# Assignment 1, Mobile Programming
# Tussupbay Daulet

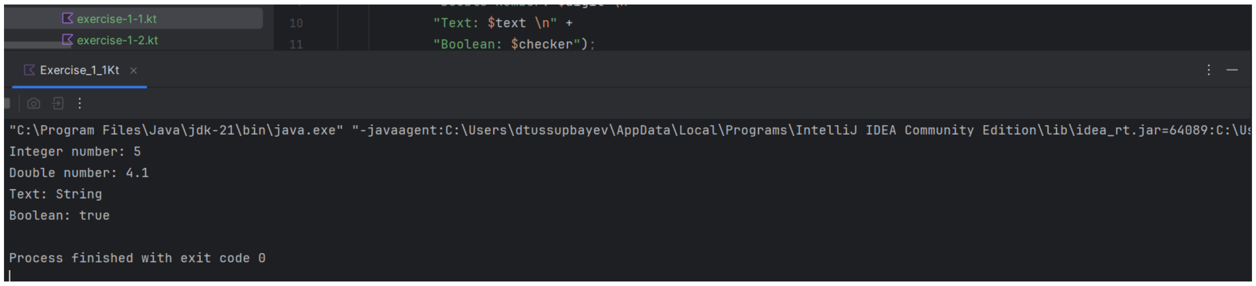
Put all deliverables into github repository in your profile. Share link to google form 24 hours before defense. Defend by explaining deliverables and answering questions. There should be proof that you did yourself.
Deliverables: report in pdf
Google form: https://docs.google.com/forms/d/e/1FAIpQLSe0GyNdOYlvM1tX_I_CtlPod5jBf-ACLGdHYZq1gVZbUeBzIg/viewform?usp=sf_link

## 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.

```
fun main(){
    val num: Int = 5;
    val digit: Double = 4.1;
    val text: String = "String";
    val checker: Boolean = true;

    println("Integer number: $num \n" +
            "Double number: $digit \n" +
            "Text: $text \n" +
            "Boolean: $checker");
}
```

Result:

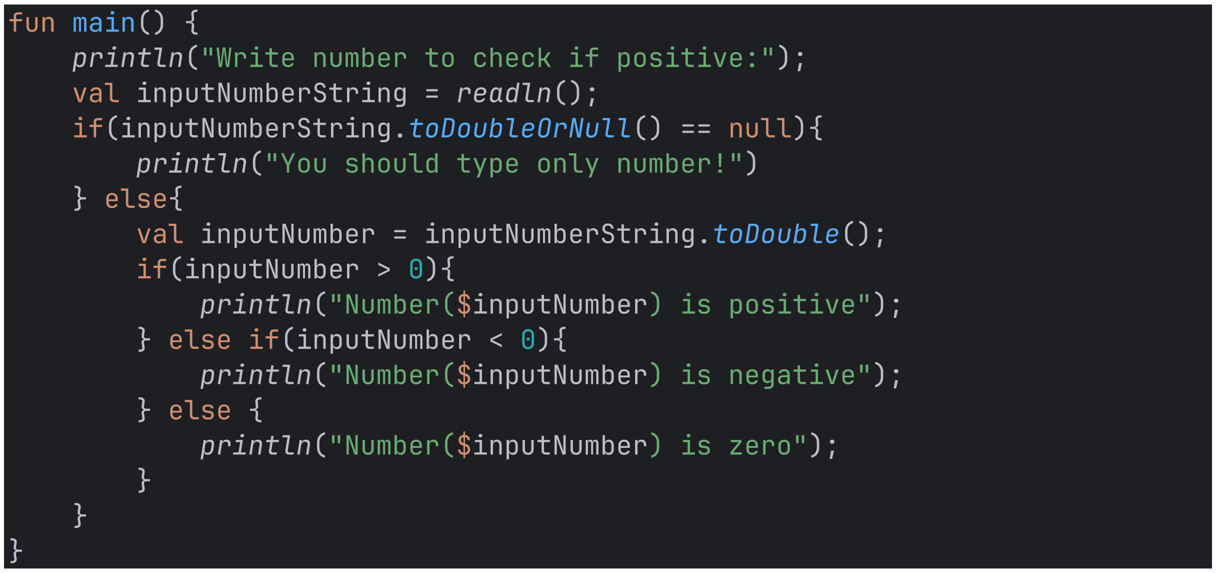


```
        "Text: $text \n" +
        "Boolean: $checker");
```

