

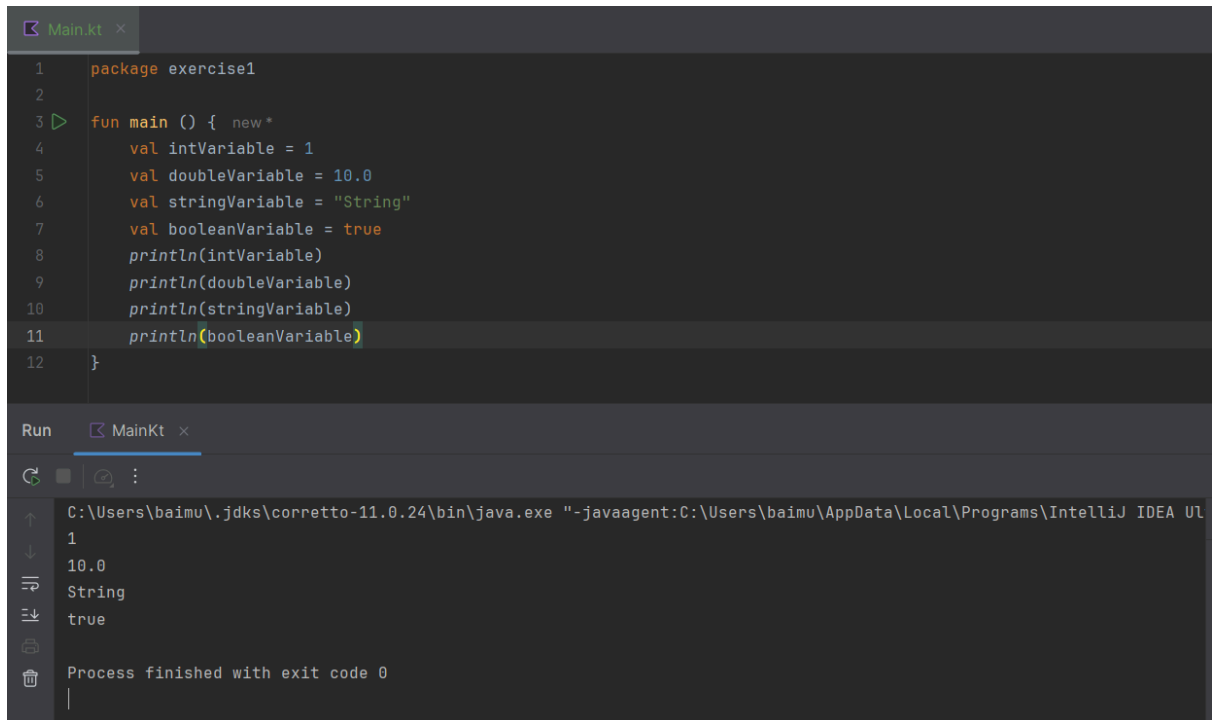
Assignment 1, Mobile Programming

Rafael Baimurzin

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 window with a Kotlin file named `Main.kt`. The code defines a package `exercise1` and a `main` function. Inside `main`, four variables are declared and initialized: `intVariable = 1`, `doubleVariable = 10.0`, `stringVariable = "String"`, and `booleanVariable = true`. Each variable is then printed using `println`. Below the code editor, the `Run` tab is active, showing the execution path and the output of the program. The output consists of four lines: `1`, `10.0`, `String`, and `true`. The process finished with exit code 0.

```
1 package exercise1
2
3 fun main () { new *
4     val intVariable = 1
5     val doubleVariable = 10.0
6     val stringVariable = "String"
7     val booleanVariable = true
8     println(intVariable)
9     println(doubleVariable)
10    println(stringVariable)
11    println(booleanVariable)
12 }
```

Run MainKt x

C:\Users\baimu\.jdfs\corretto-11.0.24\bin\java.exe "-javaagent:C:\Users\baimu\AppData\Local\Programs\IntelliJ IDEA UI

1
10.0
String
true

Process finished with exit code 0

Conditional Statements:

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

```
task2.kt x
1 package exercise1
2
3 fun main() { new *
4     val num: Int = readln().toInt()
5     if (num > 0) {
6         println("Positive")
7     } else if (num == 0) {
8         println("Zero")
9     } else {
10        println("Negative")
11    }
12 }
```

Run Task2Kt x

C:\Users\baimu\.jdk\corretto-11.0.24\bin\java.exe "-javaagent:C:\Users\baimu\AppData\Local\..."

