

Mobile Programming - Assignment 1

First and Last Name: Tamiris Abildayeva

Link to GitHub:

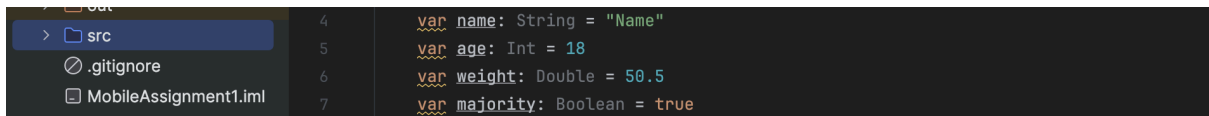
<https://github.com/TamirisK/university-mobile-programming/tree/main/%5BMobile%5D%20Assignment%201%20by%20Abildayeva%20T>

https://drive.google.com/drive/folders/1ZblzGDlGmGuCa04rBePHQNr_Ic21Gu9n?usp=share_link

Exercise 1: Kotlin Syntax Basics

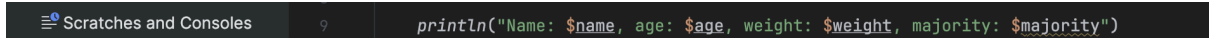
1. Variables and Data Types:

- Create variables of different data types: `Int`, `Double`, `String`, `Boolean`.



```
4  var name: String = "Name"
5  var age: Int = 18
6  var weight: Double = 50.5
7  var majority: Boolean = true
```

- Print the variables using `println`.



```
9  println("Name: $name, age: $age, weight: $weight, majority: $majority")
```

```
Project ▾
  ▾ MobileAssignment1 ~/Project
    ▾ .idea
    ▾ out
    ▾ src
    .gitignore
    MobileAssignment1.iml
  ▾ External Libraries
  ▾ Scratches and Consoles

Exercise1.kt | Example2.kt | Example3.kt
1  import java.lang.NumberFormatException
2
3  fun variableDataTypes(){
4      var name: String = "Name"
5      var age: Int = 18
6      var weight: Double = 50.5
7      var majority: Boolean = true
8
9      println("Name: $name, age: $age, weight: $weight, majority: $majority")
10 }
11
12 > fun numberChecker(){...}
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35 > fun printNumbers(){...}
36
37
38
39
40
41
42
43
44
45
46
47
48
49 > fun sumOfNumbersList(){...}
50
51
52
53
54
55
56 // without list
57 > fun sumOfNumbers(){...}
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73 ▶ fun main() {
74     println("Exercise 1")
75
76     println("\tTask 1. Variables and Data Types")
77     variableDataTypes()
78 }
```

Run Exercise1Kt x

/Users/tamirisabildayeva/Library/Java/JavaVirtualMachines/openjdk-23/Contents/Home/bin/java -javaagent:/Applications.
Exercise 1
Task 1. Variables and Data Types
Name: Name, age: 18, weight: 50.5, majority: true

Conditional Statements:

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

The screenshot shows an IDE with a project named 'MobileAssignment1'. The 'src' directory is selected in the project view. The main editor displays the code for 'Exercise1.kt', which defines a 'numberChecker()' function. This function uses a 'while (true)' loop to repeatedly prompt the user for a number. It uses 'try-catch' blocks to handle 'NumberFormatException' when the input is not a valid integer. Inside the loop, three 'if' statements check if the number is positive, zero, or negative, and print the corresponding message. A comment indicates an alternative version of these checks using 'number > 0', 'number == 0', and 'number < 0'. The loop is broken when the user enters a non-integer value, printing 'Not number entered \nGood Bye \nExiting...'. The 'Run' tab at the bottom shows the execution path and the output of the program, demonstrating the conditional logic for positive, negative, and zero values.

```
12 fun numberChecker(){
13     while (true){
14         println("Enter any number. To exit enter any word")
15
16         try {
17             var number: Int = readln().toInt()
18
19             if (number > 0) println("Number $number is positive")
20             if (number == 0) println("Number $number is zero")
21             if (number < 0) println("Number $number is negative")
22
23             // Another version of if case
24             // number > 0 -> println("Number $number is positive")
25             // number == 0 -> println("Number $number is zero")
26             // number < 0 -> println("Number $number is negative")
27
28         } catch (e: NumberFormatException){
29             println("Not number entered \nGood Bye \nExiting...")
30             break
31         }
32     }
33 }
```

