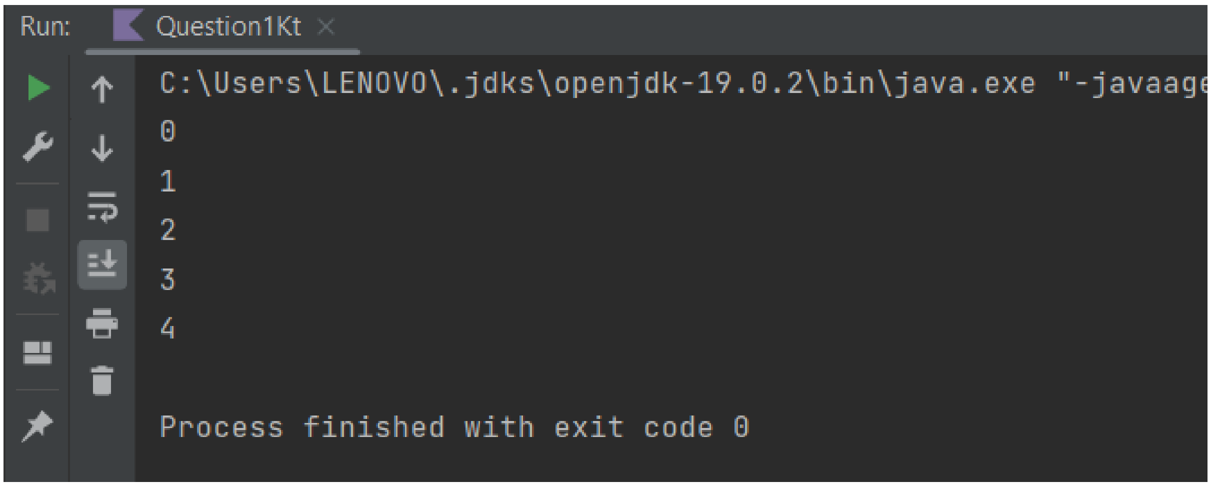
1. Write a program to execute while loop in kotlin .

Program :

```
fun main(){
    var i = 0
    while (i < 5) {
        println(i)
        i++
    }
}
```

Output :



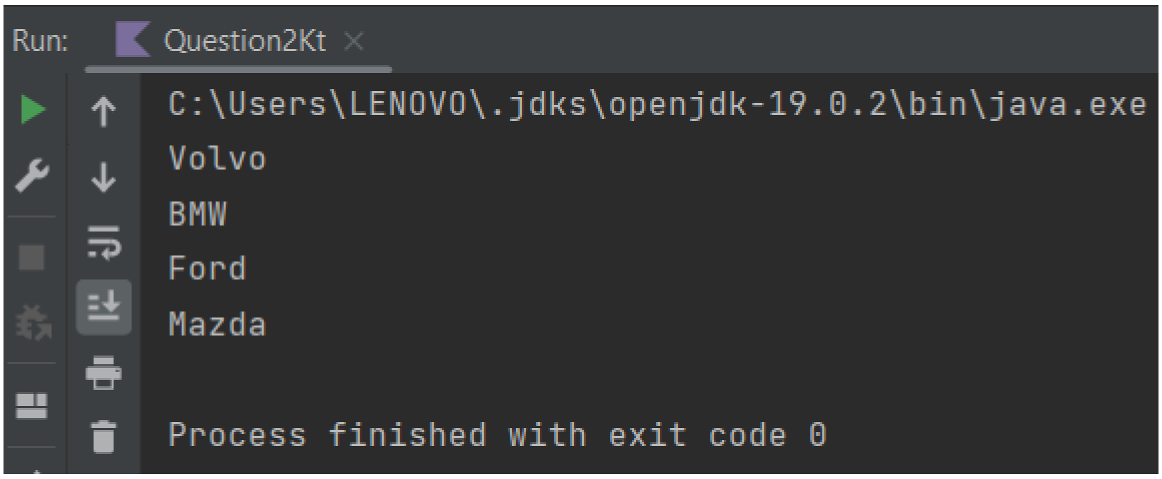
```
C:\Users\LENOVO\.jdks\openjdk-19.0.2\bin\java.exe "-javaage
0
1
2
3
4

Process finished with exit code 0
```

