

# Assignment 1

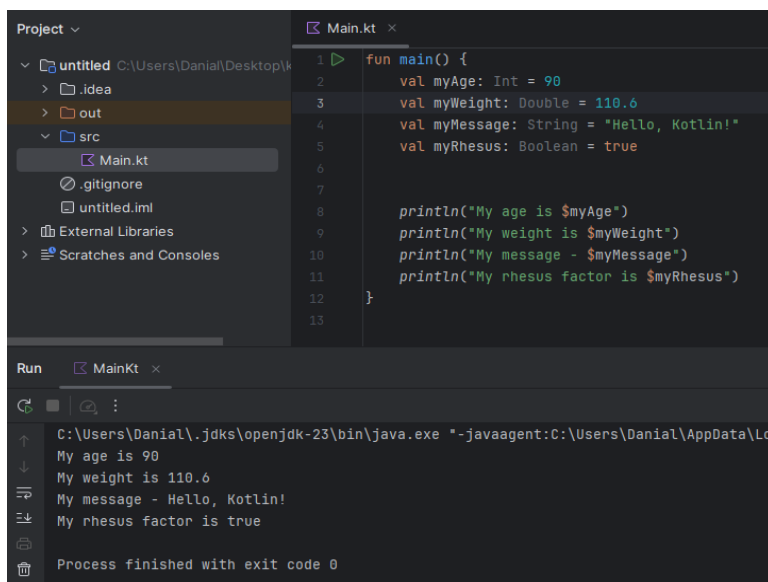
## Mobile Programming

Student: Danial Serekov

### 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**.



The screenshot shows an IDE with a project named 'untitled'. The 'src' directory contains a file 'Main.kt'. The code in 'Main.kt' is as follows:

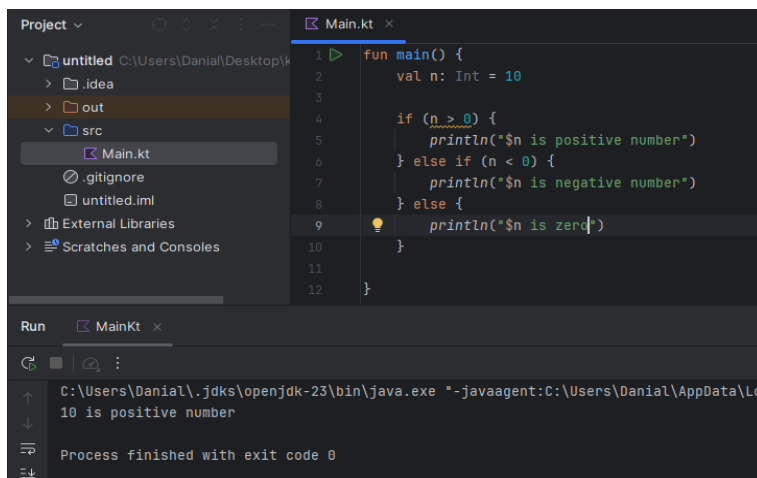
```
1 fun main() {  
2     val myAge: Int = 90  
3     val myWeight: Double = 110.6  
4     val myMessage: String = "Hello, Kotlin!"  
5     val myRhesus: Boolean = true  
6  
7  
8     println("My age is $myAge")  
9     println("My weight is $myWeight")  
10    println("My message - $myMessage")  
11    println("My rhesus factor is $myRhesus")  
12 }  
13
```

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

```
C:\Users\Danial\.jdk\openjdk-23\bin\java.exe --javaagent:C:\Users\Danial\AppData\Local\...  
My age is 90  
My weight is 110.6  
My message - Hello, Kotlin!  
My rhesus factor is true  
Process finished with exit code 0
```

#### 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 'untitled'. The 'src' directory contains a file 'Main.kt'. The code in 'Main.kt' is as follows:

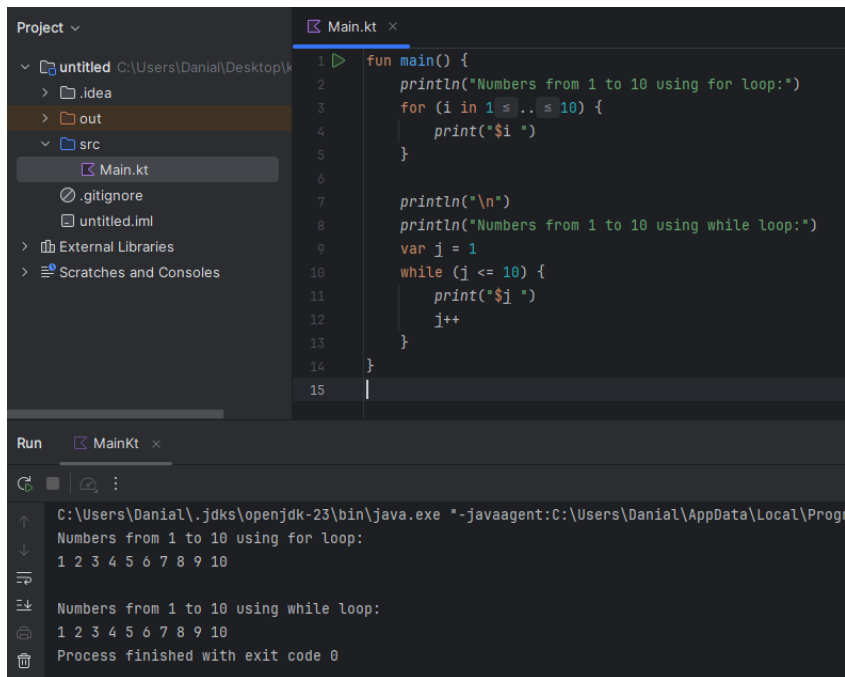
```
1 fun main() {  
2     val n: Int = 10  
3  
4     if (n > 0) {  
5         println("$n is positive number")  
6     } else if (n < 0) {  
7         println("$n is negative number")  
8     } else {  
9         println("$n is zero")  
10    }  
11  
12 }
```

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

```
C:\Users\Danial\.jdk\openjdk-23\bin\java.exe --javaagent:C:\Users\Danial\AppData\Local\...  
10 is positive number  
Process finished with exit code 0
```

## Loops:

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



```
Project ▾
└─ untitled C:\Users\Danial\Desktop\k...
   └─ .idea
      └─ out
         └─ src
            └─ Main.kt
               .gitignore
               untitled.iml
   External Libraries
   Scratches and Consoles

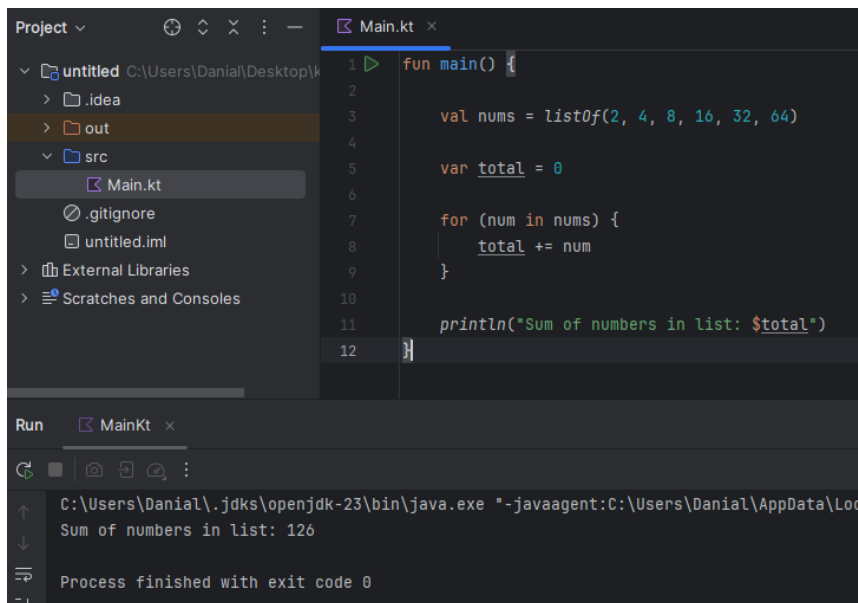
Main.kt x
1 fun main() {
2     println("Numbers from 1 to 10 using for loop:")
3     for (i in 1..10) {
4         print("$i ")
5     }
6
7     println("\n")
8     println("Numbers from 1 to 10 using while loop:")
9     var j = 1
10    while (j <= 10) {
11        print("$j ")
12        j++
13    }
14 }
15

Run MainKt x
C:\Users\Danial\.jdk\openjdk-23\bin\java.exe "-javaagent:C:\Users\Danial\AppData\Local\Progr...
Numbers from 1 to 10 using for loop:
1 2 3 4 5 6 7 8 9 10

Numbers from 1 to 10 using while loop:
1 2 3 4 5 6 7 8 9 10
Process finished with exit code 0
```