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Users\dtussupbayev\AppData\Local\Programs\IntelliJ IDEA Community Edition\lib\idea_rt.jar=64089:C:\Us
Integer number: 5
Double number: 4.1
Text: String
Boolean: true

Process finished with exit code 0
```

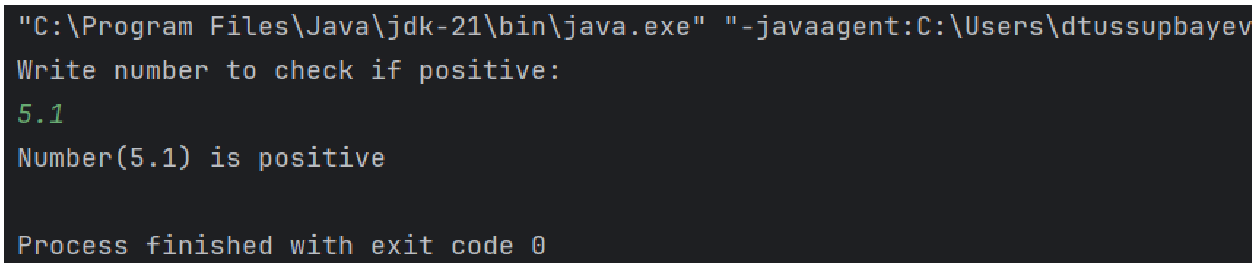
**Conditional Statements:**

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



```
fun main() {
    println("Write number to check if positive:");
    val inputNumberString = readln();
    if(inputNumberString.toDoubleOrNull() == null){
        println("You should type only number!")
    } else{
        val inputNumber = inputNumberString.toDouble();
        if(inputNumber > 0){
            println("Number($inputNumber) is positive");
        } else if(inputNumber < 0){
            println("Number($inputNumber) is negative");
        } else {
            println("Number($inputNumber) is zero");
        }
    }
}
```

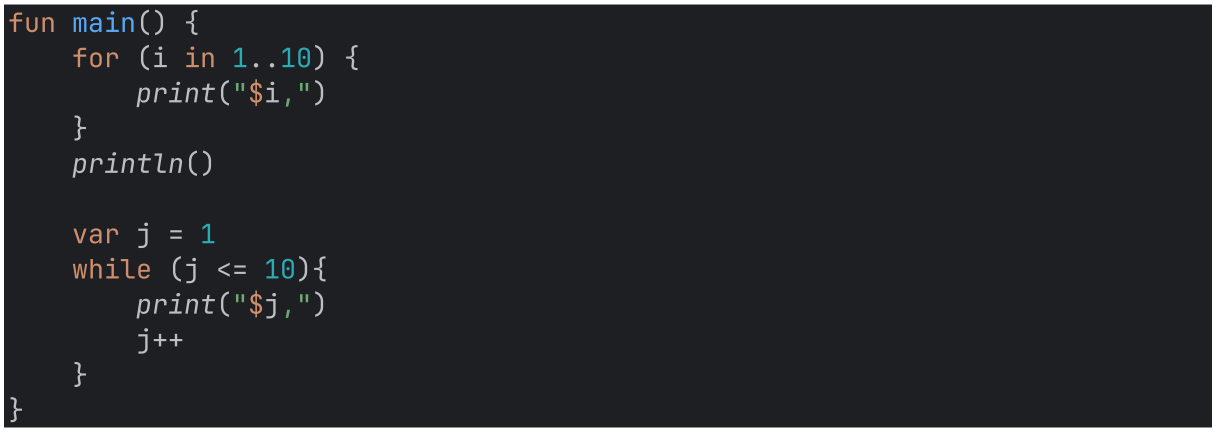
Result:



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Users\dtussupbayev
Write number to check if positive:
5.1
Number(5.1) is positive

Process finished with exit code 0
```