2

Positive

Process finished with exit code 0

Loops:

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

```
task3.kt x
1 package exercise1
2
3 fun main() { new *
4     print("FOR LOOP: ")
5     for (i in 1 .. 10) {
6         print("$i ")
7     }
8     print("\n")
9
10    var n = 1
11    print("WHILE LOOP: ")
12    while (n <= 10) {
13        print("$n ")
14        n++
15    }
16 }
```

Run Task3Kt x

C:\Users\baimu\.jdk\corretto-11.0.24\bin\java.exe "-javaagent:C:\Users\baimu\AppData\Local\..."

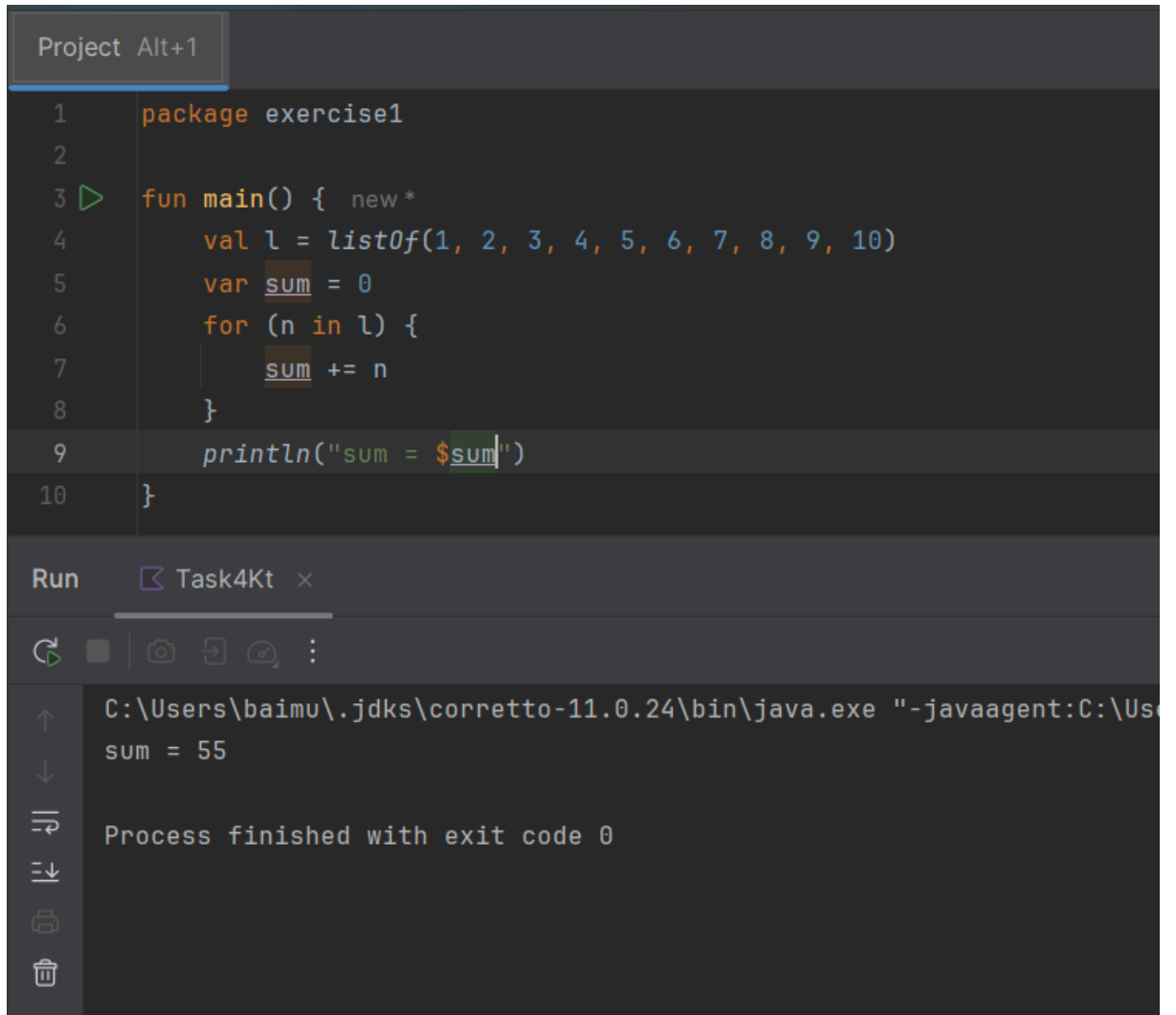
FOR LOOP: 1 2 3 4 5 6 7 8 9 10

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.