## Collections:

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



```
Project ▾
└─ untitled C:\Users\Danial\Desktop\k...
   └─ .idea
      └─ out
         └─ src
            └─ Main.kt
               .gitignore
               untitled.iml
   External Libraries
   Scratches and Consoles

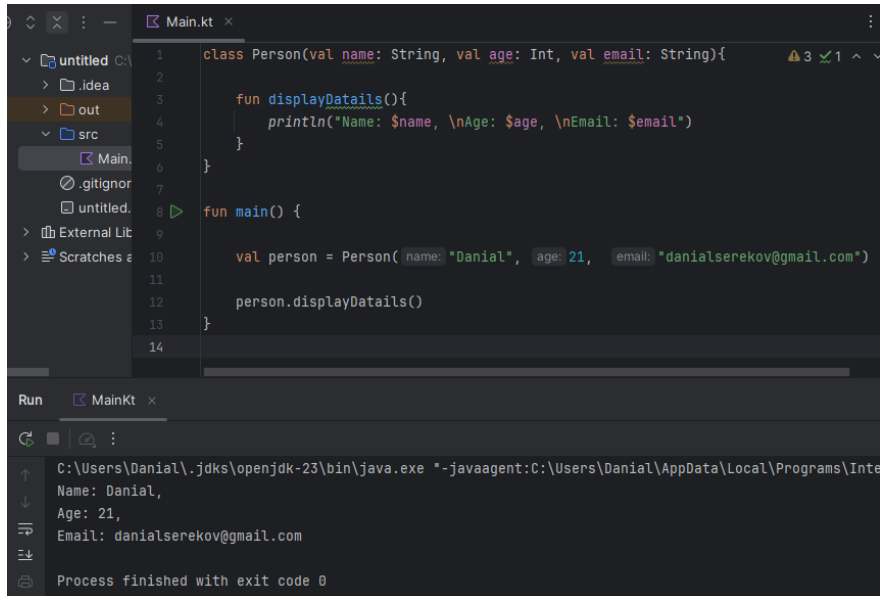
Main.kt x
1 fun main() {
2
3     val nums = listOf(2, 4, 8, 16, 32, 64)
4
5     var total = 0
6
7     for (num in nums) {
8         total += num
9     }
10
11    println("Sum of numbers in list: $total")
12 }

Run MainKt x
C:\Users\Danial\.jdk\openjdk-23\bin\java.exe "-javaagent:C:\Users\Danial\AppData\Loc...
Sum of numbers in list: 126
Process finished with exit code 0
```

## 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.



```
1 class Person(val name: String, val age: Int, val email: String){
2
3     fun displayDetails(){
4         println("Name: $name, \nAge: $age, \nEmail: $email")
5     }
6 }
7
8 fun main() {
9
10    val person = Person(name: "Danial", age: 21, email: "danialkerekov@gmail.com")
11
12    person.displayDetails()
13 }
14
```