Run Exercise1Kt

/Users/tamirisabildayeva/Library/Java/JavaVirtualMachines/openjdk-23/Contents/Home/bin/java -javaagent:/Applications/Exercise 1

Task 2. Conditional Statements

Enter any number. To exit enter any word

1

Number 1 is positive

Enter any number. To exit enter any word

-1

Number -1 is negative

Enter any number. To exit enter any word

0

Number 0 is zero

Loops:

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

```
MobileAssignment1.iml
> External Libraries
Scratches and Consoles

35 fun printNumbers(){
36     print("With For Loop: ")
37     for (i in 1 .. 10){
38         print("$i ")
39     }
40
41     print("\nWith While Loop: ")
42     var i: Int = 1
43     while (i <= 10){
44         print("$i ")
45         i++
46     }
47 }
48

Run Exercise1Kt x

/Users/tamirisabildayeva/Library/Java/JavaVirtualMachines/openjdk-23/Contents/Home/bin/java -javaagent:/Applications...
Exercise 1

Task 3. Loops
With For Loop: 1 2 3 4 5 6 7 8 9 10
With While Loop: 1 2 3 4 5 6 7 8 9 10
```

Collections:

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

The screenshot shows an IDE with a project named 'MobileAssignment1'. The source code in 'src' contains two functions: `sumOfNumbersList()` which uses `listOf` and `sum()` to calculate the sum of a list, and `sumOfNumbers()` which uses a `while` loop and `readln().toInt()` to calculate the sum of user input. The 'Run' window shows the execution output, including the list of numbers, the sum (15), and the interactive loop where the user enters '1' and '2', and finally 'exit' to stop the loop. The final output shows 'Calculating..' and 'Sum: 3'.

```
49 fun sumOfNumbersList(){
50     val numbers = listOf(1, 2, 3, 4, 5)
51     val sum = numbers.sum()
52     println("Numbers of the list is: $numbers")
53     println("The sum of the list is: $sum")
54 }
55
56 // without list
57 fun sumOfNumbers(){
58     var sum: Int = 0
59
60     while (true){
61         println("Enter any number. To exit enter any word")
62
63         try {
64             var number: Int = readln().toInt()
65             sum += number
66         } catch (e: NumberFormatException){
67             println("Calculating..\nSum: $sum")
68             break
69         }
70     }
71 }
```

Run Exercise1Kt x

/Users/tamirisabildayeva/Library/Java/JavaVirtualMachines/openjdk-23/Contents/Home/bin/java -javaagent:/Applications/Exercise 1

Task 4. Collections

Numbers of the list is: [1, 2, 3, 4, 5]

The sum of the list is: 15

Enter any number. To exit enter any word

1

Enter any number. To exit enter any word

2

Enter any number. To exit enter any word

exit

Calculating..

Sum: 3

Exercise 2: Kotlin OOP (Object-Oriented Programming)

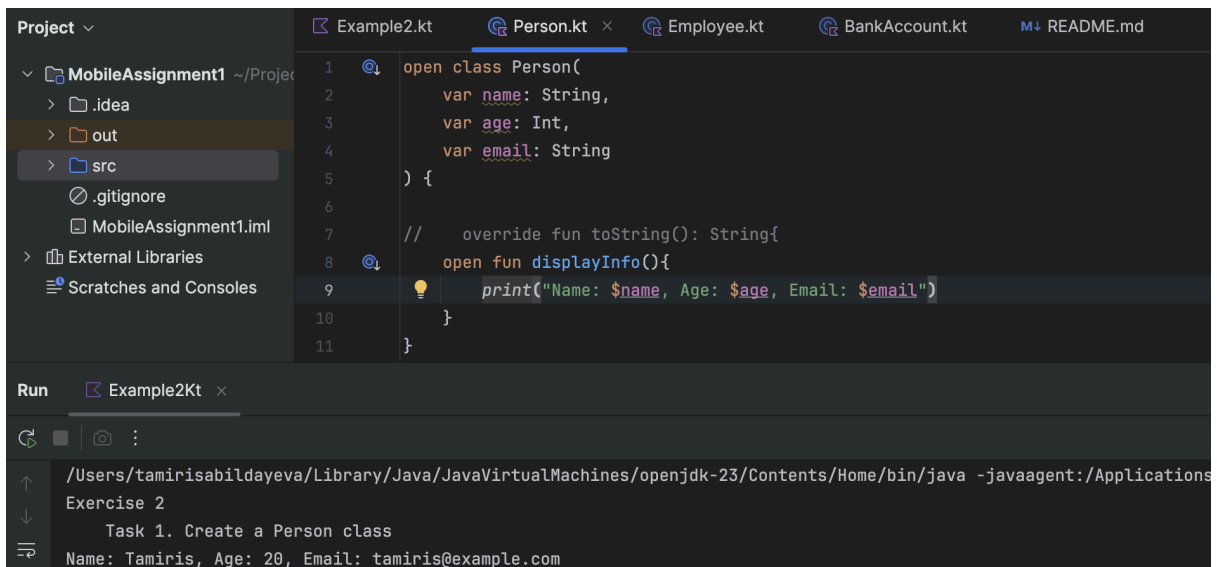
1. Create a **Person** class:

- Define properties for **name**, **age**, and **email**.

The screenshot shows an IDE with a project named 'MobileAssignment1'. The source code in 'src' shows the start of a `Person` class definition with properties `name`, `age`, and `email`.

```
1 open class Person(
2     var name: String,
3     var age: Int,
4     var email: String
5 ) {
```

- Create a method to display the person's details.



```
Project ▾
  ▾ MobileAssignment1 ~/Project
    ▾ .idea
    ▾ out
    ▾ src
    .gitignore
    MobileAssignment1.iml
  ▾ External Libraries
  ▾ Scratches and Consoles

Example2.kt  Person.kt ×  Employee.kt  BankAccount.kt  README.md

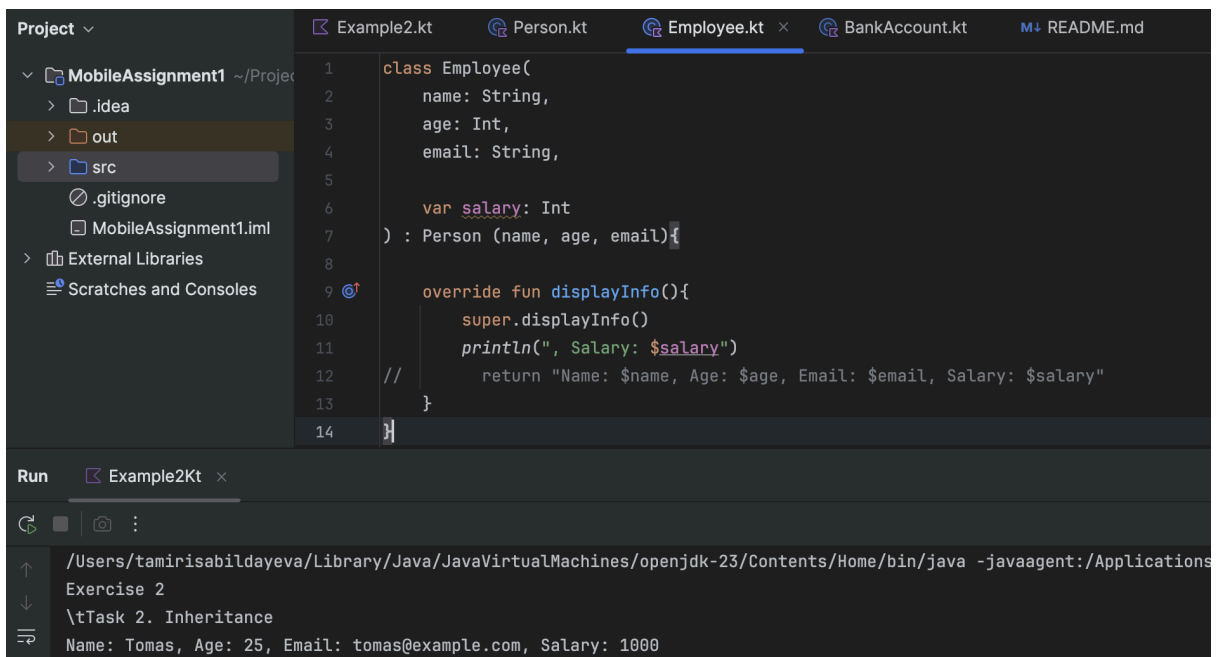
1  @  open class Person(
2      var name: String,
3      var age: Int,
4      var email: String
5  ) {
6
7      // override fun toString(): String{
8      @  open fun displayInfo(){
9          print("Name: $name, Age: $age, Email: $email")
10     }
11 }
```

```
Run  Example2Kt ×

/Users/tamirisabildayeva/Library/Java/JavaVirtualMachines/openjdk-23/Contents/Home/bin/java -javaagent:/Applications
Exercise 2
Task 1. Create a Person class
Name: Tamiris, Age: 20, Email: tamiris@example.com
```