2. Write a program to execute for loop in kotlin.

Program :

```
fun main(){
    val cars = arrayOf("Volvo", "BMW", "Ford", "Mazda")
    for (x in cars) {
        println(x)
    }
}
```

Output :



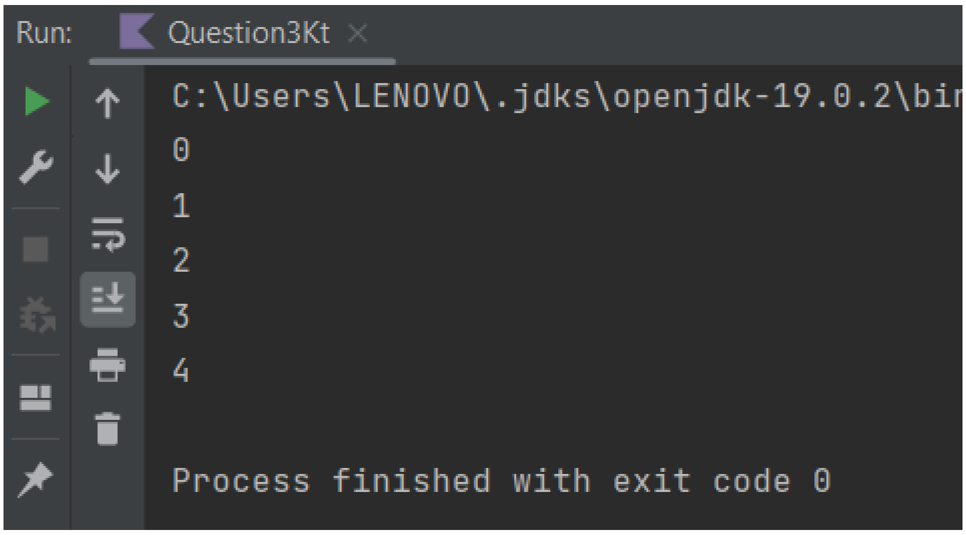
```
Run:  Question2Kt
C:\Users\LENOVO\.jdks\openjdk-19.0.2\bin\java.exe
Volvo
BMW
Ford
Mazda

Process finished with exit code 0
```

3. Write a program to execute do while loop in kotlin.

Program :

```
fun main(){
   var i = 0
   do {
      println(i)
      i++
   }
   while (i < 5)
}
```

Output :



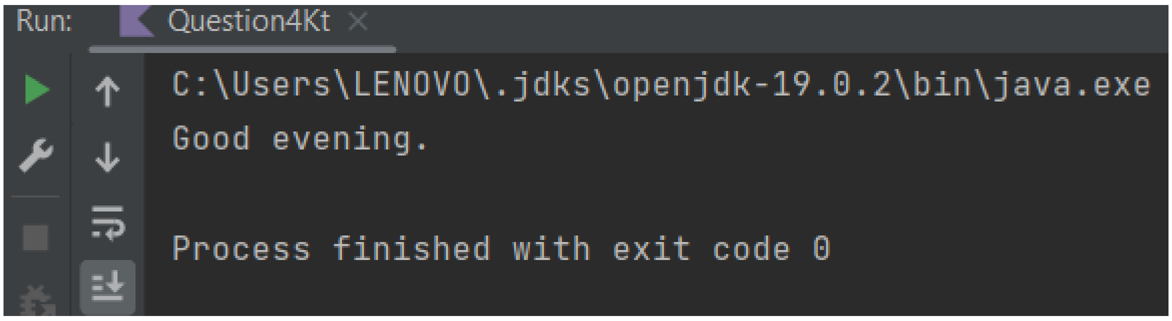
```
Run:  Question3Kt
C:\Users\LENOVO\.jdks\openjdk-19.0.2\bi
0
1
2
3
4

Process finished with exit code 0
```