The screenshot displays an IDE window with a Kotlin file named 'Task4Kt'. The code defines a package 'exercise1' and a 'main' function. Inside 'main', a list 'l' is created with values from 1 to 10 using 'listOf'. A variable 'sum' is initialized to 0. A 'for' loop iterates over each element 'n' in the list, adding it to 'sum'. Finally, 'println' outputs the value of 'sum'. Below the code editor, the 'Run' tab shows the command executed: 'C:\Users\baimu\jdk\corretto-11.0.24\bin\java.exe "-javaagent:C:\Users\baimu\jdk\corretto-11.0.24\bin\javaagent.jar"'. The output is 'sum = 55', and the status indicates 'Process finished with exit code 0'.

```
1 package exercise1
2
3 fun main() {
4     val l = listOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
5     var sum = 0
6     for (n in l) {
7         sum += n
8     }
9     println("sum = $sum")
10 }
```

Run Task4Kt x

C:\Users\baimu\jdk\corretto-11.0.24\bin\java.exe "-javaagent:C:\Users\baimu\jdk\corretto-11.0.24\bin\javaagent.jar"

sum = 55

Process finished with exit code 0

Exercise 2: Kotlin OOP (Object-Oriented Programming)

Create a **Person** class:

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

```
task1.kt x
1 package exercise2
2
3 open class Person( new *
4     val name: String,
5     val age: Int,
6     val email: String
7 ) {
8     open fun displayInfo() = println("name='$name', age=$age, email='$email'")
9 }
10
11 fun main() { new *
12     val person = Person(
13         name = "Rafael",
14         age = 22,
15         email = "rafael@example.com"
16     )
17     person.displayInfo()
18 }
```

Run Task1Kt (1) x

C:\Users\baimu\.jdk\corretto-11.0.24\bin\java.exe "-javaagent:C:\Users\baimu\AppData\Local\Temp\jvarkit\jvarkit.jar" name='Rafael', age=22, email='rafael@example.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.

```
task2.kt x
1 package exercise2
2
3 class Employee( new *
4     name: String,
5     age: Int,
6     email: String,
7     val salary: Int
8 ) : Person(name, age, email) {
9     override fun displayInfo() = println("name='$name', age=$age, email='$email', salary=$salary")
10 }
11
12 fun main() { new *
13     val employee = Employee(
14         name = "Rafael",
15         age = 22,
16         email = "rafael@example.com",
17         salary = 1_000_000
18     )
19     employee.displayInfo()
20 }
```

Run Task2Kt (1) x

```
C:\Users\baimu\.jdk\corretto-11.0.24\bin\java.exe "-javaagent:C:\Users\baimu\AppData\Local\Programs\Int
name='Rafael', age=22, email='rafael@example.com', salary=1000000
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.

```
task3.kt x
1 package exercise3
2
3 class BankAccount { new *
4     private var balance = 0
5
6     fun deposit(value: Int) { new *
7         balance += value
8         showCurrentBalance()
9     }
10
11     fun withdraw(value: Int) { new *
12         if (balance < value) {
13             println("Balance can't be negative")
14             return
15         }
16         balance -= value
17         showCurrentBalance()
18     }
19
20     private fun showCurrentBalance() = println("Current balance: $balance") new *
21 }
22
23 fun main() { new *
24     val bankAccount = BankAccount()
25     bankAccount.deposit(value: 100)
26     bankAccount.withdraw(value: 100)
27     bankAccount.withdraw(value: 1)
28 }
```

Run Task3Kt (1) x

C:\Users\baimu\.jdk\corretto-11.0.24\bin\java.exe "-javaagent:C:\U
Current balance: 100
Current balance: 0
Balance can't be negative
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