Inheritance:

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



```
Project ▾
  ▾ MobileAssignment1 ~/Project
    ▾ .idea
    ▾ out
    ▾ src
    .gitignore
    MobileAssignment1.iml
  ▾ External Libraries
  ▾ Scratches and Consoles

Example2.kt  Person.kt  Employee.kt ×  BankAccount.kt  README.md

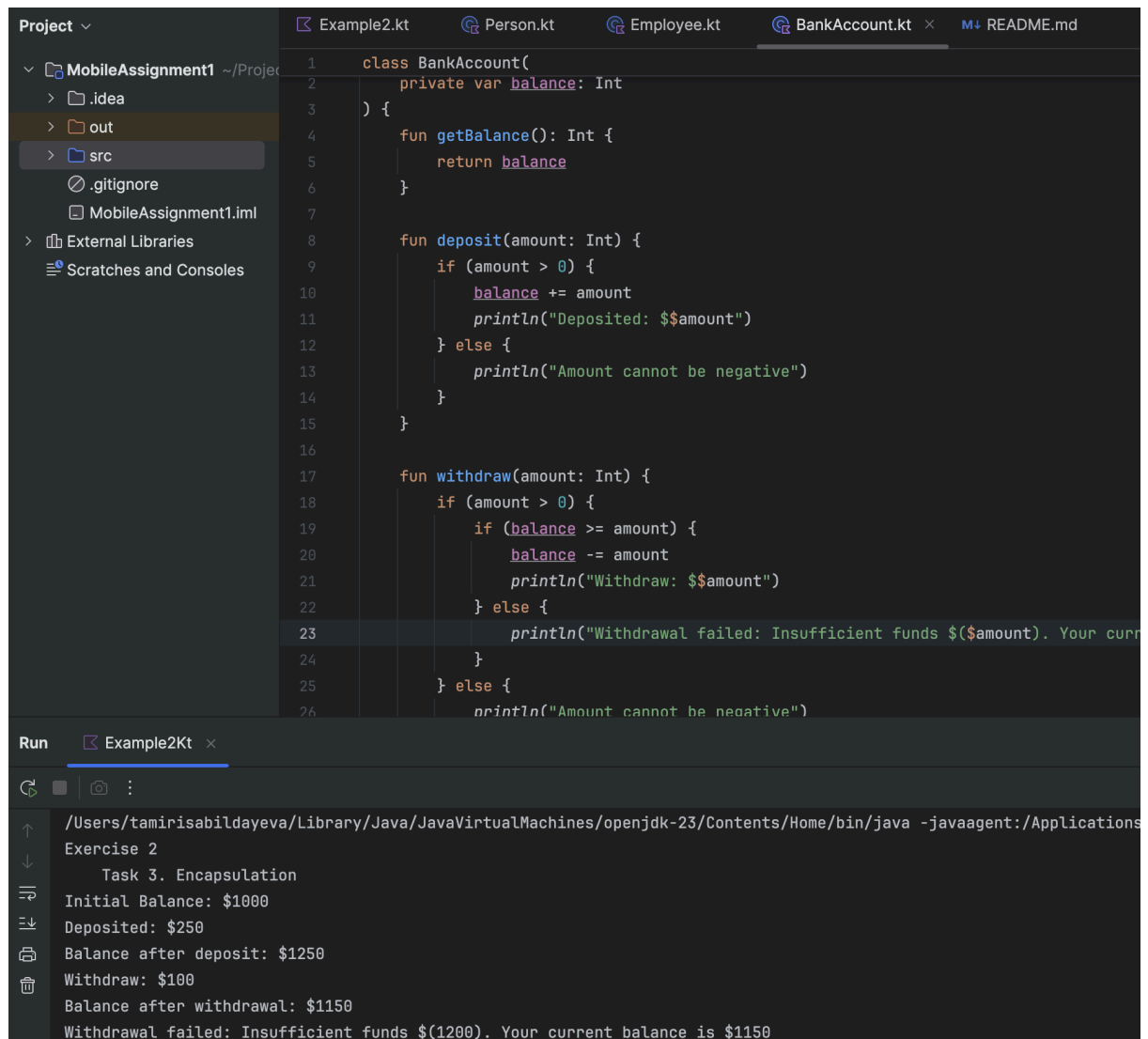
1  class Employee(
2      name: String,
3      age: Int,
4      email: String,
5
6      var salary: Int
7  ) : Person (name, age, email){
8
9      @  override fun displayInfo(){
10         super.displayInfo()
11         println(" Salary: $salary")
12         // return "Name: $name, Age: $age, Email: $email, Salary: $salary"
13     }
14 }
```

```
Run  Example2Kt ×

/Users/tamirisabildayeva/Library/Java/JavaVirtualMachines/openjdk-23/Contents/Home/bin/java -javaagent:/Applications
Exercise 2
\tTask 2. Inheritance
Name: Tomas, Age: 25, Email: tomas@example.com, Salary: 1000
```

