



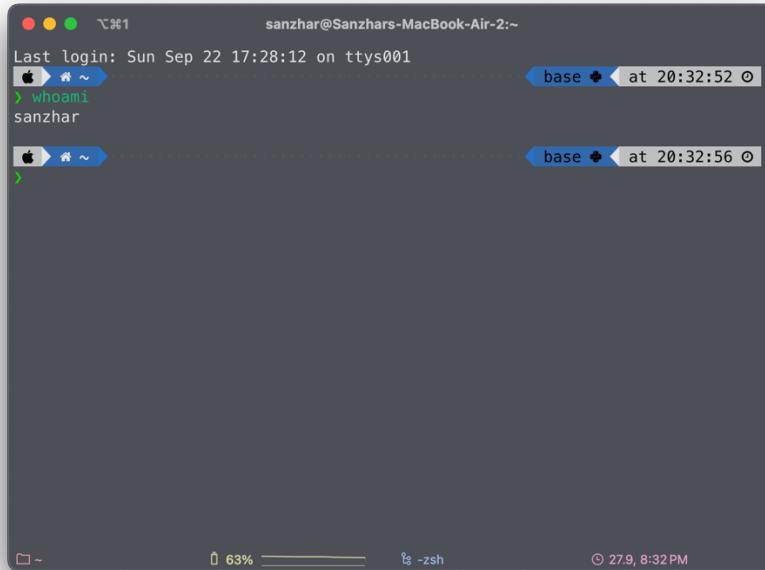
KAZAKH-BRITISH
TECHNICAL
UNIVERSITY

Assignment #1
Mobile Programming

Prepared by:
Seitbekov S.
Checked by:
Serek A.

Almaty, 2024

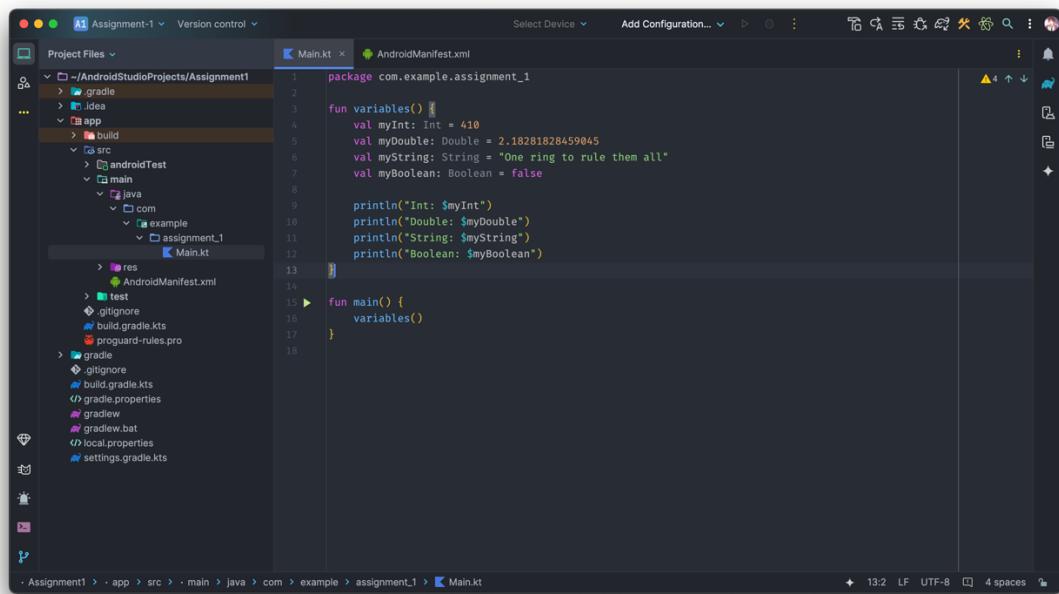
Exercise 1: Kotlin Syntax Basics



```
sanzhar@sanzhars-MacBook-Air-2:~  
Last login: Sun Sep 22 17:28:12 on ttys001  
base ✘ at 20:32:52 ⚡  
whoami  
sanzhar  
base ✘ at 20:32:56 ⚡  
↳
```

1. Variables and Data Types:

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



The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the following Kotlin code in the Main.kt file:

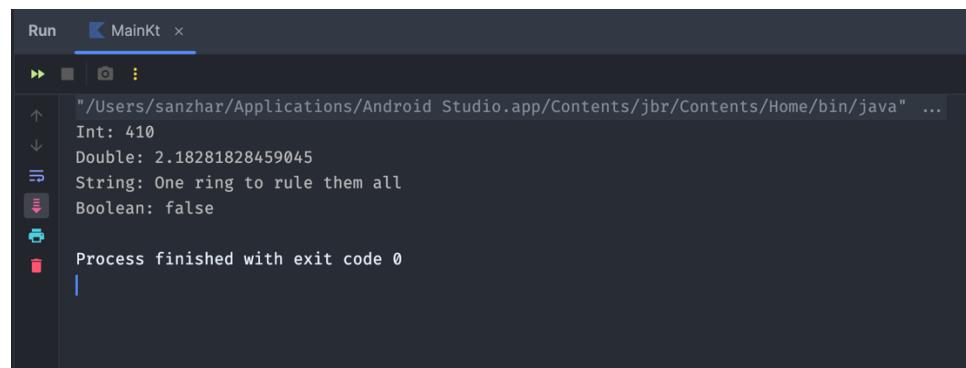
```
package com.example.assignment_1

fun variables() {
    val myInt: Int = 410
    val myDouble: Double = 2.18281828459045
    val myString: String = "One ring to rule them all"
    val myBoolean: Boolean = false

    println("Int: $myInt")
    println("Double: $myDouble")
    println("String: $myString")
    println("Boolean: $myBoolean")
}

fun main() {
    variables()
}
```

- Print the variables using `println`.