```
task1.kt x
1 package exervice3
2
3 fun sum(n1: Int, n2: Int) = n1 + n2 new *
4
5 fun main() { new *
6     println(sum(n1: 1, n2: 2))
7 }

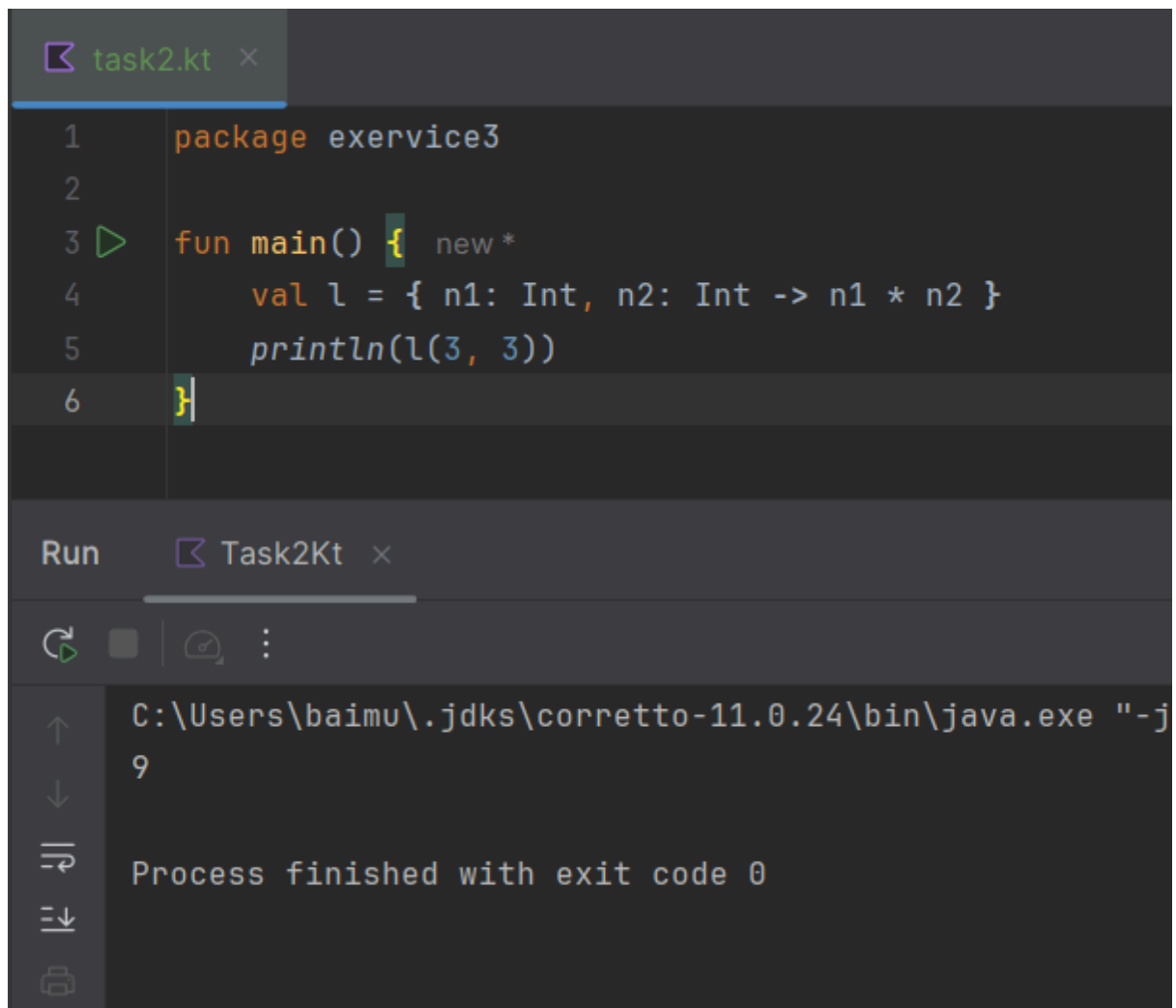
```

Run Task1Kt x

C:\Users\baimu\.jdk\corretto-11.0.24\bin\java.exe "-
3
Process finished with exit code 0

Lambda Functions:

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