4. Write a program to execute conditional statement (if-else) in kotlin.

Program :

```
fun main(){
   val time = 22
   if (time < 10) {
      println("Good morning.")
   } else if (time < 20) {
      println("Good day.")
   } else {
      println("Good evening.")
   }
}
```

Output :



```
C:\Users\LENOVO\.jdks\openjdk-19.0.2\bin\java.exe
Good evening.

Process finished with exit code 0
```

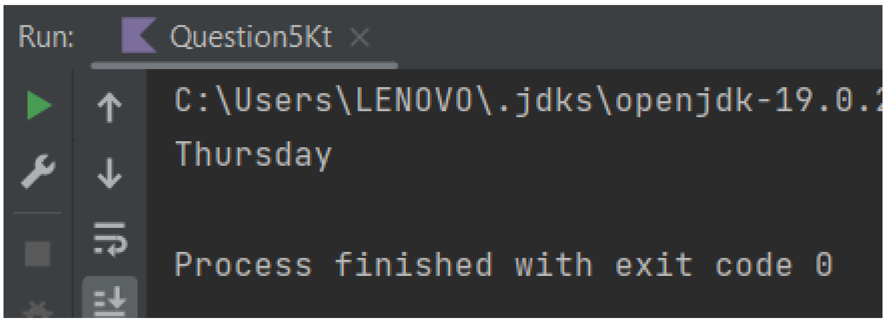
5. Write a program to execute conditional statement (when) in kotlin.

Program :

```
fun main(){
   val day = 4
   val result = when (day) {
      1 -> "Monday"
      2 -> "Tuesday"
      3 -> "Wednesday"
      4 -> "Thursday"
      5 -> "Friday"
```

```
            6 -> "Saturday"
            7 -> "Sunday"
            else -> "Invalid day."
        }
        println(result)
    }
```

Output :



```
C:\Users\LENOVO\.jdks\openjdk-19.0.
Thursday

Process finished with exit code 0
```

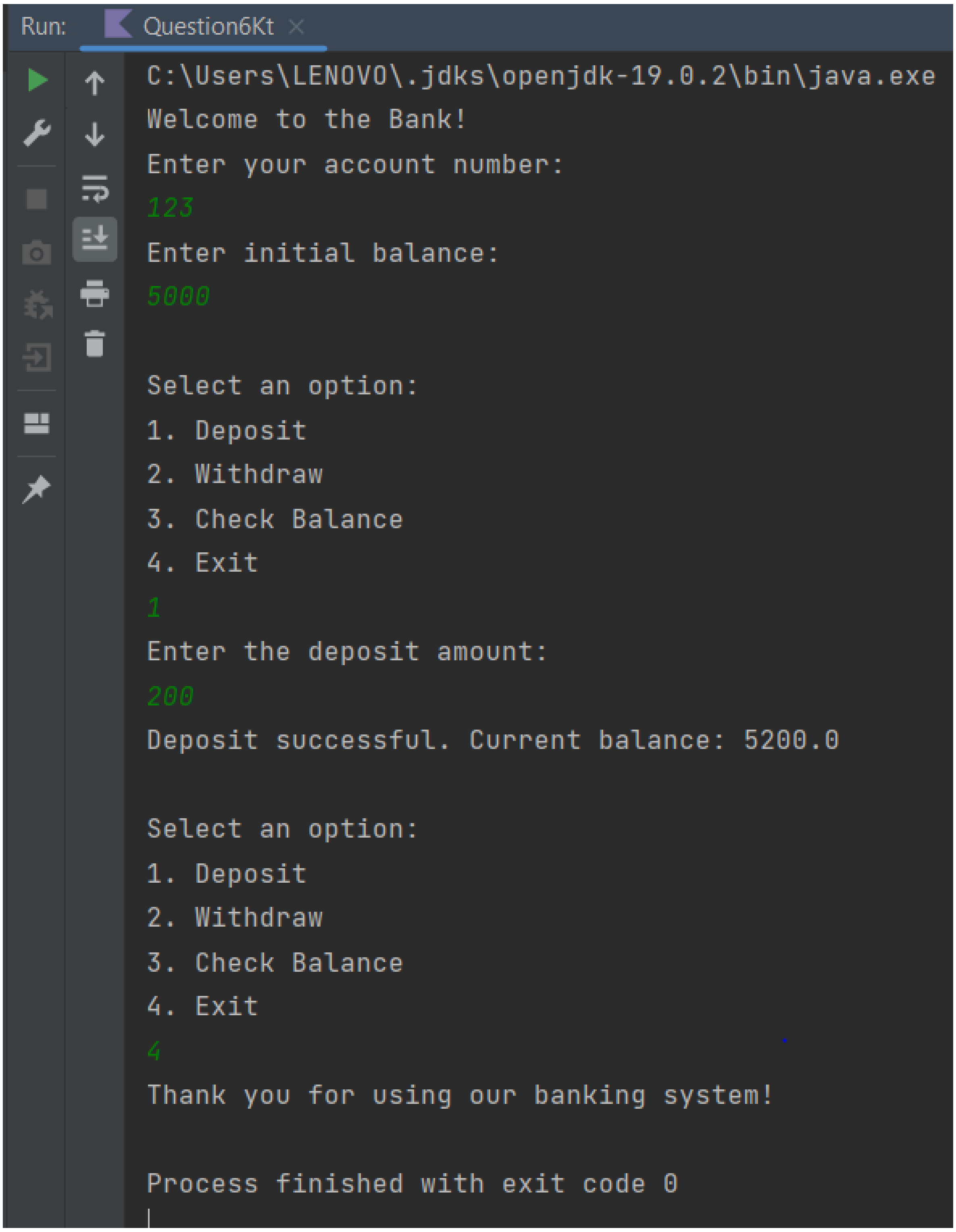
6. Write a kotlin program to performing banking operations for deposite, Withdraw and check bank balance