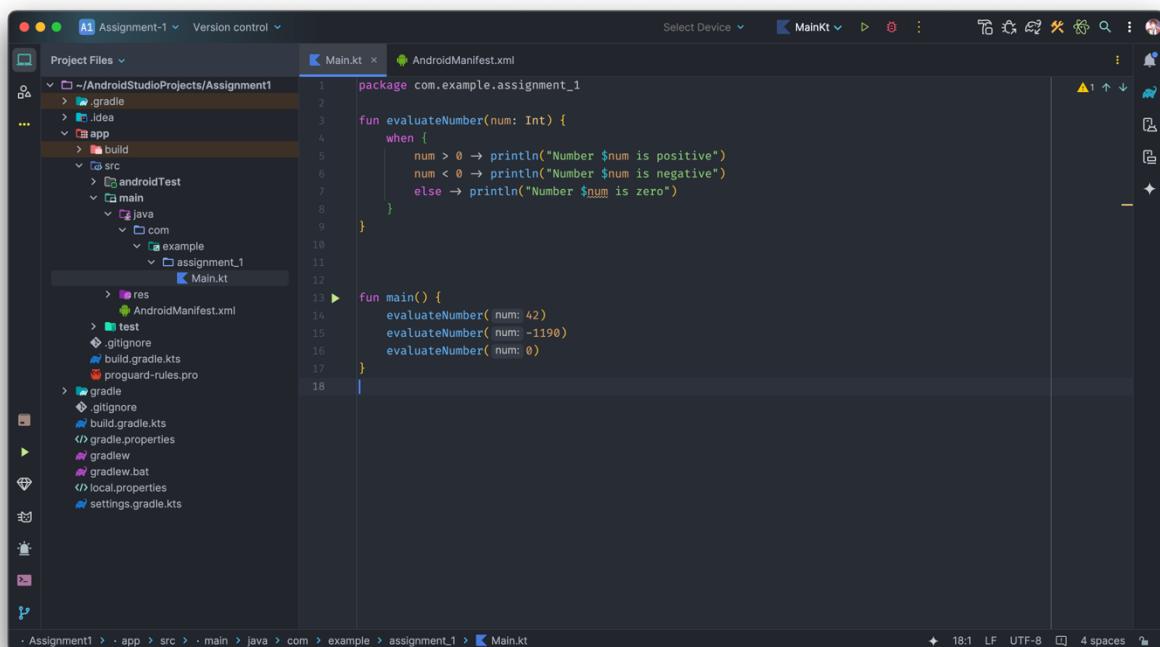


```
"/Users/sanzhar/Applications/Android Studio.app/Contents/jbr/Contents/Home/bin/java" ...
Int: 410
Double: 2.18281828459045
String: One ring to rule them all
Boolean: false

Process finished with exit code 0
```

Conditional Statements:

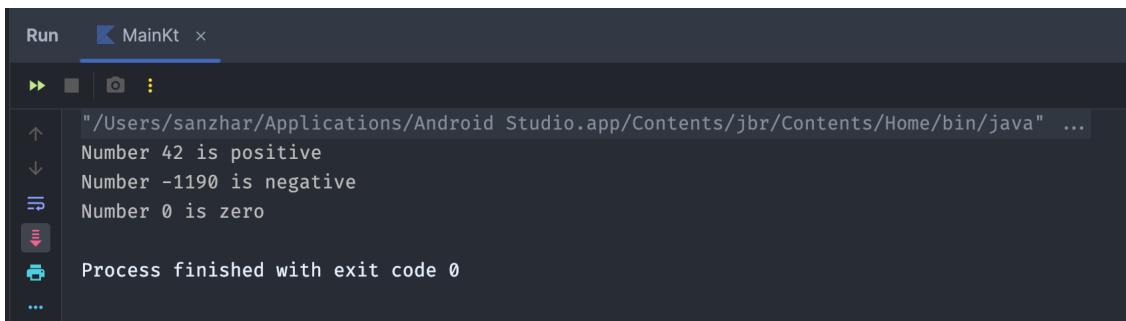
- Create a simple program that checks if a number is positive, negative, or zero.



```
package com.example.assignment_1

fun evaluateNumber(num: Int) {
    when {
        num > 0 -> println("Number $num is positive")
        num < 0 -> println("Number $num is negative")
        else -> println("Number $num is zero")
    }
}

fun main() {
    evaluateNumber( num: 42)
    evaluateNumber( num: -1190)
    evaluateNumber( num: 0)
}
```

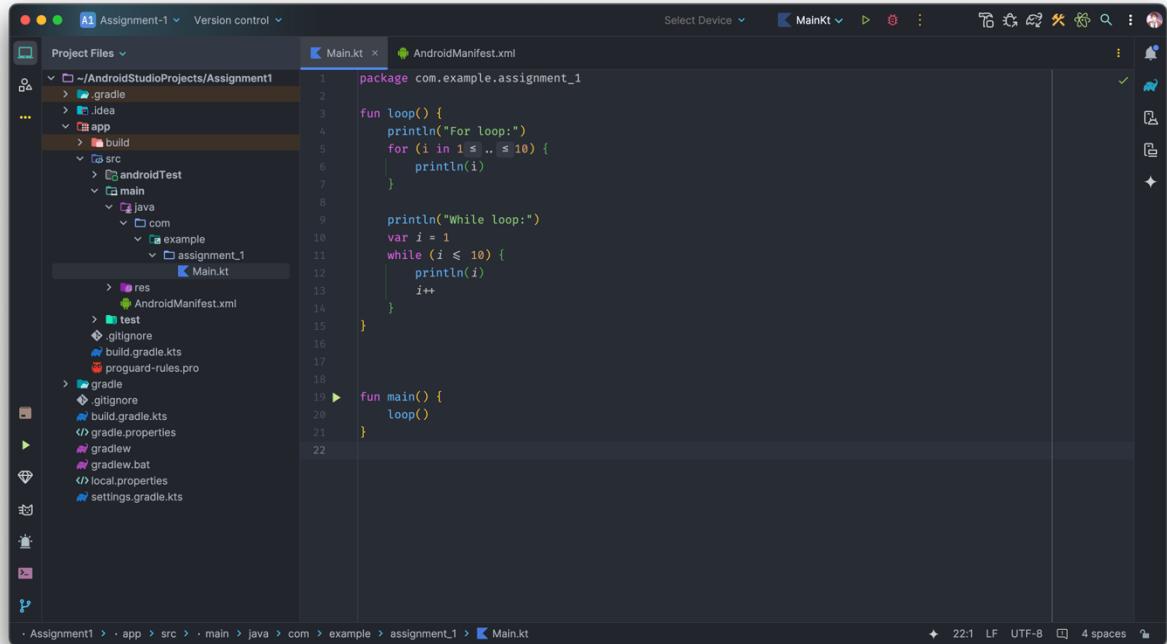


```
"/Users/sanzhar/Applications/Android Studio.app/Contents/jbr/Contents/Home/bin/java" ...
Number 42 is positive
Number -1190 is negative
Number 0 is zero

Process finished with exit code 0
```

Loops:

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



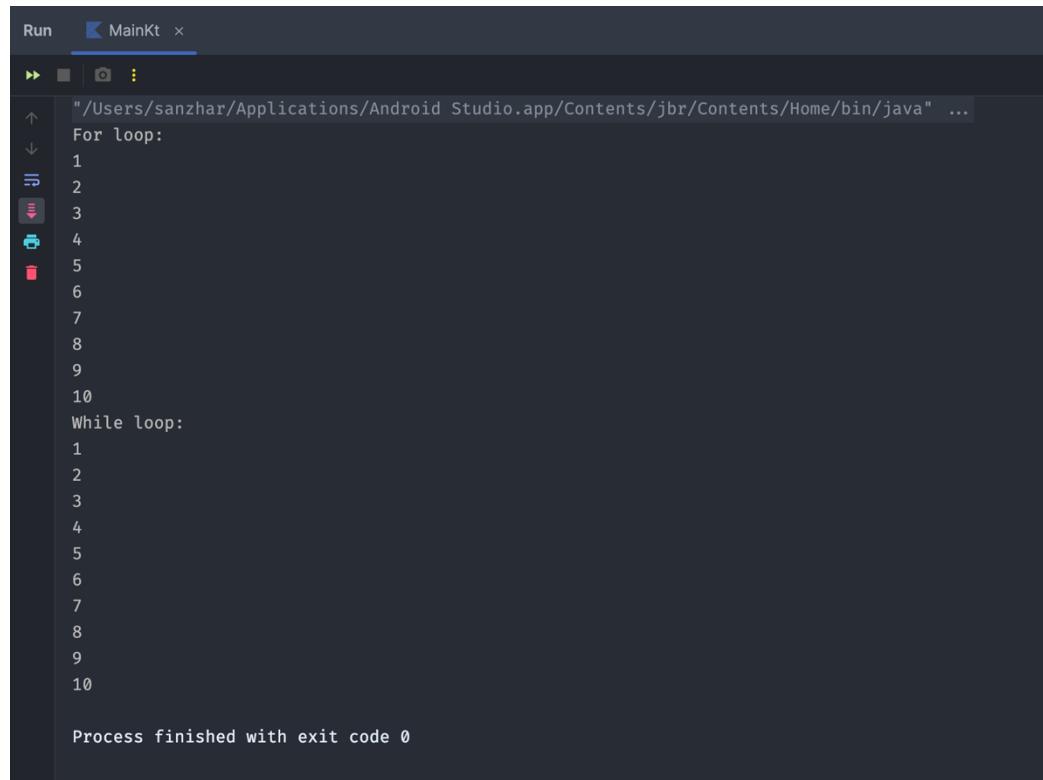
The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the `Main.kt` file containing the following Kotlin code:

```
package com.example.assignment_1

fun loop() {
    println("For loop:")
    for (i in 1 .. 10) {
        println(i)
    }

    println("While loop:")
    var i = 1
    while (i <= 10) {
        println(i)
        i++
    }
}

fun main() {
    loop()
}
```

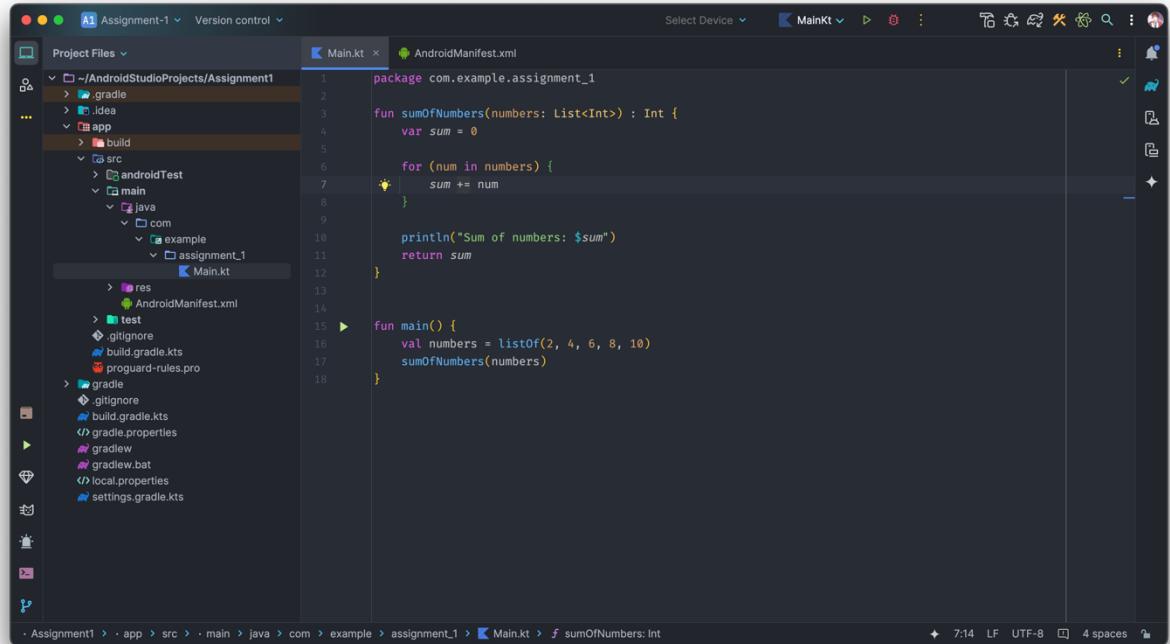


The screenshot shows the Run tab in Android Studio with the `MainKt` configuration selected. The output window displays the following text:

```
"~/Users/sanzhar/Applications/Android Studio.app/Contents/jbr/Contents/Home/bin/java" ...  
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 shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the following Kotlin code in the Main.kt file:

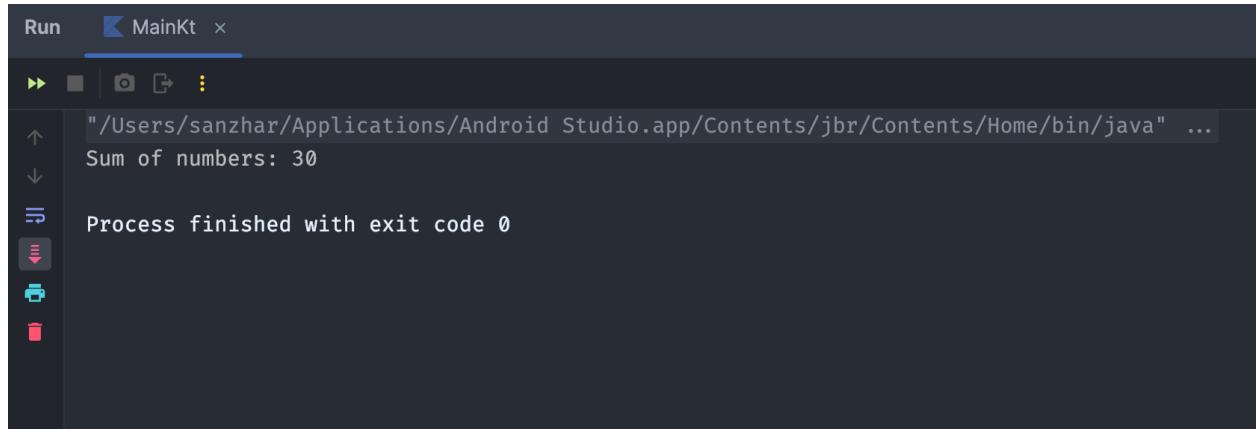
```
package com.example.assignment_1

fun sumOfNumbers(numbers: List<Int>) : Int {
    var sum = 0

    for (num in numbers) {
        sum += num
    }

    println("Sum of numbers: $sum")
    return sum
}

fun main() {
    val numbers = listOf(2, 4, 6, 8, 10)
    sumOfNumbers(numbers)
}
```



The screenshot shows the Run tab in Android Studio with the output of the program. The output window displays the following text:

```
"/Users/sanzhar/Applications/Android Studio.app/Contents/jbr/Contents/Home/bin/java" ...
Sum of numbers: 30

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.

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the following Kotlin code:

```
package com.example.assignment_1

class Person(private val name: String, private val age: Int, private val email: String) {
    fun displayInfo(): String {
        return "Name: $name, Age: $age, Email: $email"
    }
}

fun main() {
    val employee = Person(name = "Haruhi Suzumiya", age = 19, email = "john.smith@example.com")
    println(employee.displayInfo())
}
```

The screenshot shows the Run tab in Android Studio. The output window displays the following text:

```
"~/Users/sanzhar/Applications/Android.app/Contents/jbr/Contents/Home/bin/java" ...
Name: Haruhi Suzumiya, Age: 19, Email: john.smith@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.

The screenshot shows the Android Studio interface with the Main.kt file open in the editor. The code defines a Person class with an open fun displayInfo() and an Employee class that overrides it to include salary information. A main function creates an Employee object and prints its display info.

```
package com.example.assignment_1
open class Person(private val name: String, private val age: Int, private val email: String) {
    open fun displayInfo(): String {
        return "Name: $name, Age: $age, Email: $email"
    }
}

class Employee(name: String, age: Int, email: String, private val salary: Double) : Person(name, age, email) {
    override fun displayInfo(): String {
        return super.displayInfo() + ", Salary: $salary"
    }
}

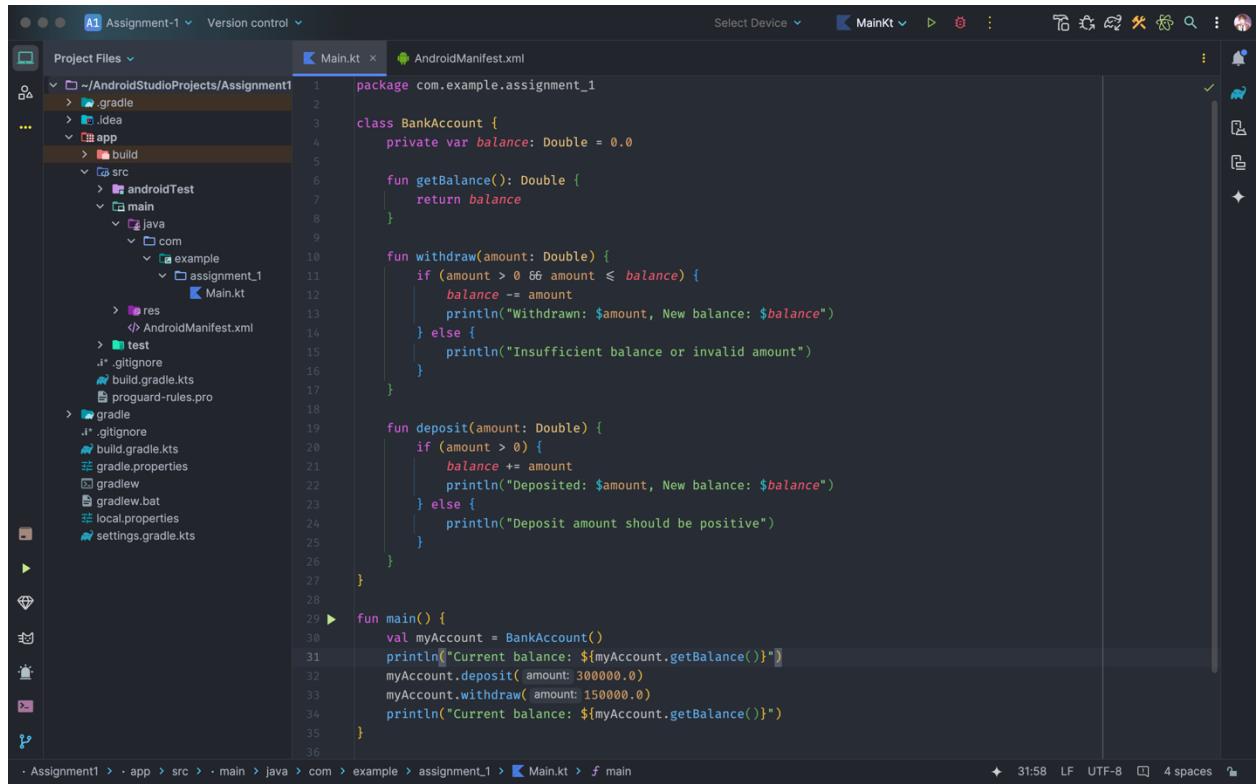
fun main() {
    val employee = Employee("Haruhi Suzumiya", 19, "john.smith@example.com", 200000.0)
    println(employee.displayInfo())
}
```

The screenshot shows the Run tab in Android Studio with the process output. It displays the printed output from the main() function, showing the employee's details and salary.

