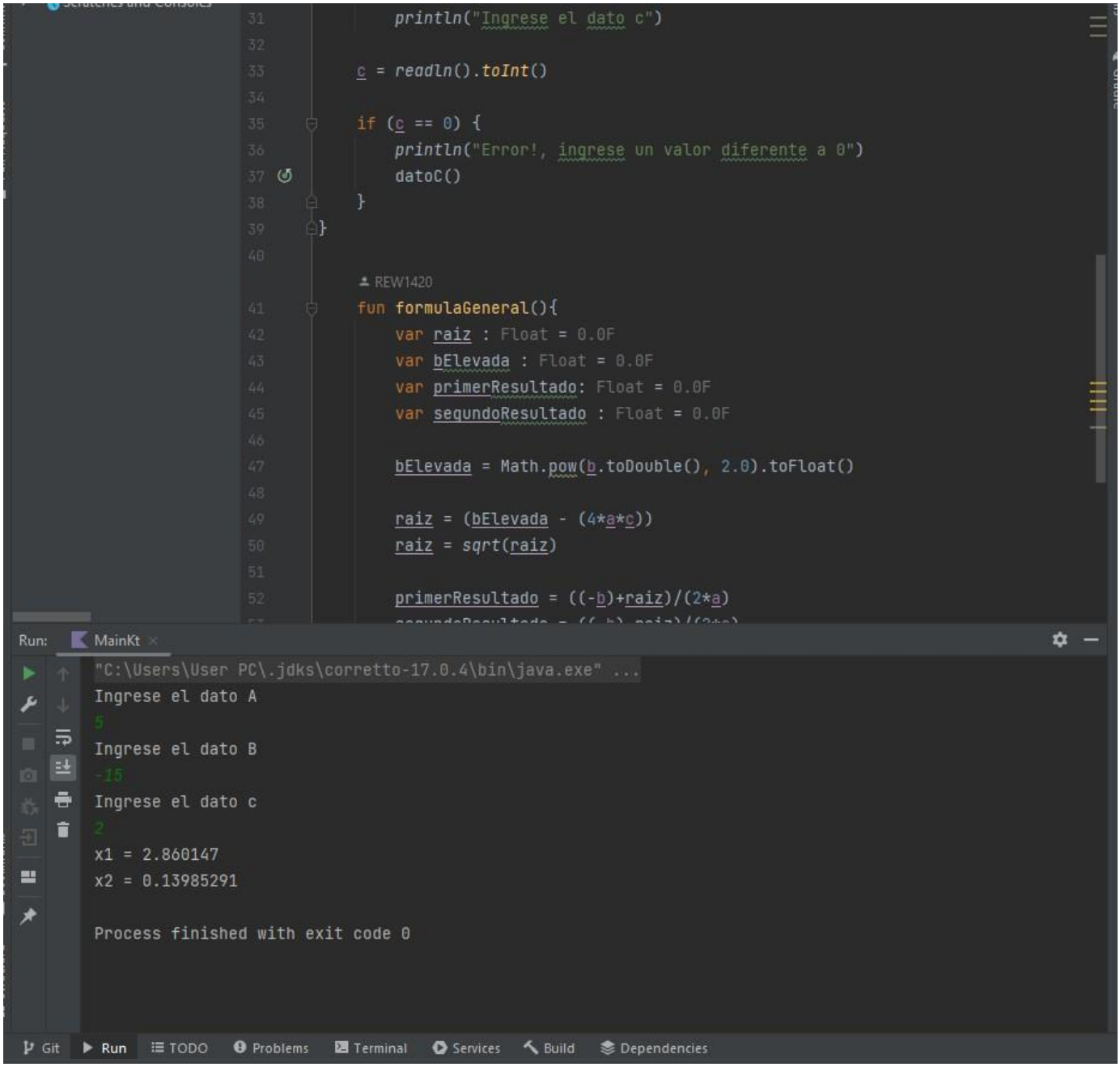
UNIVERSIDAD DON BOSCO

Facultad de Ingeniería
Aplicaciones Móviles

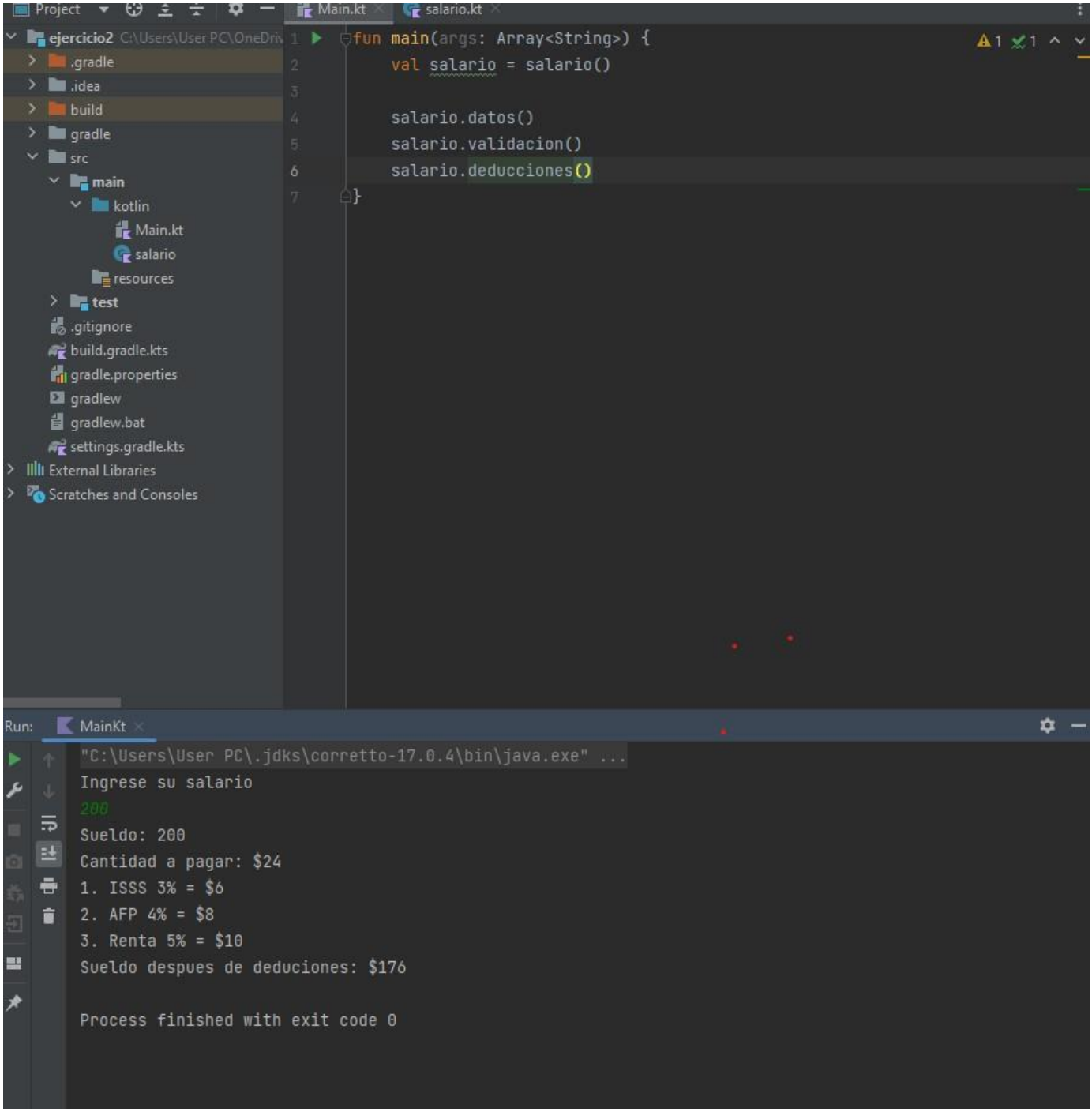
Taller 1

Nombre	Carnet
Javier Eliseo Gutiérrez Flores	GF220089

Ejercicio 1 (AVANCE 100%)
Utilice 3 variables las a,b,c para trabajar y utilice ciclos lamba para las condiciones de que no fueran 0, donde si nula validación se cumple el ciclo regresa a la petición de datos hasta que sea lo que queremos, la formula solo se sustituyó por operaciones matemáticas y se imprime el resultado