Program :

```
import java.util.Scanner
class BankAccount(private val accountNumber: String, private var
balance: Double) {
    fun deposit(amount: Double) {
        balance += amount
        println("Deposit successful. Current balance: $balance")
    }
    fun withdraw(amount: Double) {
        if (balance >= amount) {
            balance -= amount
            println("Withdrawal successful. Current balance: $balance")
        } else {
            println("Insufficient funds. Current balance: $balance")
        }
    }
    fun checkBalance() {
        println("Current balance: $balance")
    }
```

```kotlin
}
fun main() {
    val scanner = Scanner(System.`in`)
    println("Welcome to the Bank!")
    // Create a bank account
    println("Enter your account number:")
    val accountNumber = scanner.nextLine()
    println("Enter initial balance:")
    val initialBalance = scanner.nextDouble()
    val bankAccount = BankAccount(accountNumber, initialBalance)
    // Perform banking operations
    var option: Int
    do {
        println("\nSelect an option:")
        println("1. Deposit")
        println("2. Withdraw")
        println("3. Check Balance")
        println("4. Exit")
        option = scanner.nextInt()
        when (option) {
            1 -> {
                println("Enter the deposit amount:")
                val depositAmount = scanner.nextDouble()
                bankAccount.deposit(depositAmount)
            }
            2 -> {
                println("Enter the withdrawal amount:")
                val withdrawalAmount = scanner.nextDouble()
                bankAccount.withdraw(withdrawalAmount)
            }
            3 -> bankAccount.checkBalance()
            4 -> println("Thank you for using our banking system!")
            else -> println("Invalid option. Please try again.")
        }
    } while (option != 4)
    scanner.close()
}
```

Output :



Run: Question6Kt

```
C:\Users\LENOVO\.jdks\openjdk-19.0.2\bin\java.exe
Welcome to the Bank!
Enter your account number:
123
Enter initial balance:
5000

Select an option:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
1
Enter the deposit amount:
200
Deposit successful. Current balance: 5200.0

Select an option:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
4
Thank you for using our banking system!

Process finished with exit code 0
```