```
"/Users/sanzhar/Applications/Android Studio.app/Contents/jbr/Contents/Home/bin/java" ...
Name: Haruhi Suzumiya, Age: 19, Email: john.smith@example.com, Salary: 200000.0
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.



```
package com.example.assignment_1

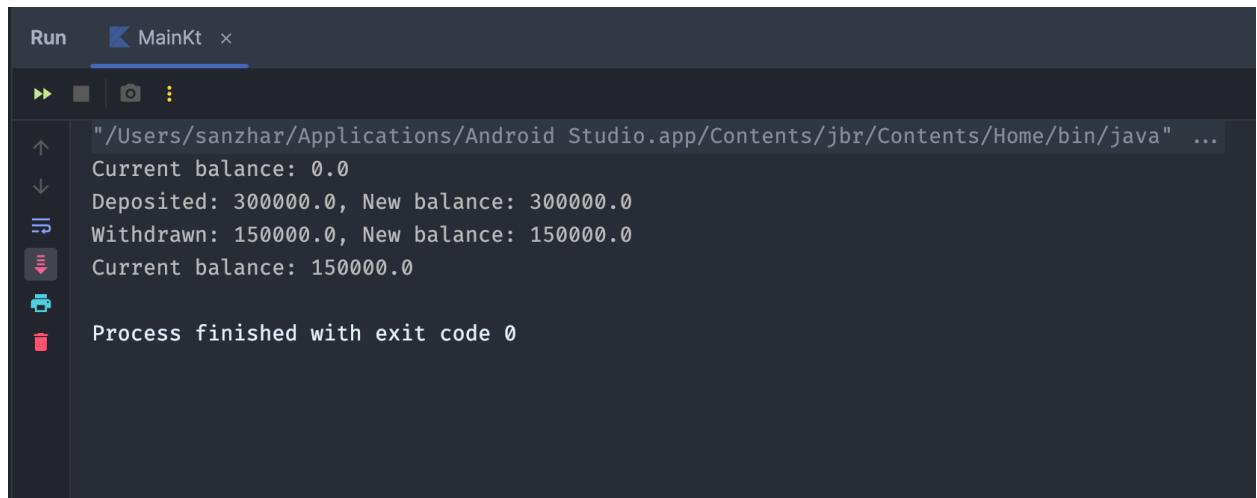
class BankAccount {
    private var balance: Double = 0.0

    fun getBalance(): Double {
        return balance
    }

    fun withdraw(amount: Double) {
        if (amount > 0 && amount <= balance) {
            balance -= amount
            println("Withdrawn: $amount, New balance: $balance")
        } else {
            println("Insufficient balance or invalid amount")
        }
    }

    fun deposit(amount: Double) {
        if (amount > 0) {
            balance += amount
            println("Deposited: $amount, New balance: $balance")
        } else {
            println("Deposit amount should be positive")
        }
    }
}

fun main() {
    val myAccount = BankAccount()
    println("Current balance: ${myAccount.getBalance()}")
    myAccount.deposit(amount: 300000.0)
    myAccount.withdraw(amount: 150000.0)
    println("Current balance: ${myAccount.getBalance()}")
}
```



```
"/Users/sanzhar/Applications/Android Studio.app/Contents/jbr/Contents/Home/bin/java" ...
Current balance: 0.0
Deposited: 300000.0, New balance: 300000.0
Withdrawn: 150000.0, New balance: 150000.0
Current balance: 150000.0

Process finished with exit code 0
```

Exercise 3: Kotlin Functions

1. Basic Function:

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

The screenshot shows the Android Studio interface. On the left is the Project Files tree, which includes the project structure: `~/AndroidStudioProjects/Assignment1`, `.gradle`, `.idea`, `app` (selected), `build`, `src` (with `androidTest`, `main` (containing `java`, `com`, `example`, `assignment_1`), and `res`), `test`, `.gitignore`, `build.gradle.kts`, `proguard-rules.pro`, `gradle` (with `.gitignore`, `build.gradle.kts`, `gradle.properties`, `gradlew`, `gradlew.bat`, `local.properties`, and `settings.gradle.kts`), and `gradle`. The right panel displays the `Main.kt` file content:

```
1 package com.example.assignment_1
2
3 fun sum(a: Int, b: Int): Int {
4     return a + b
5 }
6
7 fun main() {
8     val a: Int = 72
9     val b: Int = 36
10    println("Sum of integers: ${sum(a, b)}")
11 }
```

The screenshot shows the Run tab in Android Studio. It displays the command line output from running the `MainKt` file. The output shows the application's log and the result of the program execution:

```
"/Users/sanzhar/Applications/Android Studio.app/Contents/jbr/Contents/Home/bin/java" ...
Sum of integers: 108

Process finished with exit code 0
```

Lambda Functions:

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

The screenshot shows the Android Studio interface. On the left is the Project Files tree, which includes the project structure: ~-/AndroidStudioProjects/Assignment1, .gradle, .idea, app, build, src, androidTest, main, java, com, example, assignment_1, and Main.kt. The main editor window displays the Main.kt file with the following code:

```
1 package com.example.assignment_1
2
3 fun main() {
4     val multiply: (Int, Int) -> Int = { a, b -> a * b }
5     val a: Int = 120
6     val b: Int = 5
7     println("Multiplication of integers: ${multiply(a, b)}")
8 }
```

The status bar at the bottom shows the path Assignment1 > app > src > main > java > com > example > assignment_1 > Main.kt, and the bottom right corner shows file statistics: 9:1 LF, UTF-8, 4 spaces.

The screenshot shows the Run tab in Android Studio. It displays the output of the MainKt run. The output window shows the command used to run the application and the resulting output:

```
"/Users/sanzhar/Applications/Android Studio.app/Contents/jbr/Contents/Home/bin/java" ...
Multiplication of integers: 600

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 the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the following Kotlin code:

```
package com.example.assignment_1

fun higherOrderFunction(a: Int, b: Int, operation: (Int, Int) -> Int): Int {
    return operation(a, b)
}

fun main() {
    val multiply: (Int, Int) -> Int = { a, b -> a * b }
    val a: Int = 120
    val b: Int = 5
    val result: Int = higherOrderFunction(a, b, multiply)
    println("Multiplication of integers via higher order function: $result")
}
```

The screenshot shows the Run tab in Android Studio. The output window displays the following text:

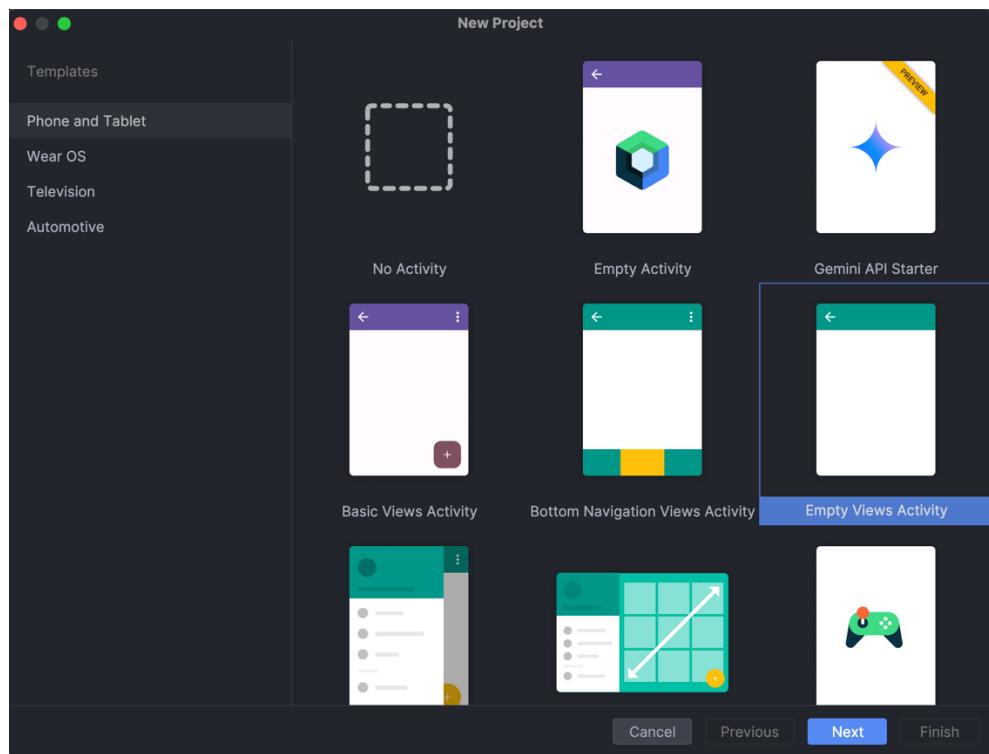
```
"/Users/sanzhar/Applications/Android Studio.app/Contents/jbr/Contents/Home/bin/java" ...
Multiplication of integers via higher order function: 600