Run MainKt x

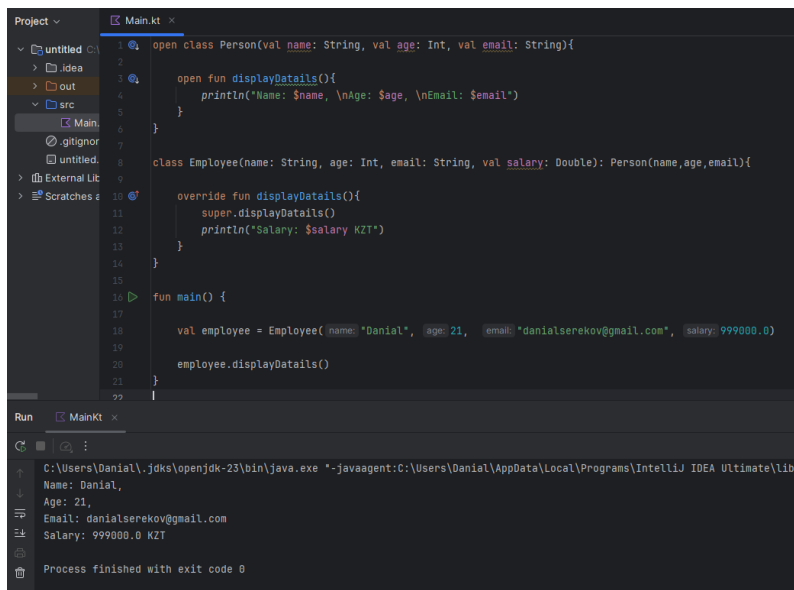
C:\Users\Danial\.jdk\openjdk-23\bin\java.exe "-javaagent:C:\Users\Danial\AppData\Local\Programs\IntelliJ IDEA Ultimate\lib\idea\_rt.jar=12137:C:\Users\Danial\AppData\Local\Programs\IntelliJ IDEA Ultimate\bin" C:\Users\Danial\IdeaProjects\untitled\src\MainKt.kt

Name: Danial,  
Age: 21,  
Email: danialkerekov@gmail.com

Process finished with exit code 0

### Inheritance:

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



```
1 open class Person(val name: String, val age: Int, val email: String){
2
3     open fun displayDetails(){
4         println("Name: $name, \nAge: $age, \nEmail: $email")
5     }
6 }
7
8 class Employee(name: String, age: Int, email: String, val salary: Double): Person(name,age,email){
9
10    override fun displayDetails(){
11        super.displayDetails()
12        println("Salary: $salary KZT")
13    }
14 }
15
16 fun main() {
17
18    val employee = Employee(name: "Danial", age: 21, email: "danialkerekov@gmail.com", salary: 999000.0)
19
20    employee.displayDetails()
21 }
22
```