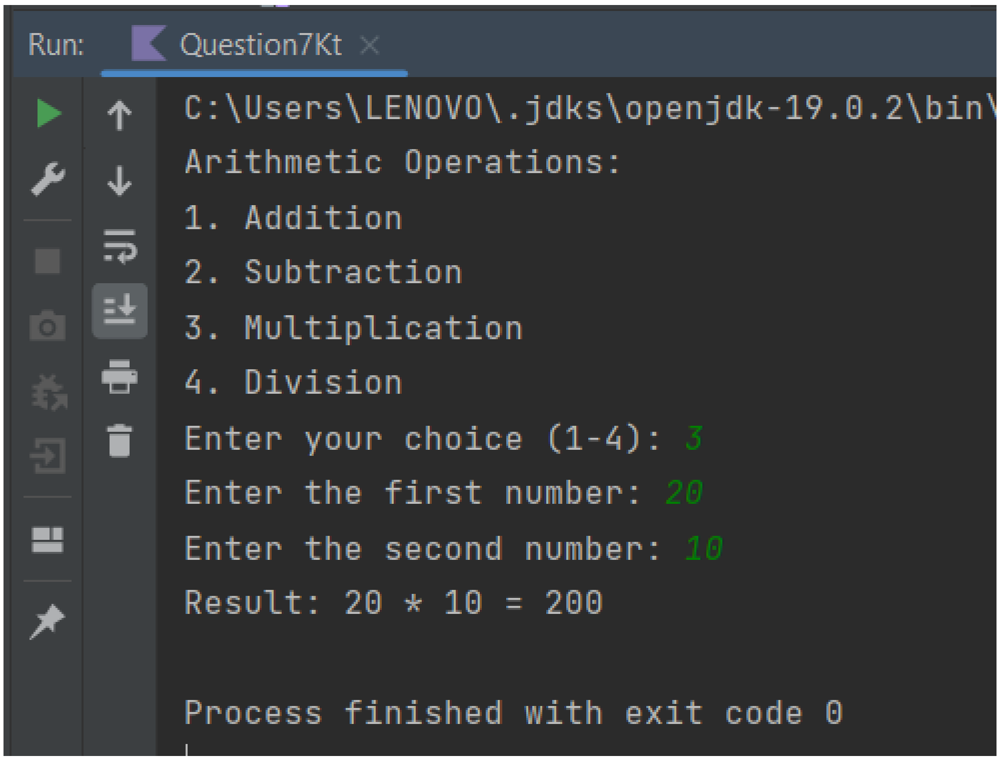
7. Write a program to perform arithmetic operation using lambda function

Program :

```
val add: (Int, Int) -> Int = { x, y -> x + y }
val subtract: (Int, Int) -> Int = { x, y -> x - y }
val multiply: (Int, Int) -> Int = { x, y -> x * y }
val divide: (Int, Int) -> Double = { x, y -> x.toDouble() / y }
fun main() {
   println("Arithmetic Operations:")
   println("1. Addition")
   println("2. Subtraction")
   println("3. Multiplication")
   println("4. Division")
   print("Enter your choice (1-4): ")
   val choice = readLine()?.toIntOrNull()
   if (choice !in 1..4) {
      println("Invalid choice!")
      return
   }
   print("Enter the first number: ")
   val num1 = readLine()?.toIntOrNull()
   if (num1 == null) {
      println("Invalid number!")
      return
   }
   print("Enter the second number: ")
   val num2 = readLine()?.toIntOrNull()
   if (num2 == null) {
      println("Invalid number!")
      return
   }
   val result = when (choice) {
      1 -> add(num1, num2)
      2 -> subtract(num1, num2)
      3 -> multiply(num1, num2)
      4 -> divide(num1, num2)
```

```
        else -> throw IllegalArgumentException("Invalid choice!")
    }
    val operator = when (choice) {
        1 -> "+"
        2 -> "-"
        3 -> "*"
        4 -> "/"
        else -> throw IllegalArgumentException("Invalid choice!")
    }
    println("Result: $num1 $operator $num2 = $result")
}
```

Output :



