x + y })
26     println("num1 = $v_num1, num2 = $v_num2, result = $result4")
27 }
```

Run Exercise3Kt

```
/Users/torebekazbergen/Library/Java/JavaVirtualMachines/corretto-22.0.2/Contents/Home/bin/java -javaagent:/Users/torebekazbergen/Applications/IntelliJ IDEA Ultimate.app/Contents
num1 = 4, num2 = 6, result = 10
num1 = 4, num2 = 6, result = 24
num1 = 5, num2 = 9, result = 45
num1 = 5, num2 = 9, result = 14
Process finished with exit code 0
```



```
1 fun f_add(p_num1: Int, p_num2: Int): Int {
2     return p_num1 + p_num2
3 }
4
5 val f_multiply: (Int, Int) -> Int = { x, y -> x * y }
6
7 fun f_apply(p_num1: Int, p_num2: Int, operation: (Int, Int) -> Int): Int {
8     return operation(p_num1, p_num2)
9 }
10
11 fun main() {
12     var v_num1: Int = 4
13     var v_num2: Int = 6
14     val result1 = f_add(v_num1, v_num2)
15     println("num1 = $v_num1, num2 = $v_num2, result = $result1")
16
17     val result2 = f_multiply(v_num1, v_num2)
18     println("num1 = $v_num1, num2 = $v_num2, result = $result2")
19
20     v_num1 = 5
21     v_num2 = 9
22     val result3 = f_apply(v_num1, v_num2, f_multiply)
23     println("num1 = $v_num1, num2 = $v_num2, result = $result3")
24 }
```

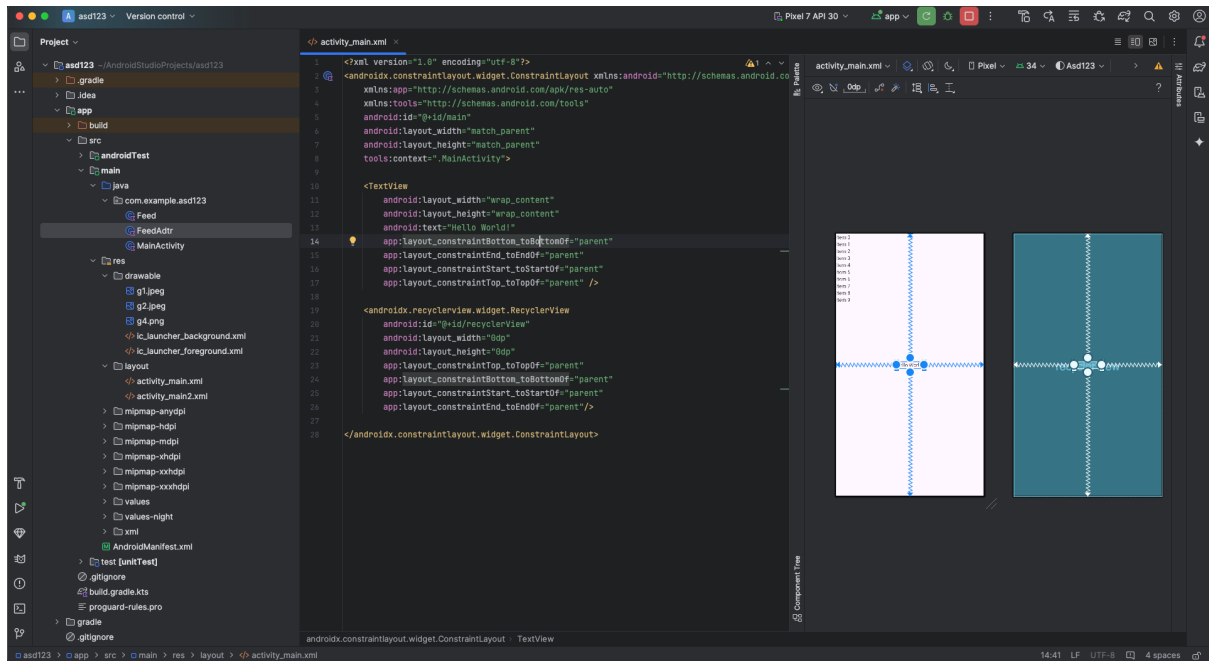
Run Exercise3Kt