Run MainKt x

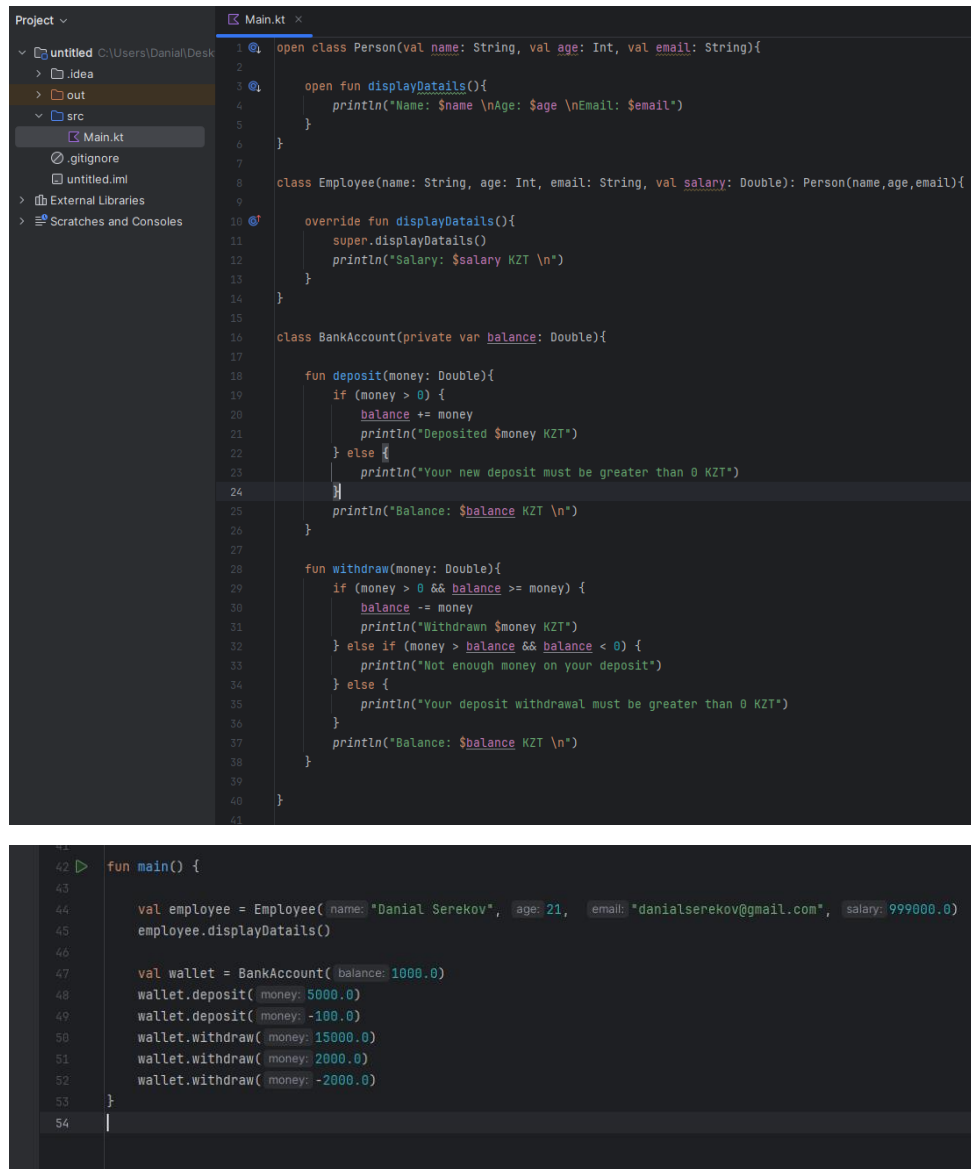
C:\Users\Danial\.jdk\openjdk-23\bin\java.exe "-javaagent:C:\Users\Danial\AppData\Local\Programs\IntelliJ IDEA Ultimate\lib\idea\_rt.jar=12137:C:\Users\Danial\AppData\Local\Programs\IntelliJ IDEA Ultimate\bin" C:\Users\Danial\IdeaProjects\untitled\src\MainKt.kt