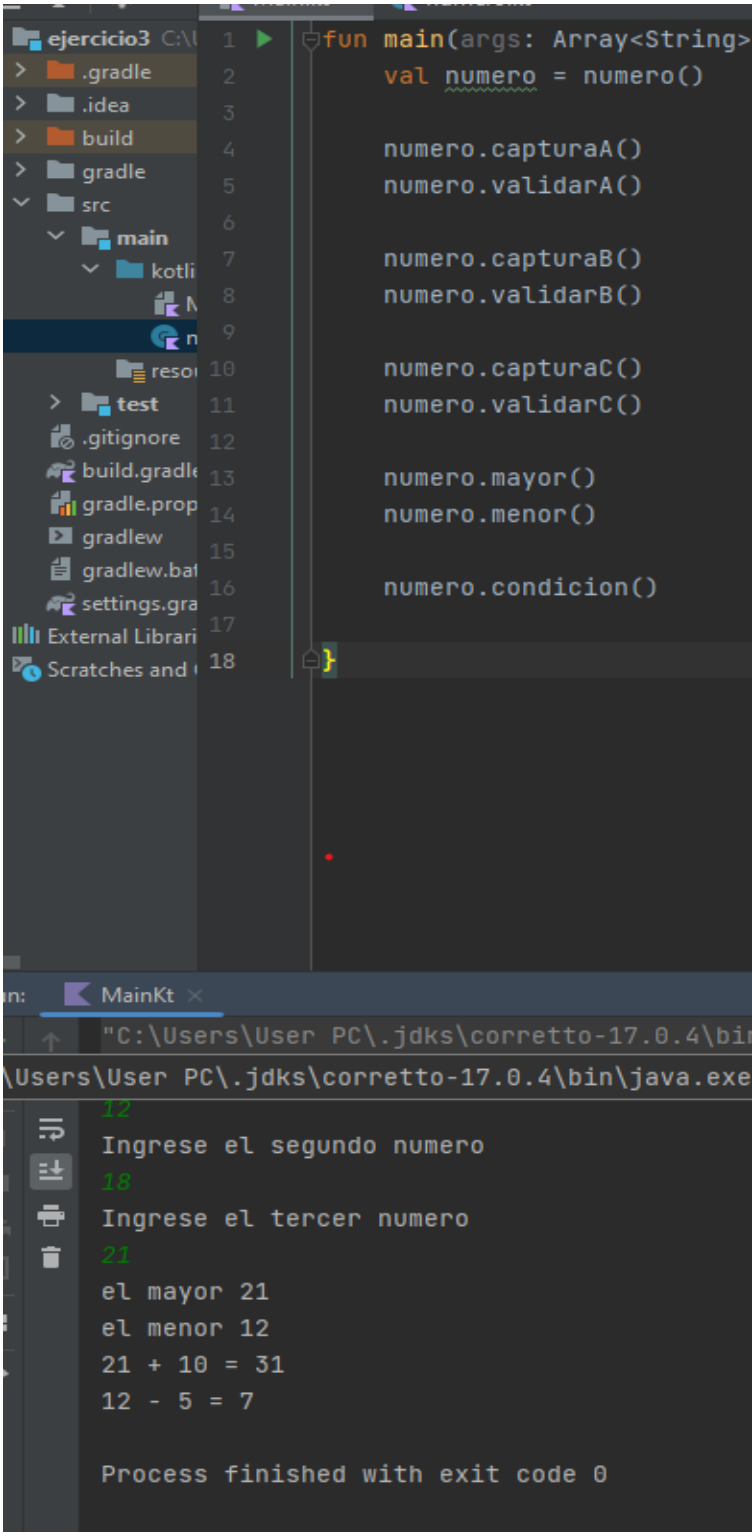

println("Ingrese el dato c")
c = readln().toInt()
if (c == 0) {
println("Error!, ingrese un valor diferente a 0")
datoC()
}
}
REW1420
fun formulaGeneral(){
var raiz : Float = 0.0F
var bElevada : Float = 0.0F
var primerResultado: Float = 0.0F
var segundoResultado : Float = 0.0F
bElevada = Math.pow(b.toDouble(), 2.0).toFloat()
raiz = (bElevada - (4*a*c))
raiz = sqrt(raiz)
primerResultado = ((-b)+raiz)/(2*a)
Run: MainKt
"C:\Users\User PC\.jdks\corretto-17.0.4\bin\java.exe" ...
Ingrese el dato A
5
Ingrese el dato B
-15
Ingrese el dato c
2
x1 = 2.860147
x2 = 0.13985291
Process finished with exit code 0
Git
Run
TODO
Problems
Terminal
Services
Build
Dependencies

Ejercicio 2 (AVANCE 100%)