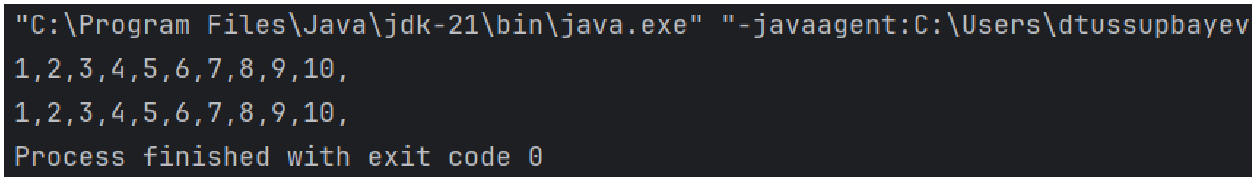
**Loops:**

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



```
fun main() {
    for (i in 1..10) {
        print("$i,")
    }
    println()

    var j = 1
    while (j <= 10){
        print("$j,")
        j++
    }
}
```

Result:



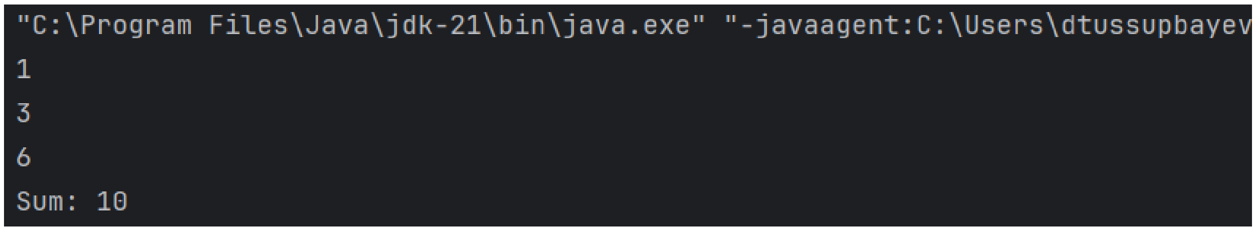
```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Users\dtussupbayev
1,2,3,4,5,6,7,8,9,10,
1,2,3,4,5,6,7,8,9,10,
Process finished with exit code 0
```

**Collections:**

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

```kotlin
fun main() {
    val numbers = listOf(1,3,6)
    for (number in numbers){
        println(number)
    }
    println("Sum: ${numbers.sum()}")
}
```

Result:



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Users\dtussupbayev
1
3
6
Sum: 10
```

## 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.

```kotlin
2. open class Person(val name: String,val age: Int, val email: String,) {
       open fun displayInfo (){
           println("Name: ${this.name}, age: ${this.age}, email:
   ${this.email}")
       }
   }
```

**Inheritance:**

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