Process finished with exit code 0
```

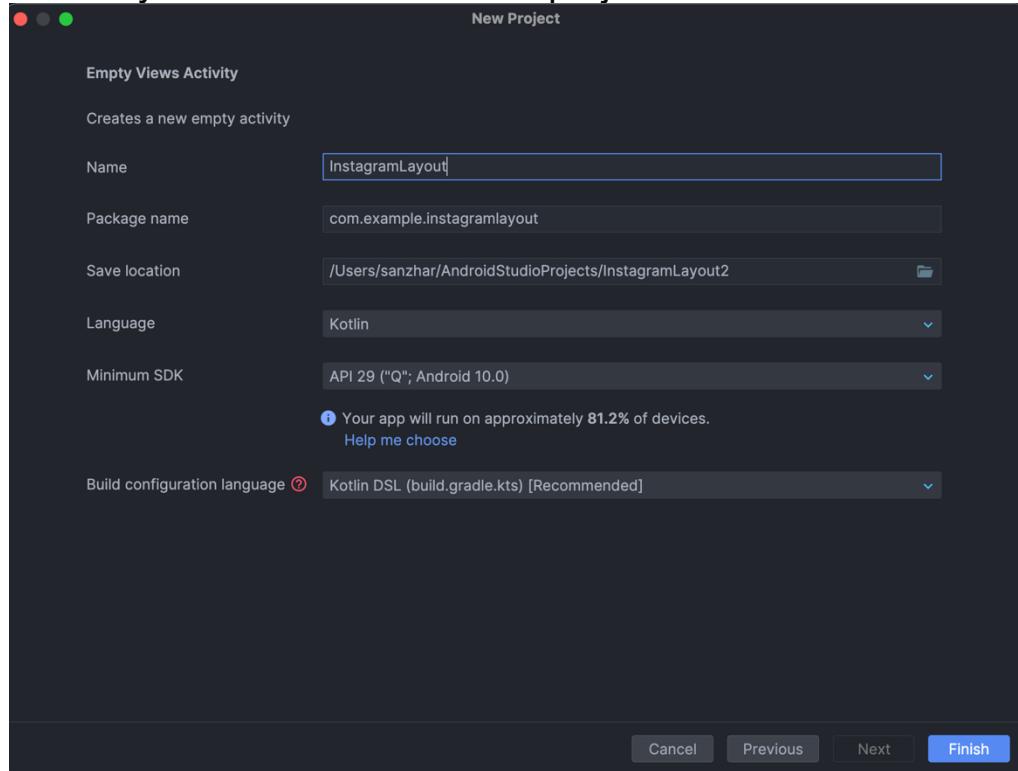
Exercise 4: Android Layout in Kotlin (Instagram-like Layout)

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.

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/profileSection"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="8dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent">

        <ImageView
            android:id="@+id/profileImage"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:src="@drawable/kotlin"
            android:scaleType="centerCrop"
            android:layout_marginEnd="8dp"
            app:layout_constraintStart_toEndOf="@+id/profileSection"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintBottom_toBottomOf="parent"/>

        <TextView
            android:id="@+id/profileUsername"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="I just started learning Kotlin!"
            android:textStyle="bold"
            android:textSize="18sp"
            app:layout_constraintStart_toEndOf="@+id/profileImage"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintBottom_toBottomOf="parent"/>
    

```

- Include elements like ImageView, TextView, and RecyclerView for the feed

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/postSection"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp"
        app:layout_constraintBottom_toBottomOf="parent">

        <ImageView
            android:id="@+id/postImage"
            android:layout_width="0dp"
            android:layout_height="300dp"
            android:scaleType="centerCrop"
            android:src="@drawable/kotlin"
            app:layout_constraintTop_toBottomOf="@+id/profileSection"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent"/>

        <TextView
            android:id="@+id/postDescription"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="I just started learning Kotlin!"
            android:textSize="16sp"
            android:padding="16dp"
            app:layout_constraintTop_toBottomOf="@+id/postImage"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent"/>
    

```

Create the RecyclerView Adapter:

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

Android Studio screenshot showing the Comment.kt file content. The code defines a data class Comment with properties: username (String), message (String), and reply (String? = null). The file is located in the com.example.instagramlayout package under the Comment adapter.

```
1 package com.example.instagramlayout
2
3 data class Comment(
4     val username: String,
5     val message: String,
6     val reply: String? = null
7 )
```

Android Studio screenshot showing the CommentAdapter.kt file content. The code defines a CommentAdapter class that extends RecyclerView.Adapter<CommentViewHolder>. It includes methods for onCreateViewHolder, onBindViewHolder, and getItemCount. The adapter uses a CommentViewHoder item view to display comments.