Name: Danial,  
Age: 21,  
Email: danialkerekov@gmail.com  
Salary: 999000.0 KZT

Process finished with exit code 0

## 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
3     open fun displayDetails(){
4         println("Name: $name \nAge: $age \nEmail: $email")
5     }
6 }
7
8 class Employee(name: String, age: Int, email: String, val salary: Double): Person(name,age,email){
9
10    override fun displayDetails(){
11        super.displayDetails()
12        println("Salary: $salary KZT \n")
13    }
14 }
15
16 class BankAccount(private var balance: Double){
17
18    fun deposit(money: Double){
19        if (money > 0) {
20            balance += money
21            println("Deposited $money KZT")
22        } else {
23            println("Your new deposit must be greater than 0 KZT")
24        }
25        println("Balance: $balance KZT \n")
26    }
27
28    fun withdraw(money: Double){
29        if (money > 0 && balance >= money) {
30            balance -= money
31            println("Withdrawn $money KZT")
32        } else if (money > balance && balance < 0) {
33            println("Not enough money on your deposit")
34        } else {
35            println("Your deposit withdrawal must be greater than 0 KZT")
36        }
37        println("Balance: $balance KZT \n")
38    }
39 }
40
41
42 fun main() {
43
44     val employee = Employee( name: "Danial Serekov", age: 21, email: "danialserekov@gmail.com", salary: 999000.0)
45     employee.displayDetails()
46
47     val wallet = BankAccount( balance: 1000.0)
48     wallet.deposit( money: 5000.0)
49     wallet.deposit( money: -100.0)
50     wallet.withdraw( money: 15000.0)
51     wallet.withdraw( money: 2000.0)
52     wallet.withdraw( money: -2000.0)
53 }
54
```

**Inheritance:** class **Employee** inherits class **Person** on line 8 under “: Person(...)”.

**Polymorphism:** method **displayDetails** is overridden in class **Employee** with **override**.

**Encapsulation:** balance property in **BankAccount** is private, accessed via methods.

**Abstraction:** Details of implementation methods such as **deposit** and **withdraw** are hidden.

```
Run MainKt x
C:\Users\Danial\.jdk\openjdk-23\bin\java.exe "-javaagent:C:\Users\Danial\AppData\Local\Temp\1\jbr-23\bin\javaagent.jar"
Name: Danial Serekov
Age: 21
Email: danialserekov@gmail.com
Salary: 999000.0 KZT
Deposited 5000.0 KZT
Balance: 6000.0 KZT
Your new deposit must be greater than 0 KZT
Balance: 6000.0 KZT
Your deposit withdrawal must be greater than 0 KZT
Balance: 6000.0 KZT
Withdrawn 2000.0 KZT
Balance: 4000.0 KZT
Your deposit withdrawal must be greater than 0 KZT
Balance: 4000.0 KZT
Process finished with exit code 0
```

## Exercise 3: Kotlin Functions

### 1. Basic Function:

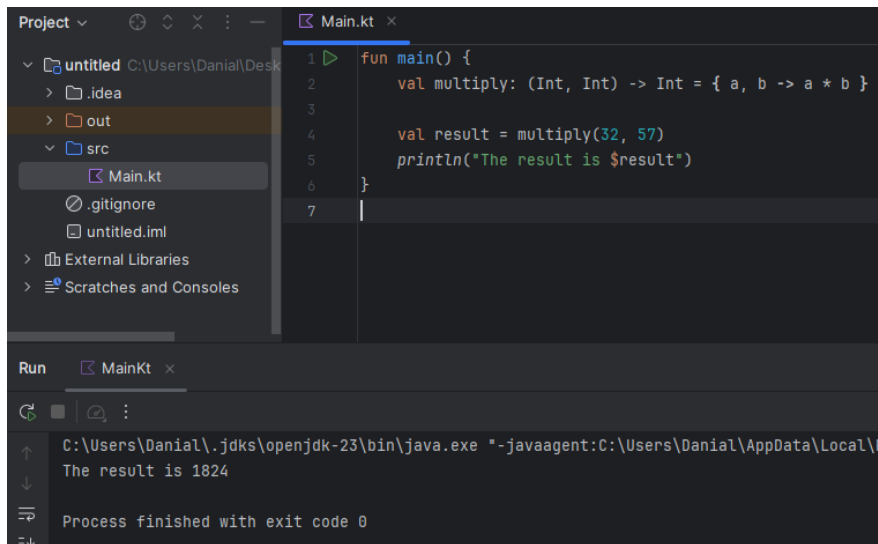
- Write a function that takes two integers as arguments and returns their sum.