The screenshot shows an IDE window titled 'task2.kt'. The code is as follows:

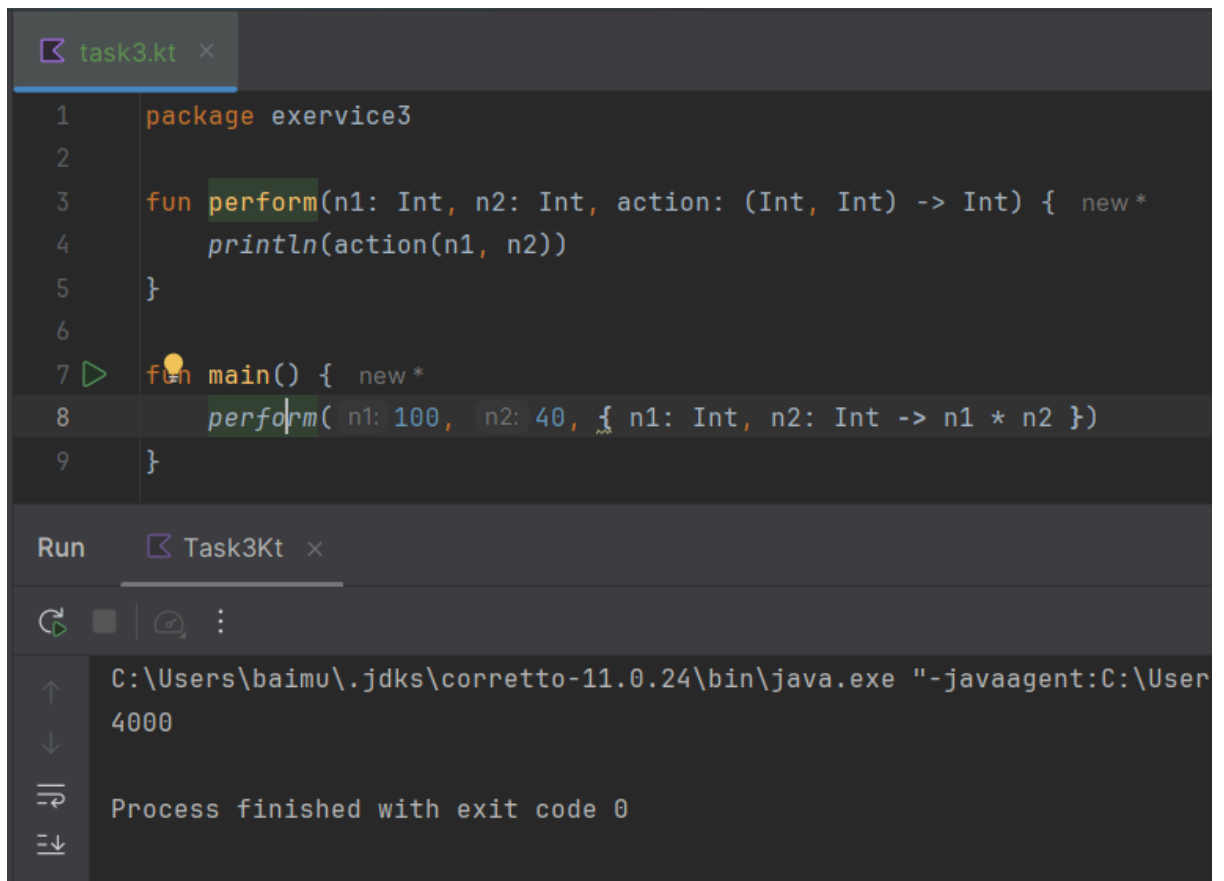
```
1 package exervice3
2
3 fun main() { new *
4     val l = { n1: Int, n2: Int -> n1 * n2 }
5     println(l(3, 3))
6 }
```

Below the code editor is a 'Run' tab titled 'Task2Kt'. It shows the execution command and the output:

```
C:\Users\baimu\.jdk\corretto-11.0.24\bin\java.exe "-j
9
Process finished with exit code 0
```

Higher-Order Functions:

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



The screenshot shows an IDE window with a file named `task3.kt`. The code defines a package `exervice3`, a function `perform` that takes two integers and an action, and a `main` function that calls `perform` with the values 100 and 40, and a lambda action that multiplies the two integers. The code is as follows:

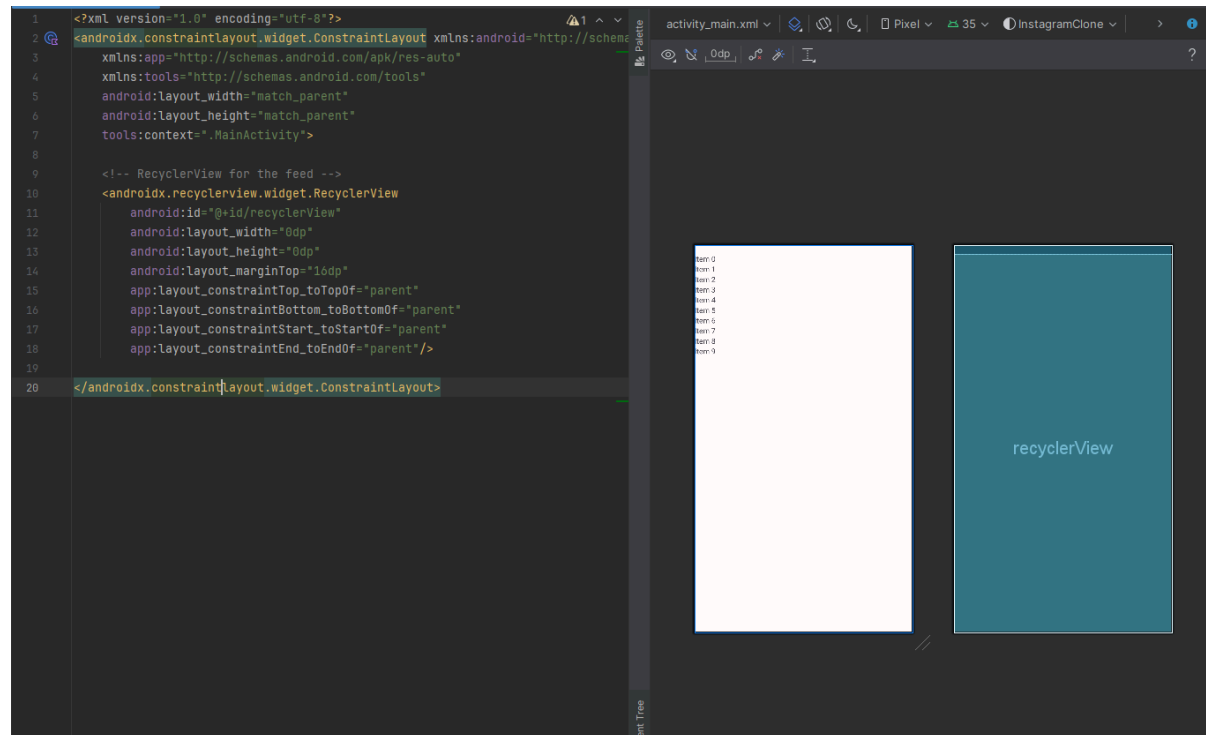
```
1 package exervice3
2
3 fun perform(n1: Int, n2: Int, action: (Int, Int) -> Int) { new *
4     println(action(n1, n2))
5 }
6
7 fun main() { new *
8     perform(n1: 100, n2: 40, { n1: Int, n2: Int -> n1 * n2 })
9 }
```

Below the code editor, the `Run` button is visible, and the output console shows the command used to run the program and the result:

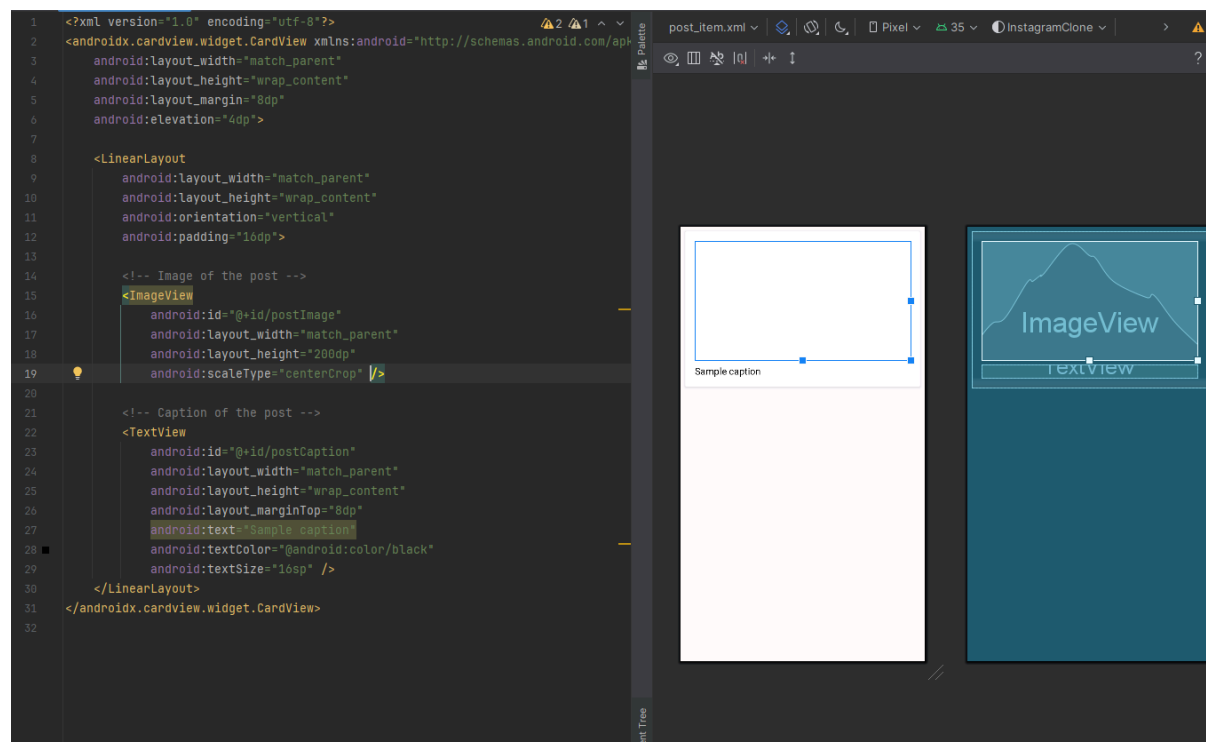
```
C:\Users\baimu\.jdk\corretto-11.0.24\bin\java.exe "-javaagent:C:\User
4000
Process finished with exit code 0
```

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.