Encapsulation:

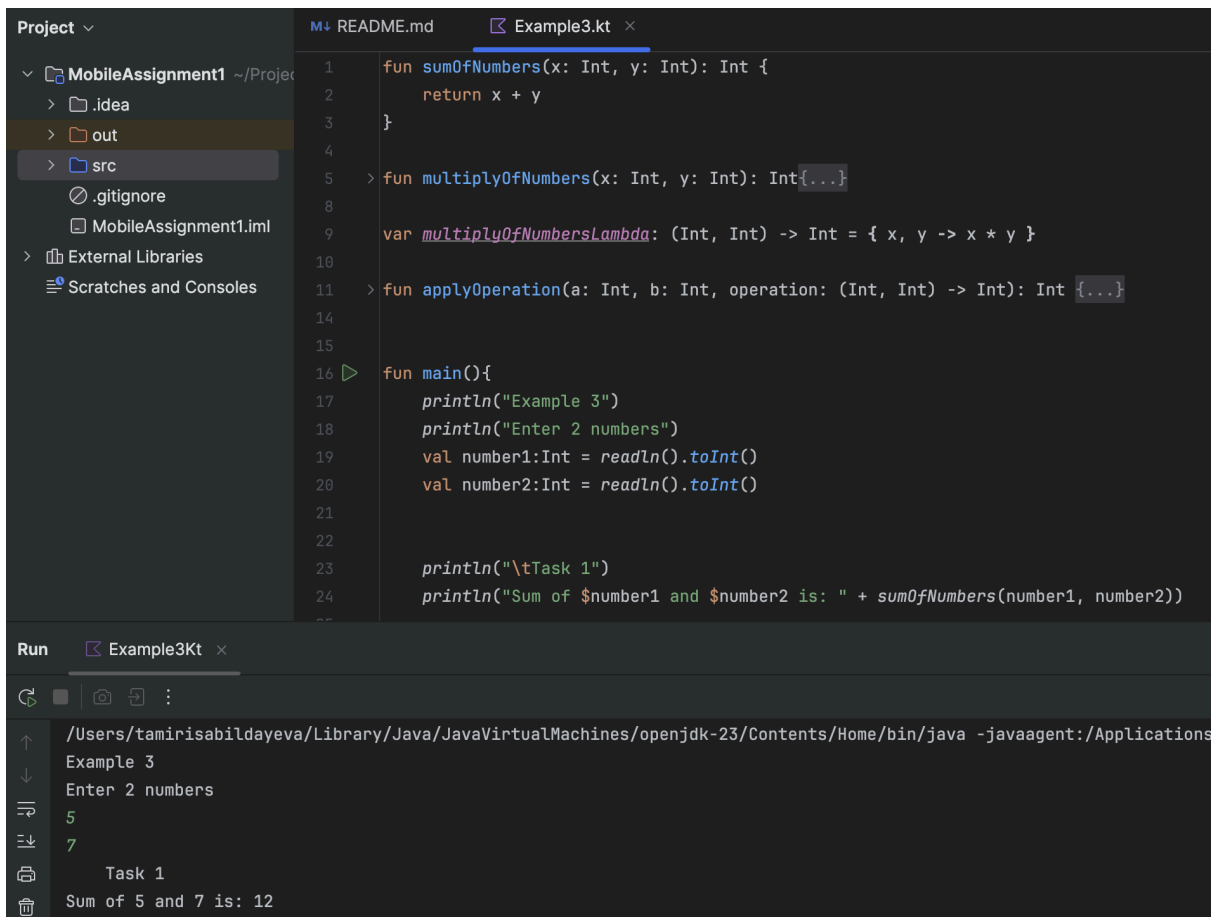
- Create a **BankAccount** class with a private property **balance**.
- Provide methods to **deposit** and **withdraw** money, ensuring the balance never goes negative.