```
● class Employee(name: String, age: Int, email: String, val salary: Int) :
  Person(name, age, email) {
      override fun displayInfo () {
          println("Name: ${this.name}, age: ${this.age}, email:
  ${this.email}, salary: $$salary")
      }
  }
```

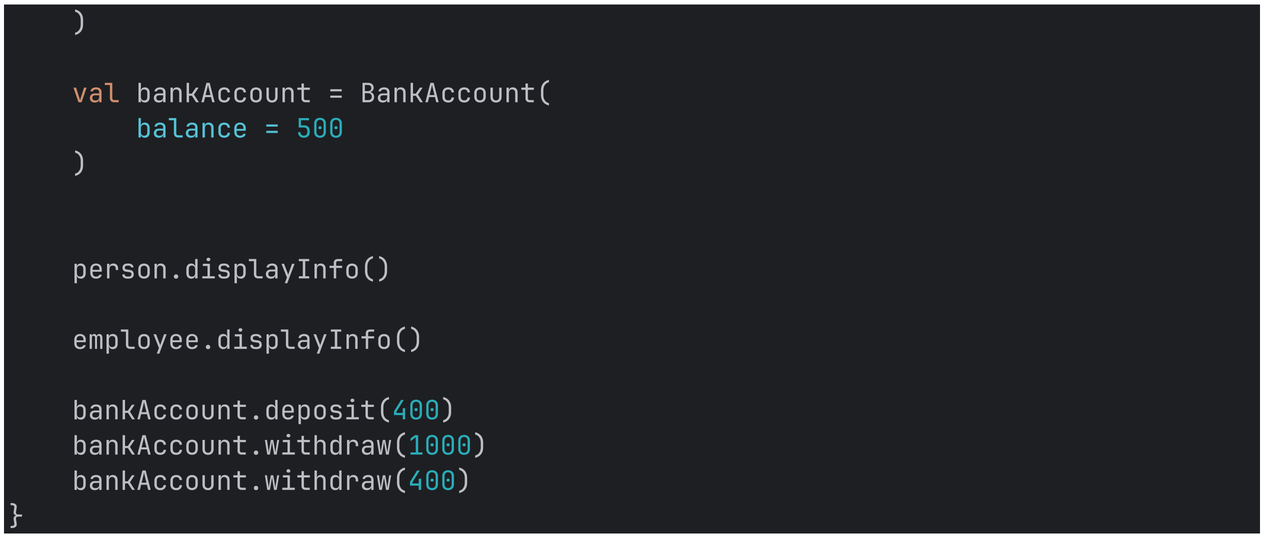
**Encapsulation:**

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

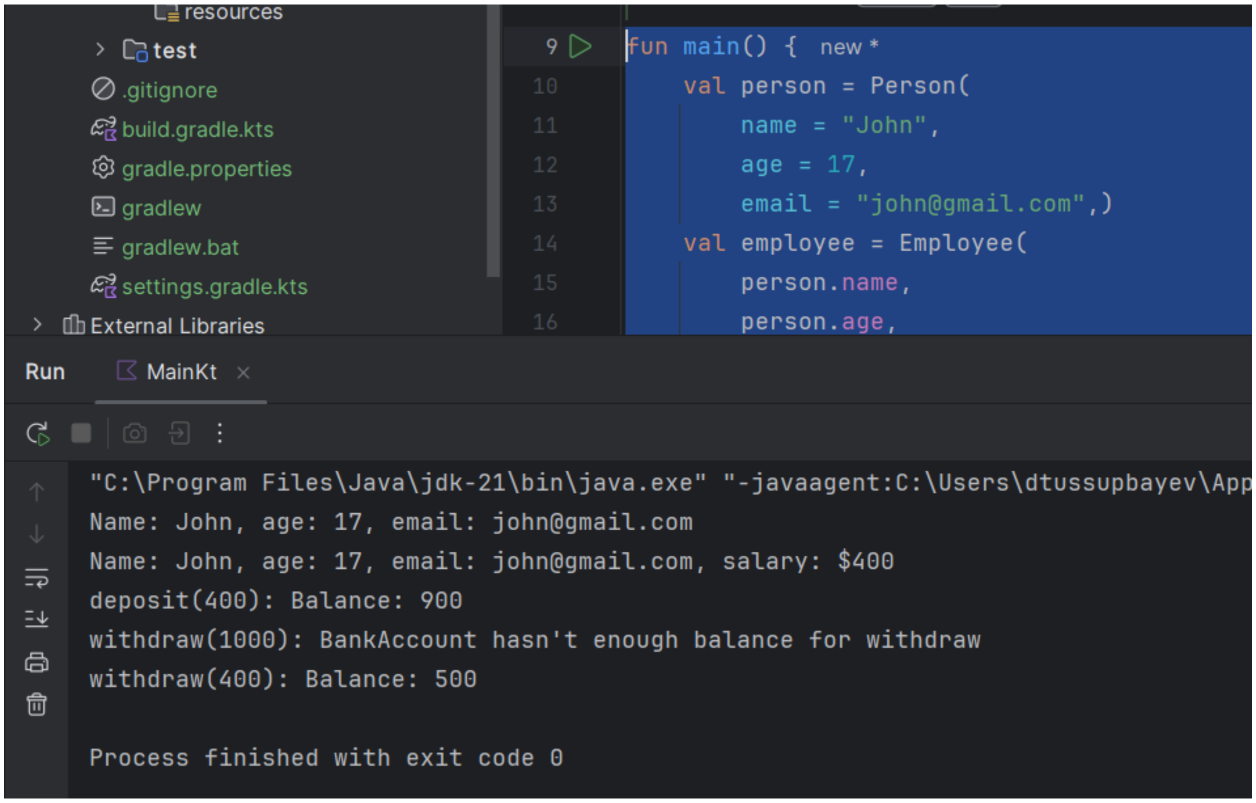
```
class BankAccount(var balance: Int) {