```
1 package com.example.instagramlayout
2
3 import android.view.LayoutInflater
4 import android.view.View
5 import android.view.ViewGroup
6 import android.widget.TextView
7 import androidx.recyclerview.widget.RecyclerView
8
9 class CommentAdapter(private val comments: List) :
10     RecyclerView.Adapter<CommentAdapter.CommentViewHolder>() {
11
12     class CommentViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
13         val usernameTextView: TextView = itemView.findViewById(R.id.username)
14         val messageTextView: TextView = itemView.findViewById(R.id.message)
15         val replyTextView: TextView = itemView.findViewById(R.id.reply)
16     }
17
18     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): CommentViewHolder {
19         val view = LayoutInflater.from(parent.context)
20             .inflate(R.layout.comment_item, parent, attachToRoot: false)
21         return CommentViewHolder(view)
22     }
23
24     override fun onBindViewHolder(holder: CommentViewHolder, position: Int) {
25         val comment = comments[position]
26         holder.usernameTextView.text = comment.username
27         holder.messageTextView.text = comment.message
28         if (comment.reply != null) {
29             holder.replyTextView.visibility = View.VISIBLE
30             holder.replyTextView.text = comment.reply
31         } else {
32             holder.replyTextView.visibility = View.GONE
33         }
34     }
35
36     override fun getItemCount() = comments.size
37
38 }
```

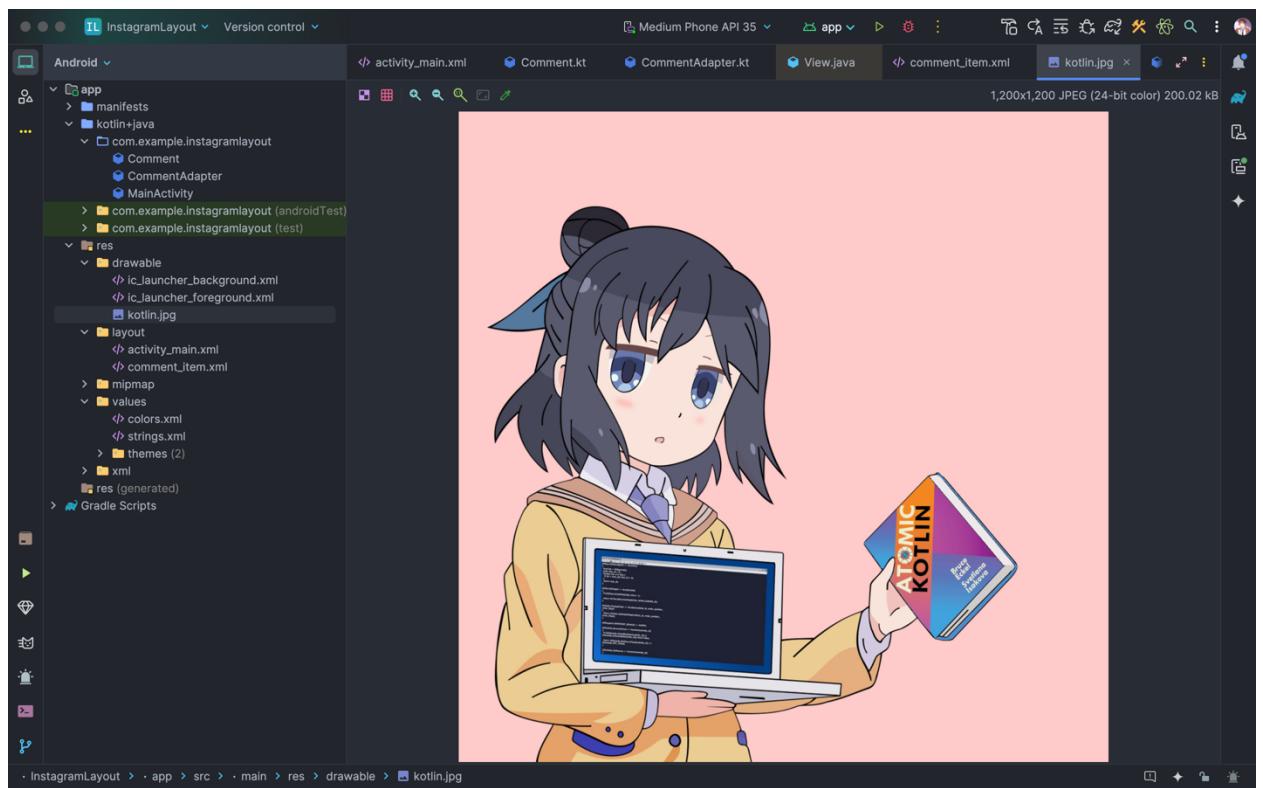
Screenshot of Android Studio showing the XML code for `comment_item.xml`. The code defines a ConstraintLayout containing three TextViews for a comment item.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="8dp">

    <TextView
        android:id="@+id/username"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Username"
        android:textStyle="bold"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"/>

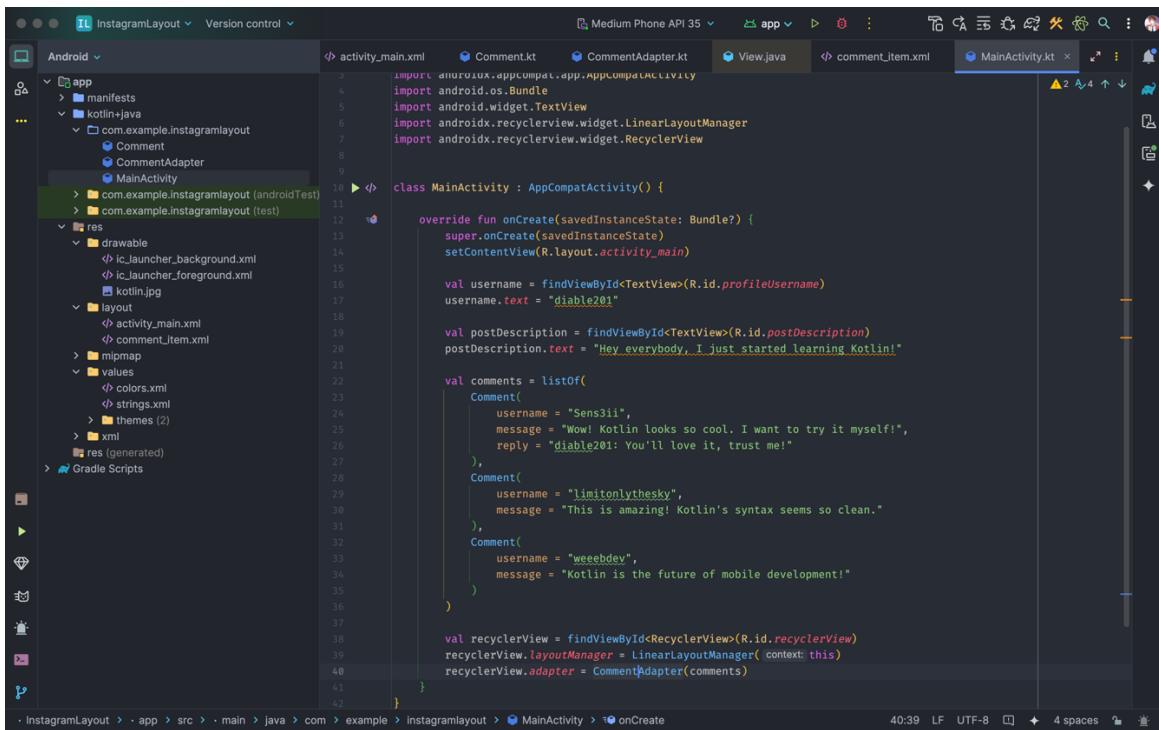
    <TextView
        android:id="@+id/message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is a comment"
        app:layout_constraintTop_toBottomOf="@+id/username"
        app:layout_constraintStart_toStartOf="parent"/>

    <TextView
        android:id="@+id/reply"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Reply goes here"
        android:textColor="#666"
        android:paddingStart="16dp"
        android:paddingEnd="16dp"
        android:visibility="gone"
        app:layout_constraintTop_toBottomOf="@+id/message"
        app:layout_constraintStart_toStartOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```



MainActivity Setup:

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



The screenshot shows the Android Studio interface with the code editor open to `MainActivity.kt`. The code initializes a RecyclerView and populates it with sample data. The data includes comments from users like Sens3li, diable201, limitonlythesky, and weeebdev.