Este es sencillo solo se toma el valor del salario y con ciclo lambda se condiciona que no sea 0 ni negativo, se aplica la fórmula para calcular porcentajes se suma las deducciones y se le resta al salario neto.


Project
ejercicio2 C:\Users\User PC\OneDriv
.gradle
.idea
build
gradle
src
main
kotlin
Main.kt
salario
resources
test
.gitignore
build.gradle.kts
gradle.properties
gradlew
gradlew.bat
settings.gradle.kts
External Libraries
Scratches and Consoles
Main.kt
salario.kt
fun main(args: Array<String>) {
val salario = salario()
salario.datos()
salario.validacion()
salario.deducciones()
}
Run:
MainKt
"C:\Users\User PC\.jdks\corretto-17.0.4\bin\java.exe" ...
Ingrese su salario
200
Sueldo: 200
Cantidad a pagar: $24
1. ISSS 3% = $6
2. AFP 4% = $8
3. Renta 5% = $10
Sueldo despues de deduciones: $176
Process finished with exit code 0

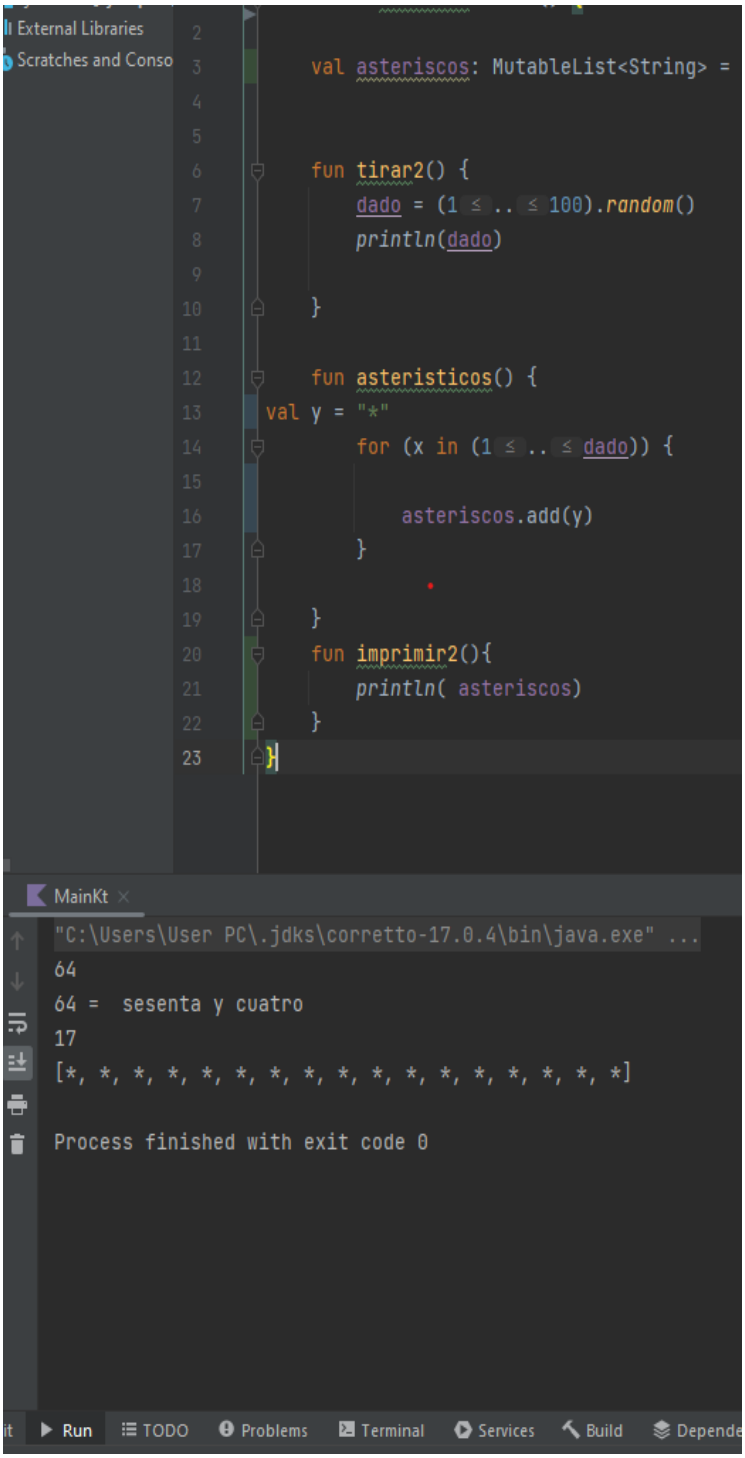
Ejercicio 3 (AVANCE 90%)