    fun deposit(amount: Int){
        balance+=amount
        println("deposit($amount): ${displayInfo()}")
    }

    fun withdraw(amount: Int){
        if(balance - amount >= 0){
            balance-=amount
            println("withdraw($amount): ${displayInfo()}")
        }else{
            println("withdraw($amount): BankAccount hasn't enough balance
for withdraw")
        }
    }

    private fun displayInfo(): String{
        return "Balance: $balance"
    }
}
```

## Result:

```
fun main() {
    val person = Person(
        name = "John",
        age = 17,
        email = "john@gmail.com",)
    val employee = Employee(
        person.name,
        person.age,
        person.email,
        salary = 400
```



```
    )

    val bankAccount = BankAccount(
        balance = 500
    )


    person.displayInfo()

    employee.displayInfo()

    bankAccount.deposit(400)
    bankAccount.withdraw(1000)
    bankAccount.withdraw(400)
}
```



```
fun main() {  new *
    val person = Person(
        name = "John",
        age = 17,
        email = "john@gmail.com",)
    val employee = Employee(
        person.name,
        person.age,
```

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Users\dtussupbayev\App
Name: John, age: 17, email: john@gmail.com
Name: John, age: 17, email: john@gmail.com, salary: $400
deposit(400): Balance: 900
withdraw(1000): BankAccount hasn't enough balance for withdraw
withdraw(400): Balance: 500

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

**Lambda Functions:**

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

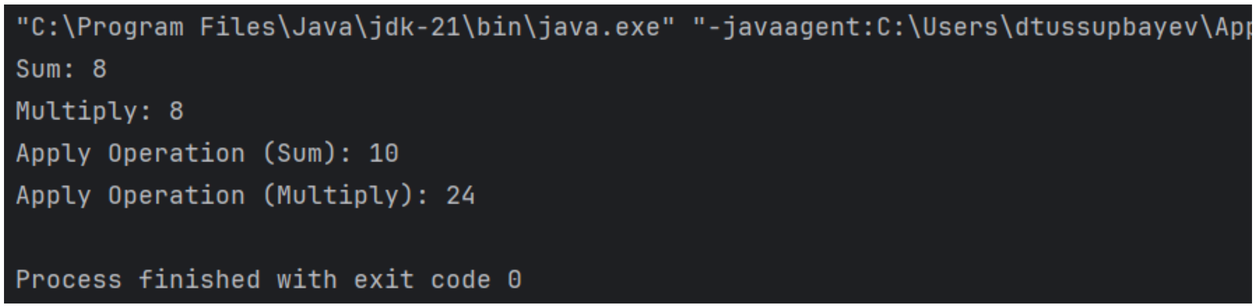
**Higher-Order Functions:**

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

```kotlin
// Basic function
fun sum(a:Int, b: Int): Int{
    return a + b;
}

// Multiply lambda function
val multiply: (Int, Int) -> Int = { x, y -> x * y }

// Pass lambda to function
fun applyOperation(a: Int, b: Int, operation: (Int, Int) -> Int): Int {
    return operation(a, b)
}
```

Main:

```kotlin
fun main() {
    // Test sum function
    val resultSum = sum(5, 3)
    println("Sum: $resultSum")

    // Test multiply lambda function
    val resultMultiply = multiply(4, 2)
    println("Multiply: $resultMultiply")

    // Test applyOperation function with sum
    val resultApplySum = applyOperation(7, 3, ::sum)
    println("Apply Operation (Sum): $resultApplySum")