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

The screenshot shows an IDE with a project named 'MobileAssignment1'. The file 'Example3.kt' is open, containing the following Kotlin code:

```
5 fun multiplyOfNumbers(x: Int, y: Int): Int{
6     return x * y
7 }
8
9 var multiplyOfNumbersLambda: (Int, Int) -> Int = { x, y -> x * y }
10
11 > fun applyOperation(a: Int, b: Int, operation: (Int, Int) -> Int): Int {...}
12
13
14
15
16 fun main(){
17     println("Example 3")
18     println("Enter 2 numbers")
19     val number1: Int = readln().toInt()
20     val number2: Int = readln().toInt()
21
22
23     // println("\tTask 1")
24     // println("Sum of $number1 and $number2 is: " + sumOfNumbers(number1, number2))
25
26
27     println("\n\tTask 2")
28     println("Multiply of $number1 and $number2 is: " + multiplyOfNumbers(number1, number2))
29     println("Lambda multiply of $number1 and $number2 is: " + multiplyOfNumbersLambda(number1, number2))
30 }
```

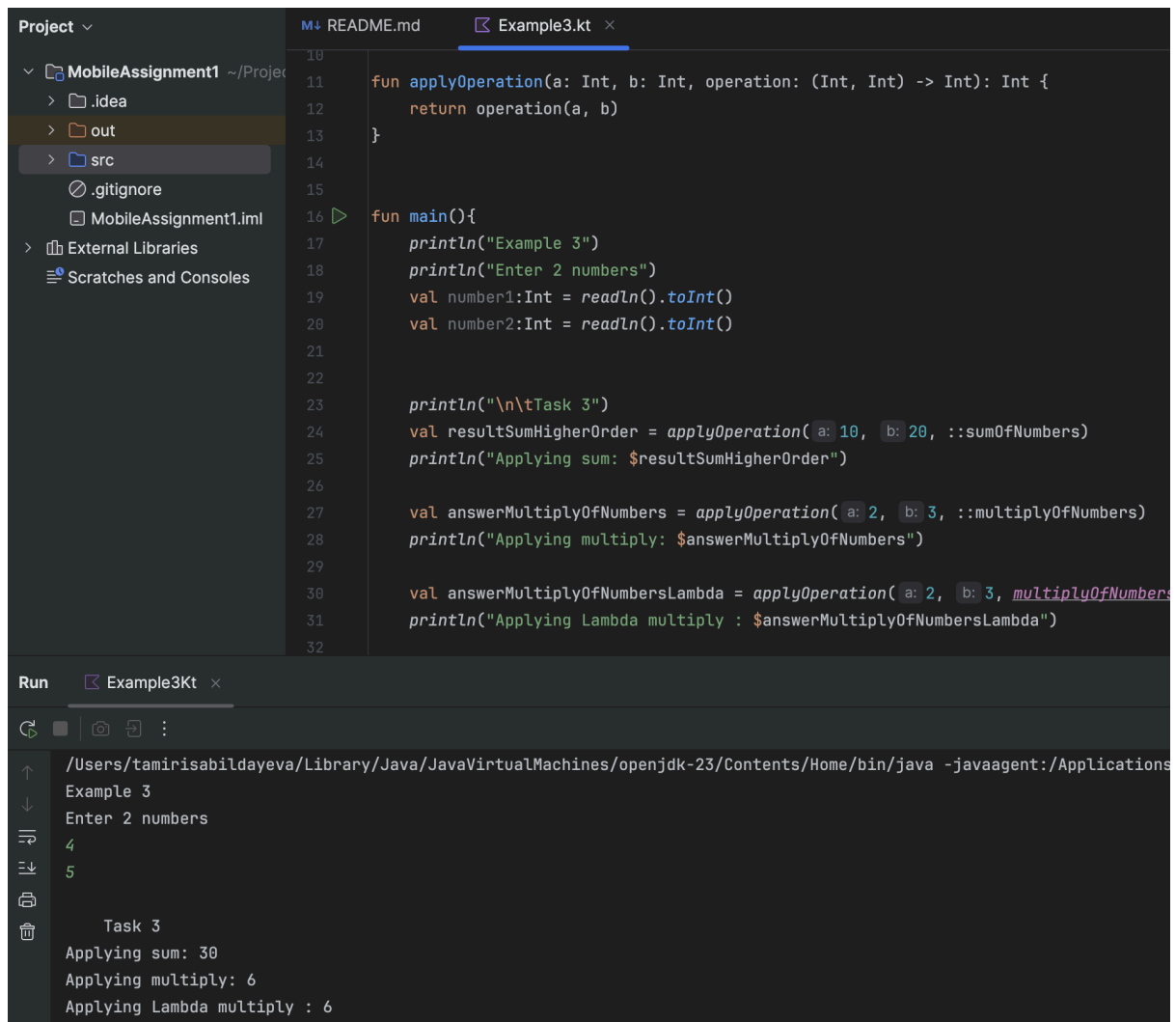
The 'Run' tab shows the output of the program:

```
/Users/tamirisabildayeva/Library/Java/JavaVirtualMachines/openjdk-23/Contents/Home/bin/java -javaagent:/Applications/Example 3
Example 3
Enter 2 numbers
3
4

Task 2
Multiply of 3 and 4 is: 12
Lambda multiply of 3 and 4 is: 12
```