```
import android.os.Bundle
import android.widget.TextView
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView

class MainActivity : AppCompatActivity() {

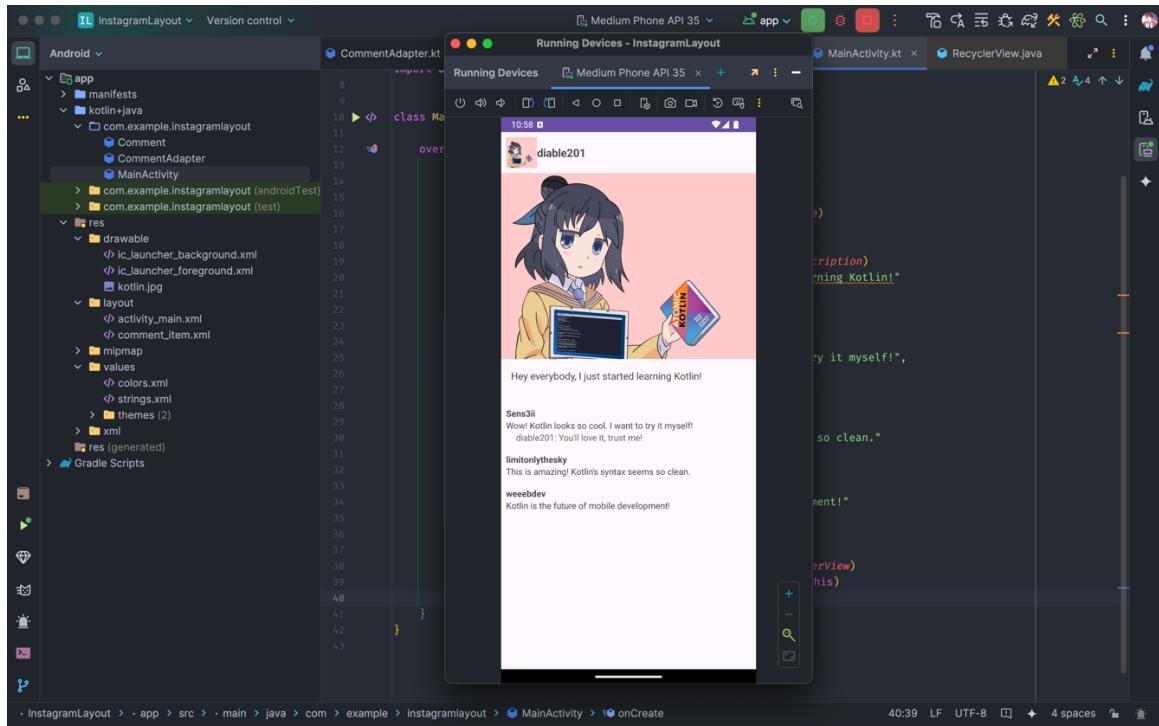
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val username = findViewById<TextView>(R.id.profileUsername)
        username.text = "diable201"

        val postDescription = findViewById<TextView>(R.id.postDescription)
        postDescription.text = "Hey everybody, I just started learning Kotlin!"

        val comments = listOf(
            Comment(
                username = "Sens3li",
                message = "Wow! Kotlin looks so cool. I want to try it myself!",
                reply = "diable201: You'll love it, trust me!"
            ),
            Comment(
                username = "limitonlythesky",
                message = "This is amazing! Kotlin's syntax seems so clean."
            ),
            Comment(
                username = "weeebdev",
                message = "Kotlin is the future of mobile development!"
            )
        )

        val recyclerView = findViewById<RecyclerView>(R.id.recyclerView)
        recyclerView.layoutManager = LinearLayoutManager(context)
        recyclerView.adapter = CommentAdapter(comments)
    }
}
```



The screenshot shows the Android Studio Build Output window. The title bar includes tabs for Build, Sync, Build Output (which is selected), and Build Analyzer. A message at the top indicates a successful build: "Build InstagramLayout: finished At 27.09.2024, 221 sec, 786 ms" with a "Download Info" link. The main pane displays a list of executed tasks for a debug build:

```
Executing tasks: [:app:assembleDebug] in project /Users/sanzhar/AndroidStudioProjects/InstagramLayout
> Task :app:preBuild UP-TO-DATE
> Task :app:preDebugBuild UP-TO-DATE
> Task :app:mergeDebugNativeDebugMetadata NO-SOURCE
> Task :app:checkDebugAarMetadata UP-TO-DATE
> Task :app:generateDebugResValues UP-TO-DATE
> Task :app:mapDebugSourceSetPaths
> Task :app:generateDebugResources
> Task :app:packageDebugResources
> Task :app:mergeDebugResources
> Task :app:createDebugCompatibleScreenManifests UP-TO-DATE
> Task :app:extractDeepLinksDebug UP-TO-DATE
> Task :app:processDebugMainManifest UP-TO-DATE
> Task :app:processDebugManifest UP-TO-DATE
> Task :app:parseDebugLocalResources
> Task :app:processDebugManifestForPackage UP-TO-DATE
> Task :app:javaPreCompileDebug UP-TO-DATE
> Task :app:mergeDebugShaders UP-TO-DATE
> Task :app:compileDebugShaders NO-SOURCE
```

The bottom status bar shows the path "- InstagramLayout > - app > src > - main > java > com > example > instagramlayout > MainActivity > onCreate on onCreate", the time "40:39", and other settings like LF, UTF-8, and code style.