    // Test applyOperation function with multiply
    val resultApplyMultiply = applyOperation(6, 4, multiply)
    println("Apply Operation (Multiply): $resultApplyMultiply")
}
```



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Users\dtussupbayev\App
Sum: 8
Multiply: 8
Apply Operation (Sum): 10
Apply Operation (Multiply): 24

Process finished with exit code 0
```

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

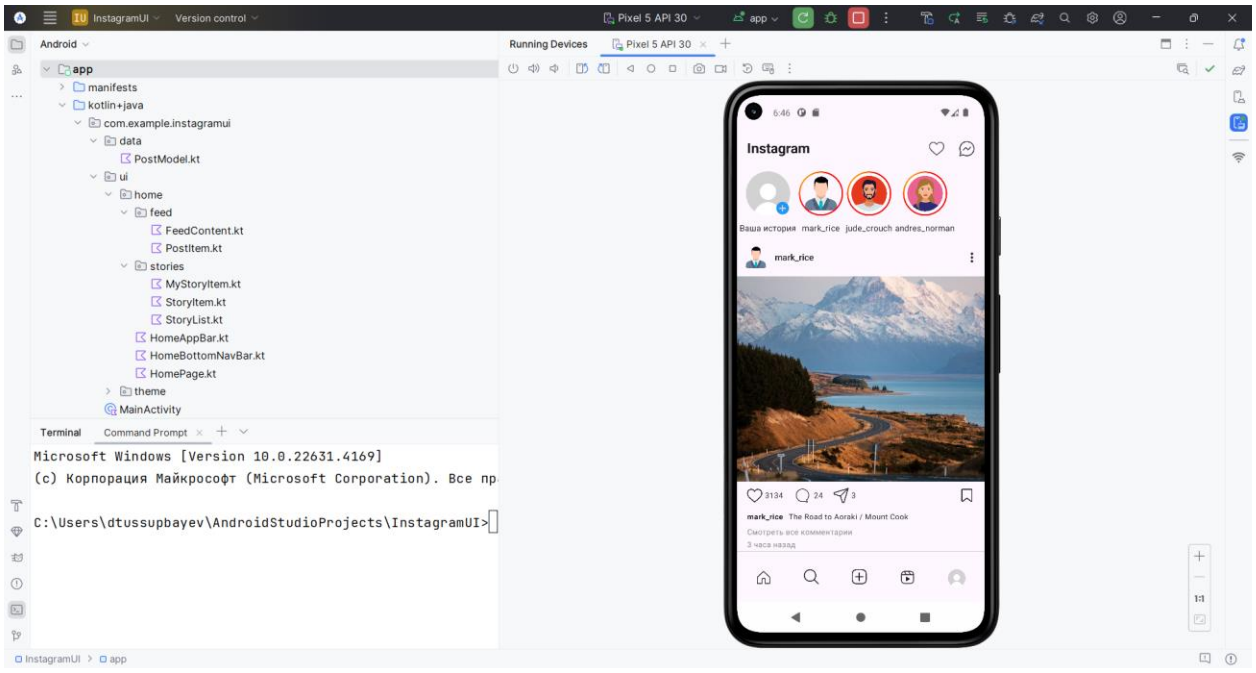
1. **Set Up the Android Project:**
   ○ Create a new Android project in Android Studio.
   ○ Ensure you have a Kotlin-based project.
2. **Design the Layout:**
   ○ Create a new XML layout file (activity_main.xml) for a simple Instagram-like user interface.
   ○ Include elements like ImageView, TextView, 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



```
Microsoft Windows [Version 10.0.22631.4169]
(c) Корпорация Майкрософт (Microsoft Corporation). Все пр

C:\Users\dtussupbayev\AndroidStudioProjects\InstagramUI>
```

```kotlin
@Composable
fun FeedContent(modifier: Modifier = Modifier, posts: List<PostModel>) {
    LazyColumn(
        modifier = modifier
    ) {
        item {
            Stories(posts)
        }
        items(posts) { post ->
            PostItem(post)
        }
    }
}
```

Used LazyColumn Composable instead of RecyclerView