Higher-Order Functions:

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



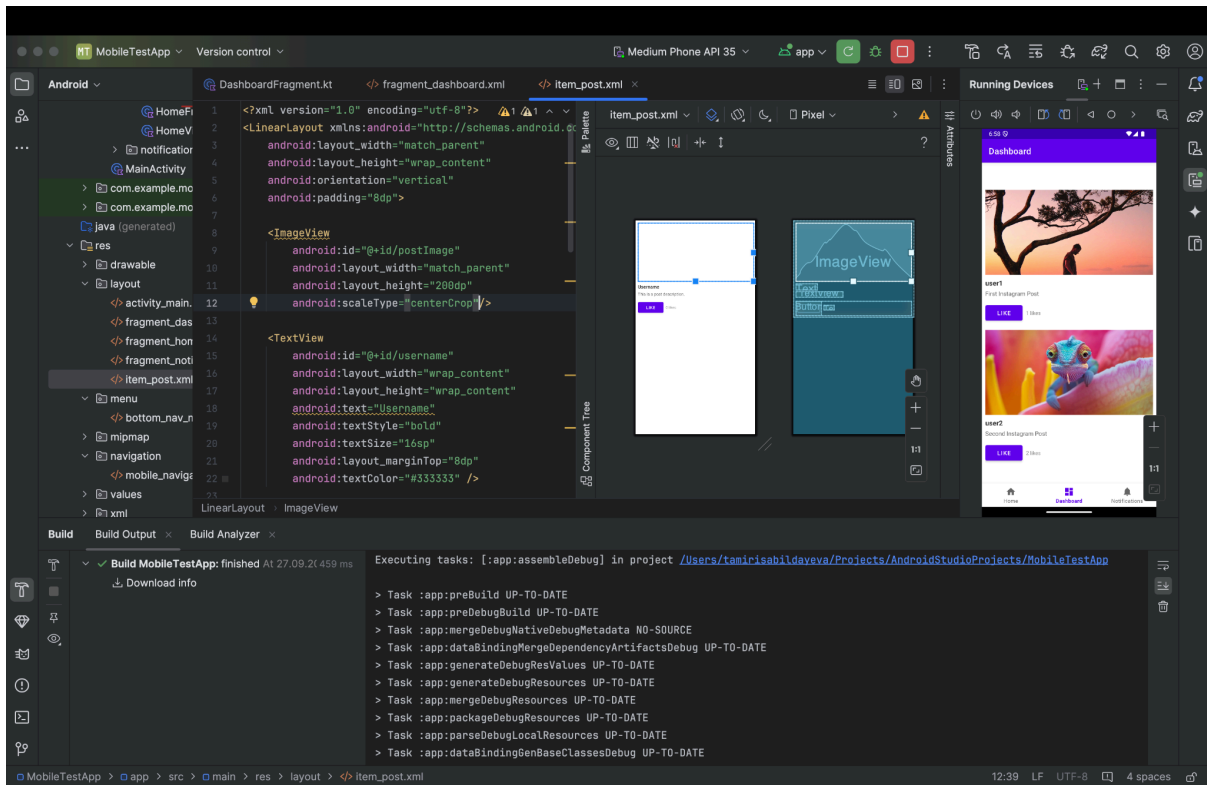
Exercise 4: Android Layout in Kotlin (Instagram-like Layout)