Run: Question7Kt

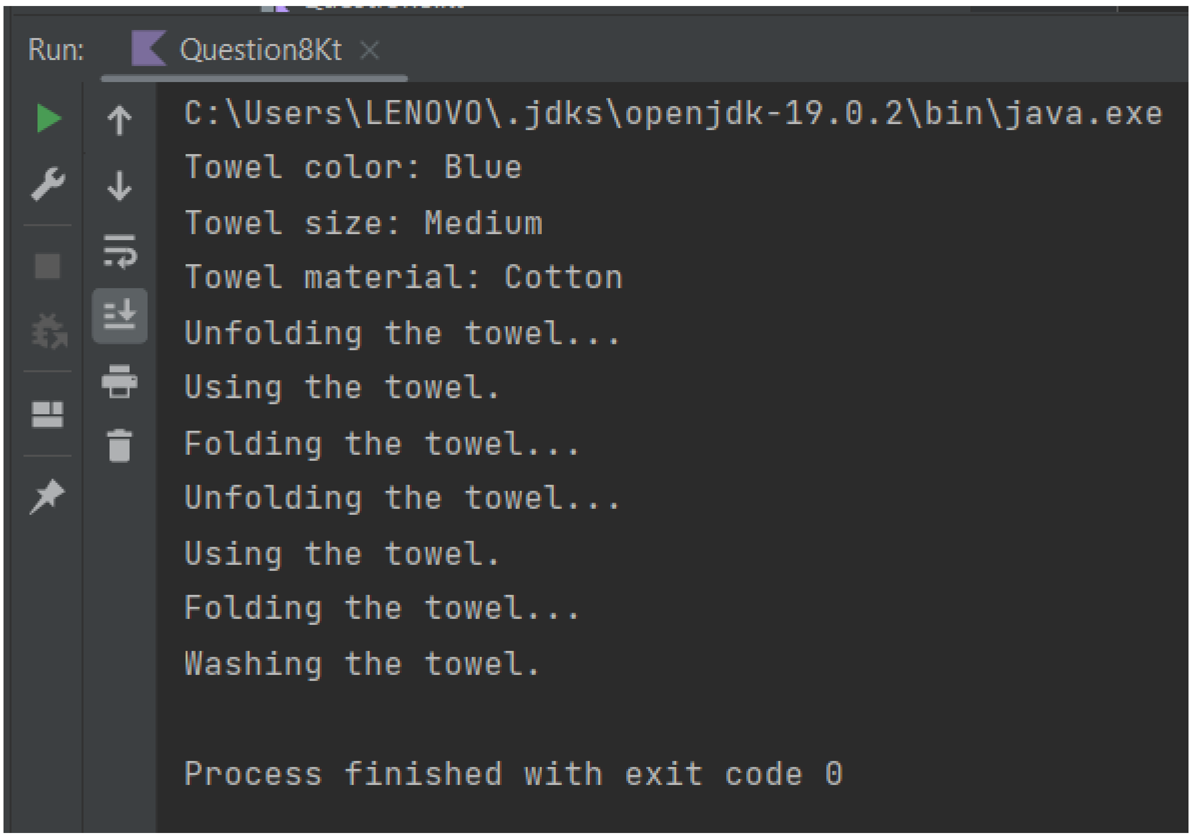
```
C:\Users\LENOVO\.jdks\openjdk-19.0.2\bin\
Arithmetic Operations:
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your choice (1-4): 3
Enter the first number: 20
Enter the second number: 10
Result: 20 * 10 = 200

Process finished with exit code 0
```

8. **In the ‘Towel’ class ,what are the properties and functions available to work with towels?**