```
Project Main.kt x
1 fun total(a: Int, b: Int): Int {
2     return a + b
3 }
4
5 fun main() {
6     val result = total(a: 829, b: 653)
7     println("The result is $result")
8 }
9

Run MainKt x
C:\Users\Danial\.jdk\openjdk-23\bin\java.exe "-javaagent:C:\Users\Danial\AppData\Local\Temp\1\jbr-23\bin\javaagent.jar"
The result is 1482
Process finished with exit code 0
```

## Lambda Functions:

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



```
Project ▾
├── untitled C:\Users\Danial\Desktop
│   ├── .idea
│   ├── out
│   └── src
│       └── Main.kt
├── .gitignore
├── untitled.iml
├── External Libraries
└── Scratches and Consoles

Run MainKt x

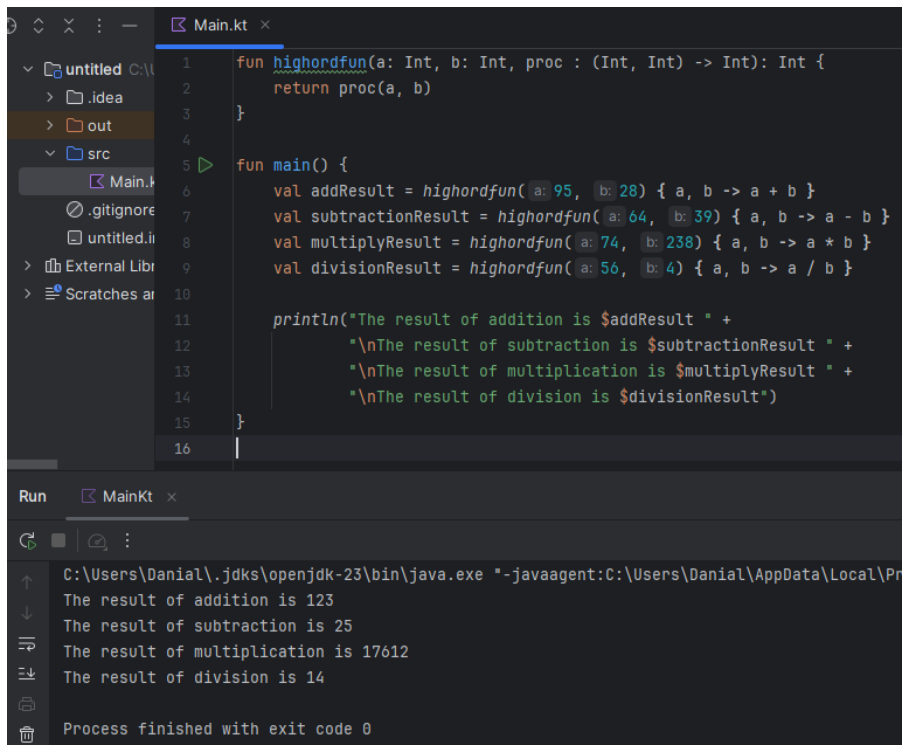
C:\Users\Danial\.jdk\openjdk-23\bin\java.exe "-javaagent:C:\Users\Danial\AppData\Local\Pr
The result is 1824

Process finished with exit code 0
```

```
1 fun main() {
2     val multiply: (Int, Int) -> Int = { a, b -> a * b }
3
4     val result = multiply(32, 57)
5     println("The result is $result")
6 }
7
```

## Higher-Order Functions:

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



```
untitled C:\Users\Danial\Desktop
├── .idea
├── out
└── src
    └── Main.kt
.gitignore
untitled.iml
External Libraries
Scratches and Consoles

Run MainKt x

C:\Users\Danial\.jdk\openjdk-23\bin\java.exe "-javaagent:C:\Users\Danial\AppData\Local\Pr
The result of addition is 123
The result of subtraction is 25
The result of multiplication is 17612
The result of division is 14