Es más complejo ya que utiliza varias condiciones, para todas utilice lambda ya que es más fácil hacer regresar al condigo si la condición no cumple, valide negativo, cero, iguales, y primos de 3 excluyendo al 9 por el tipo de ciclo lamba ocurren detalles en esta última condición, pero es funcional todo y se aplican las condiciones al menor y mayor


ejercicio3
.gradle
.idea
build
gradle
src
main
test
.gitignore
gradlew
External Librari
Scratches and
fun main(args: Array<String>
val numero = numero()
numero.capturaA()
numero.validarA()
numero.capturaB()
numero.validarB()
numero.capturaC()
numero.validarC()
numero.mayor()
numero.menor()
numero.condicion()
MainKt
"C:\Users\User PC\.jdks\corretto-17.0.4\bi
\Users\User PC\.jdks\corretto-17.0.4\bin\java.exe
12
Ingrese el segundo numero
18
Ingrese el tercer numero
21
el mayor 21
el menor 12
21 + 10 = 31
12 - 5 = 7
Process finished with exit code 0

Ejercicio 4 (AVANCE 90%)
Ya que la única manera de hacer que se escriban los nombres de los números en letras fue con if anidados lo realice así y los condicione a que se ejecute de esa manera, luego con el ciclo for se imprimen los asteriscos dependiendo del tamaño que se haya dado en la función random.

Para evitar que se impriman desordenadamente utilice listas mutables para que se agreguen los asteriscos en la lista por un ciclo for que este va de 1 a lo que el dado decida


External Libraries
Scratches and Conso
val asteriscos: MutableList<String> =
fun tirar2() {
dado = (1 ≤ .. ≤ 100).random()
println(dado)
}
fun asteristicos() {
val y = "*"
for (x in (1 ≤ .. ≤ dado)) {
asteriscos.add(y)
}
}
fun imprimir2(){
println(asteriscos)
}
}
MainKt
"C:\Users\User PC\.jdks\corretto-17.0.4\bin\java.exe" ...
64
64 = sesenta y cuatro
17
[*, *, *, *, *, *, *, *, *, *, *, *, *, *, *, *, *]
Process finished with exit code 0
Run
TODO
Problems
Terminal
Services
Build