```

1 package com.example.instagramclone.presentation
2
3 import android.view.LayoutInflater
4 import android.view.View
5 import android.view.ViewGroup
6 import android.widget.ImageView
7 import android.widget.TextView
8 import androidx.recyclerview.widget.RecyclerView
9 import com.example.instagramclone.R
10 import com.example.instagramclone.models.Post
11
12 class PostAdapter(
13     private val posts: List<Post>
14 ): RecyclerView.Adapter<PostAdapter.PostViewHolder>() {
15
16     class PostViewHolder(view: View): RecyclerView.ViewHolder(view) {
17         val postImage: ImageView = view.findViewById(R.id.postImage)
18         val postCaption: TextView = view.findViewById(R.id.postCaption)
19     }
20
21     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): PostViewHolder {
22         val view = LayoutInflater.from(parent.context).inflate(R.layout.post_item, parent, attachToRoot: false)
23         return PostViewHolder(view)
24     }
25
26     override fun onBindViewHolder(holder: PostViewHolder, position: Int) {
27         val post = posts[position]
28         holder.postImage.setImageResource(post.imageResource)
29         holder.postCaption.text = post.caption
30     }
31
32     override fun getItemCount(): Int {
33         return posts.size
34     }
35 }

```

MainActivity Setup:

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

```

1 package com.example.instagramclone
2
3 > import ...
12
13 class MainActivity : AppCompatActivity() {
14     override fun onCreate(savedInstanceState: Bundle?) {
15         super.onCreate(savedInstanceState)
16         enableEdgeToEdge()
17         setContentView(R.layout.activity_main)
18
19         val postService = PostServiceImpl()
20         val recyclerView = findViewById<RecyclerView>(R.id.recyclerView)
21         recyclerView.layoutManager = LinearLayoutManager(context, this)
22         recyclerView.adapter = PostAdapter(postService.getAll())
23     }
24 }

```