```
/Users/torebekazbergen/Library/Java/JavaVirtualMachines/corretto-22.0.2/Contents/Home/bin/java -javaagent:/Users/torebekazbergen/Applications/IntelliJ IDEA Ultimate.app/Contents
num1 = 4, num2 = 6, result = 10
num1 = 4, num2 = 6, result = 24
num1 = 5, num2 = 9, result = 45
num1 = 5, num2 = 9, result = 14
Process finished with exit code 0
```

Exercise 4: Android Layout in Kotlin (Instagram-like Layout)

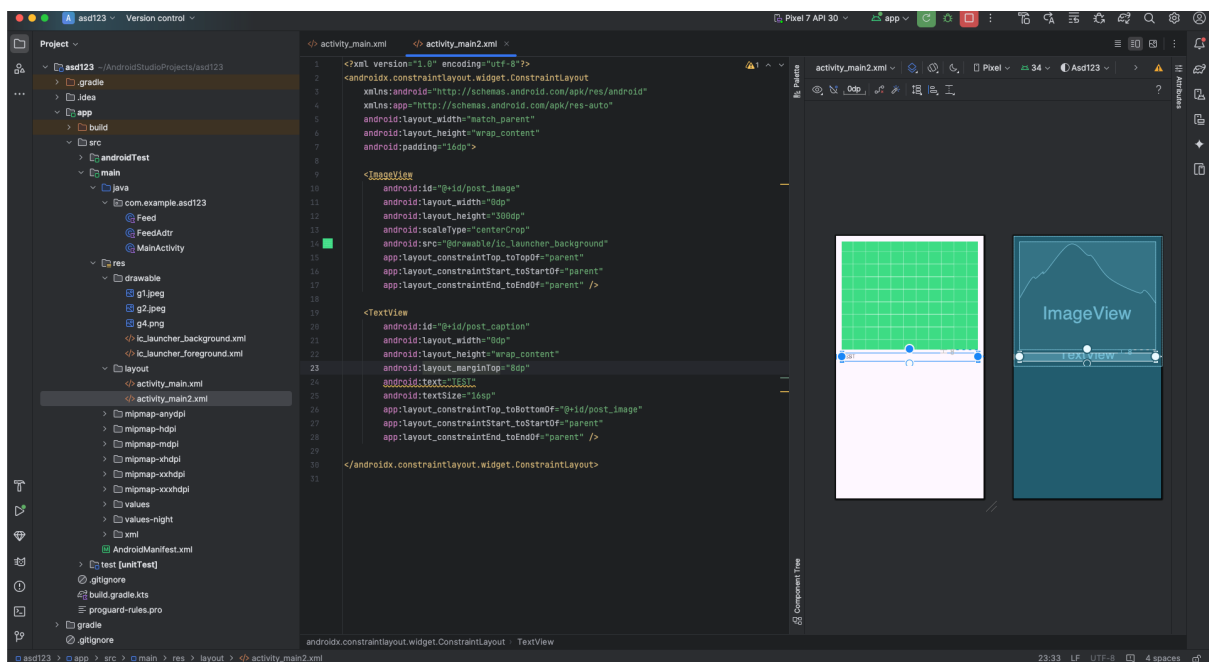
1. Set Up the Android Project:

- Create a new Android project in Android Studio.
- Ensure you have a Kotlin-based project.



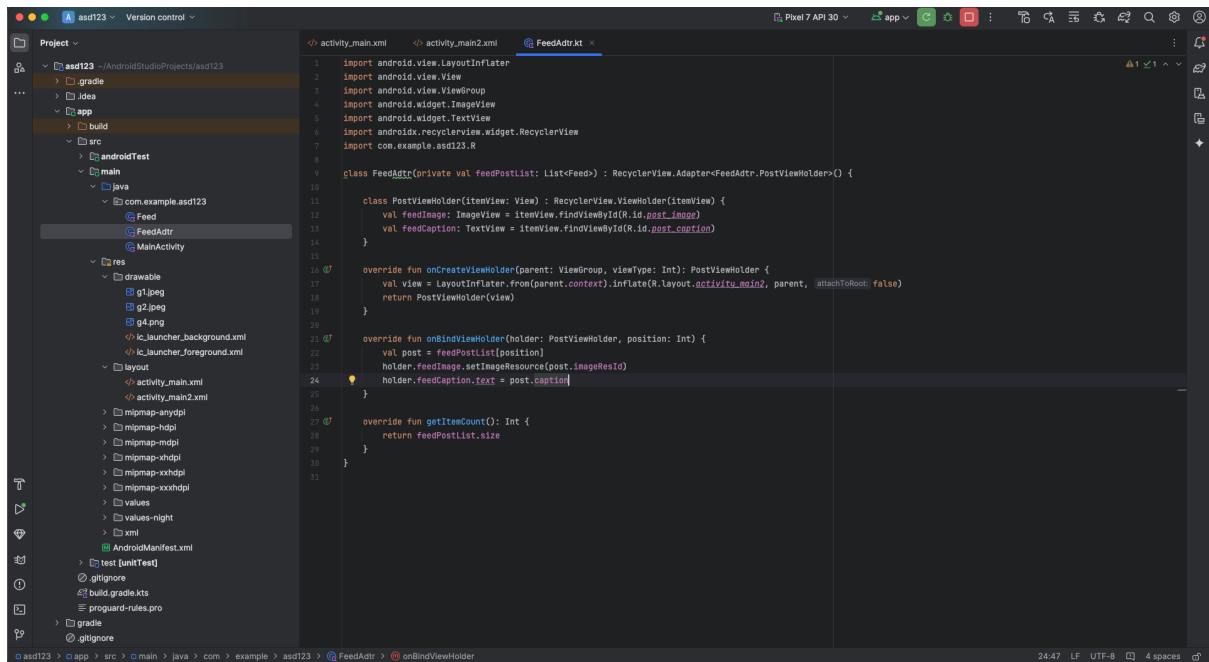
2. Design the Layout:

- Create a new XML layout file (**activity_main.xml**) for a simple Instagram-like user interface.
- Include elements like **ImageView**, **TextView**, and **RecyclerView** for the feed



Create the RecyclerView Adapter:

- Set up the RecyclerView to display a feed of posts with **ImageView** for the picture and **TextView** for the caption.



MainActivity Setup:

- Initialize the **RecyclerView** in **MainActivity** and populate it with sample data