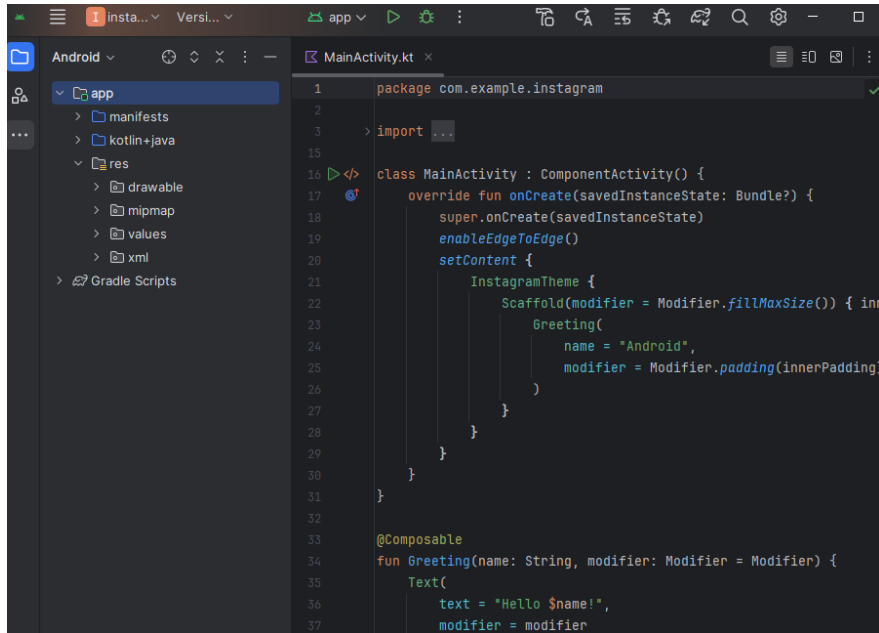
Process finished with exit code 0
```

```
1 fun highordfun(a: Int, b: Int, proc : (Int, Int) -> Int): Int {
2     return proc(a, b)
3 }
4
5 fun main() {
6     val addResult = highordfun(a: 95, b: 28) { a, b -> a + b }
7     val subtractionResult = highordfun(a: 64, b: 39) { a, b -> a - b }
8     val multiplyResult = highordfun(a: 74, b: 238) { a, b -> a * b }
9     val divisionResult = highordfun(a: 56, b: 4) { a, b -> a / b }
10
11     println("The result of addition is $addResult " +
12         "\nThe result of subtraction is $subtractionResult " +
13         "\nThe result of multiplication is $multiplyResult " +
14         "\nThe result of division is $divisionResult")
15 }
16
```

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

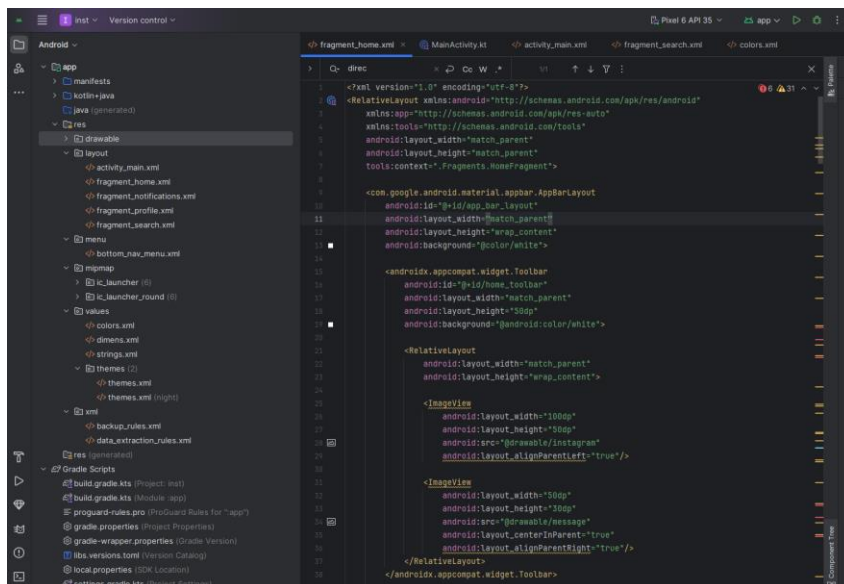
### 1. Set Up the Android Project:

- 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.