1. Set Up the Android Project:

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

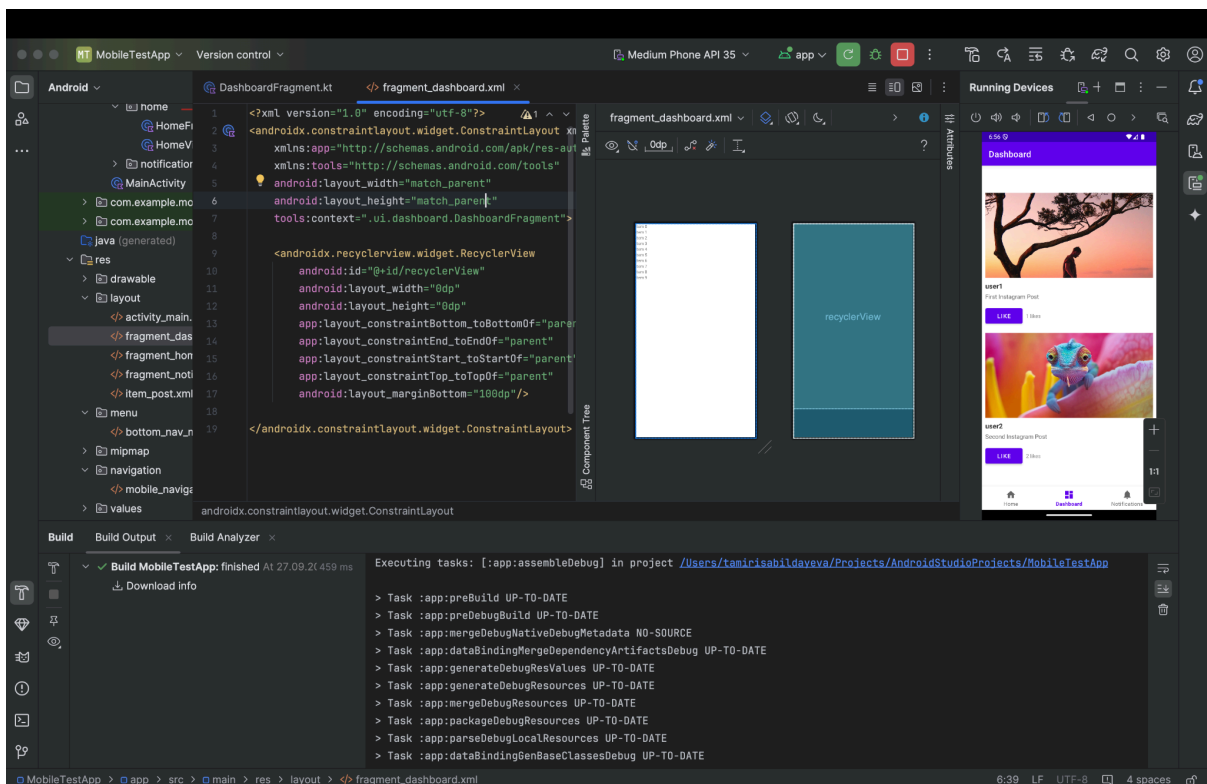
2. Design the Layout:

- 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