Program :

```
class Towel(val color: String, val size: String, val material: String) {
    var isFolded: Boolean = true
        private set

    fun fold() {
        if (!isFolded) {
            println("Folding the towel...")
            isFolded = true
        } else {
            println("The towel is already folded.")
        }
    }
    fun unfold() {
        if (isFolded) {
            println("Unfolding the towel...")
            isFolded = false
        } else {
            println("The towel is already unfolded.")
        }
    }
    fun use() {
        if (isFolded) {
            unfold()
        }
        println("Using the towel.")
    }
    fun wash() {
        if (!isFolded) {
            fold()
        }
        println("Washing the towel.")
    }
    // Additional properties and functions can be added as per the
requirements

}
fun main() {
    val towel = Towel("Blue", "Medium", "Cotton")
    println("Towel color: ${towel.color}")
    println("Towel size: ${towel.size}")
    println("Towel material: ${towel.material}")

    towel.use()
    towel.fold()
    towel.use()
    towel.wash()
}
```

Output :



```
C:\Users\LENOVO\.jdks\openjdk-19.0.2\bin\java.exe
Towel color: Blue
Towel size: Medium
Towel material: Cotton
Unfolding the towel...
Using the towel.
Folding the towel...
Unfolding the towel...
Using the towel.
Folding the towel...
Washing the towel.

Process finished with exit code 0
```

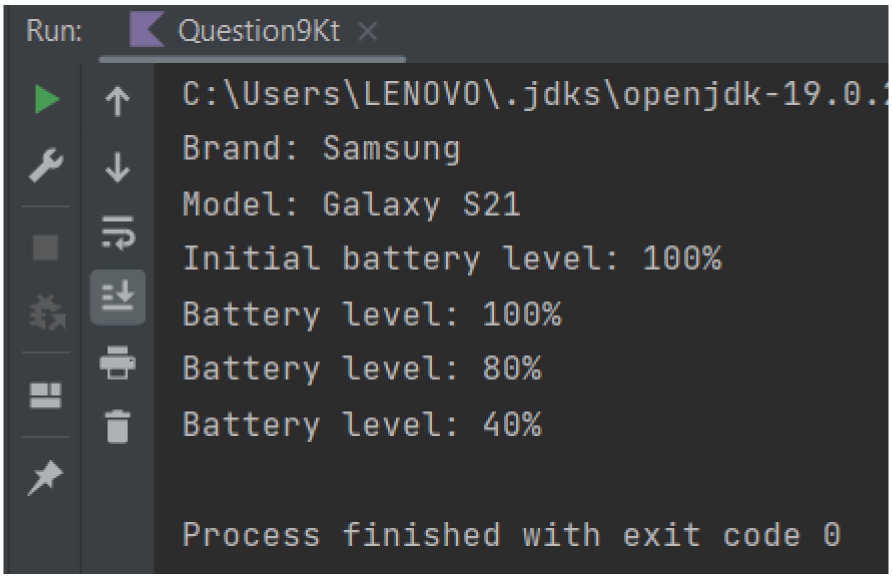
9. How dose the ‘Mobile’ class handle making calls and managning the battery level

Program :

```
class Mobile(val brand: String, val model: String) {
   var batteryLevel: Int = 100
      private set
   fun chargeBattery(chargeAmount: Int) {
      if (chargeAmount < 0) {
         throw IllegalArgumentException("Charge amount cannot be
negative.")
      }
      batteryLevel += chargeAmount
      if (batteryLevel > 100) {
         batteryLevel = 100
      }
      println("Battery level: $batteryLevel%")
   }
   fun useBattery(drainAmount: Int) {
      if (drainAmount < 0) {
         throw IllegalArgumentException("Drain amount cannot be
negative.")
      }
      batteryLevel -= drainAmount
      if (batteryLevel < 0) {
         batteryLevel = 0
      }
      println("Battery level: $batteryLevel%")
   }
}
fun main() {
   val myMobile = Mobile("Samsung", "Galaxy S21")
   println("Brand: ${myMobile.brand}")
   println("Model: ${myMobile.model}")
   println("Initial battery level: ${myMobile.batteryLevel}%")
   myMobile.chargeBattery(50)
   myMobile.useBattery(20)
```

```
        myMobile.useBattery(40)
    }
```

Output :



```
C:\Users\LENOVO\.jdks\openjdk-19.0.
Brand: Samsung
Model: Galaxy S21
Initial battery level: 100%
Battery level: 100%
Battery level: 80%
Battery level: 40%

Process finished with exit code 0
```