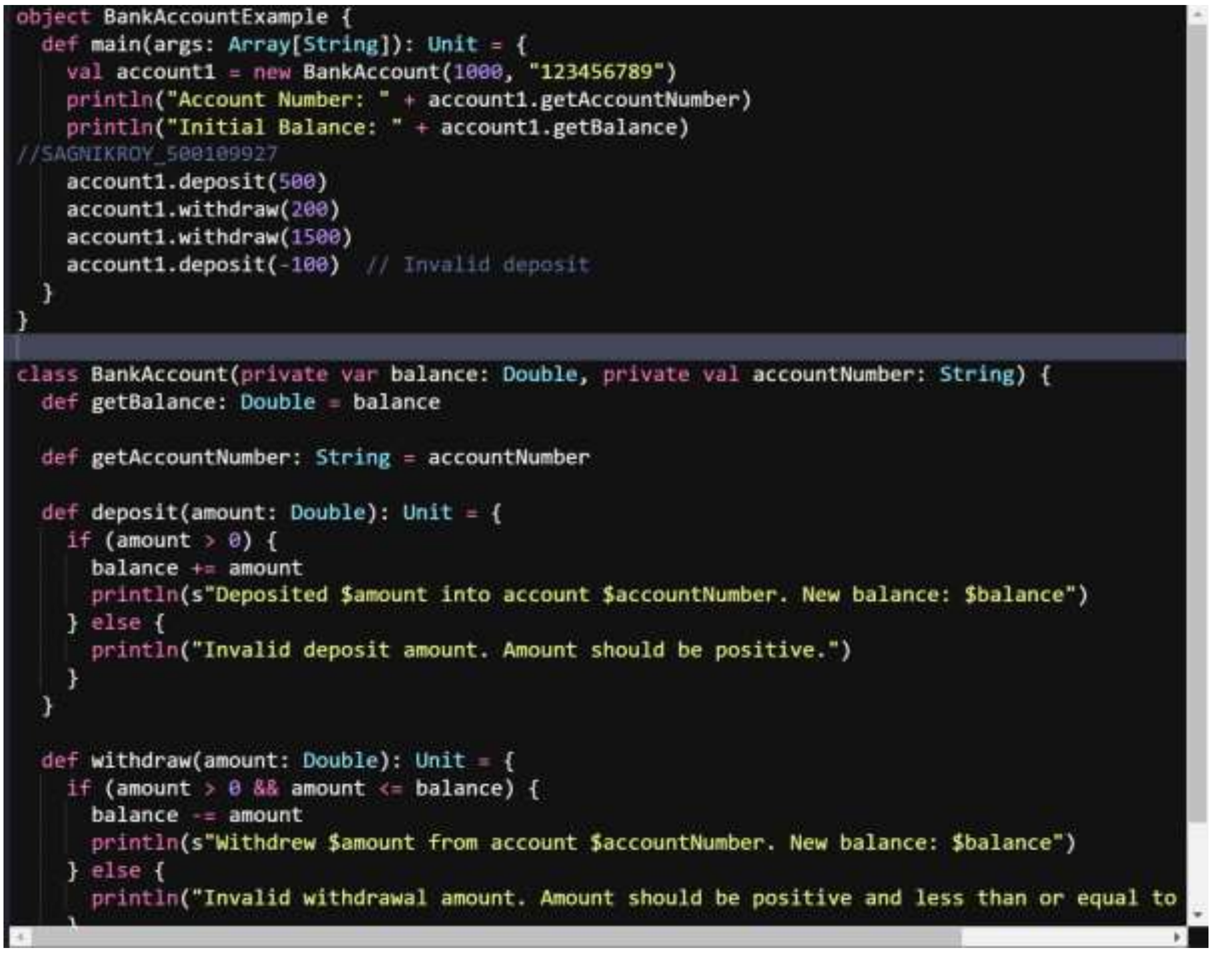
EXPERIMENT-6

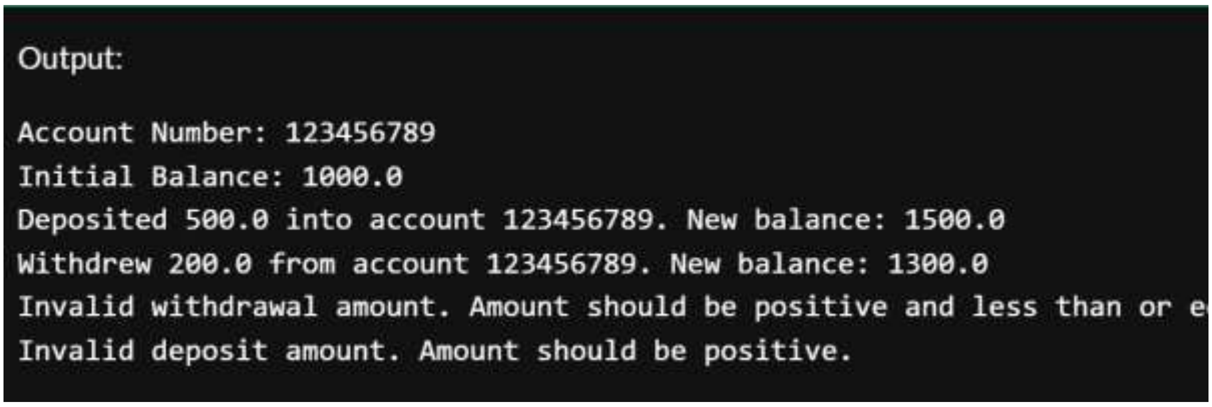
Q1. Define a class BankAccount with private fields balance and accountNumber. Provide methods to access these private fields. Also, include methods deposit and withdraw to modify the balance. Create object the class and illustrate use of the created methods.

Solution:



```scala
object BankAccountExample {
  def main(args: Array[String]): Unit = {
    val account1 = new BankAccount(1000, "123456789")
    println("Account Number: " + account1.getAccountNumber)
    println("Initial Balance: " + account1.getBalance)
//SAGNIKROY_500109927
    account1.deposit(500)
    account1.withdraw(200)
    account1.withdraw(1500)
    account1.deposit(-100)  // Invalid deposit
  }
}

class BankAccount(private var balance: Double, private val accountNumber: String) {
  def getBalance: Double = balance

  def getAccountNumber: String = accountNumber

  def deposit(amount: Double): Unit = {
    if (amount > 0) {
      balance += amount
      println(s"Deposited $amount into account $accountNumber. New balance: $balance")
    } else {
      println("Invalid deposit amount. Amount should be positive.")
    }
  }

  def withdraw(amount: Double): Unit = {
    if (amount > 0 && amount <= balance) {
      balance -= amount
      println(s"Withdrew $amount from account $accountNumber. New balance: $balance")
    } else {
      println("Invalid withdrawal amount. Amount should be positive and less than or equal to
```

Output:

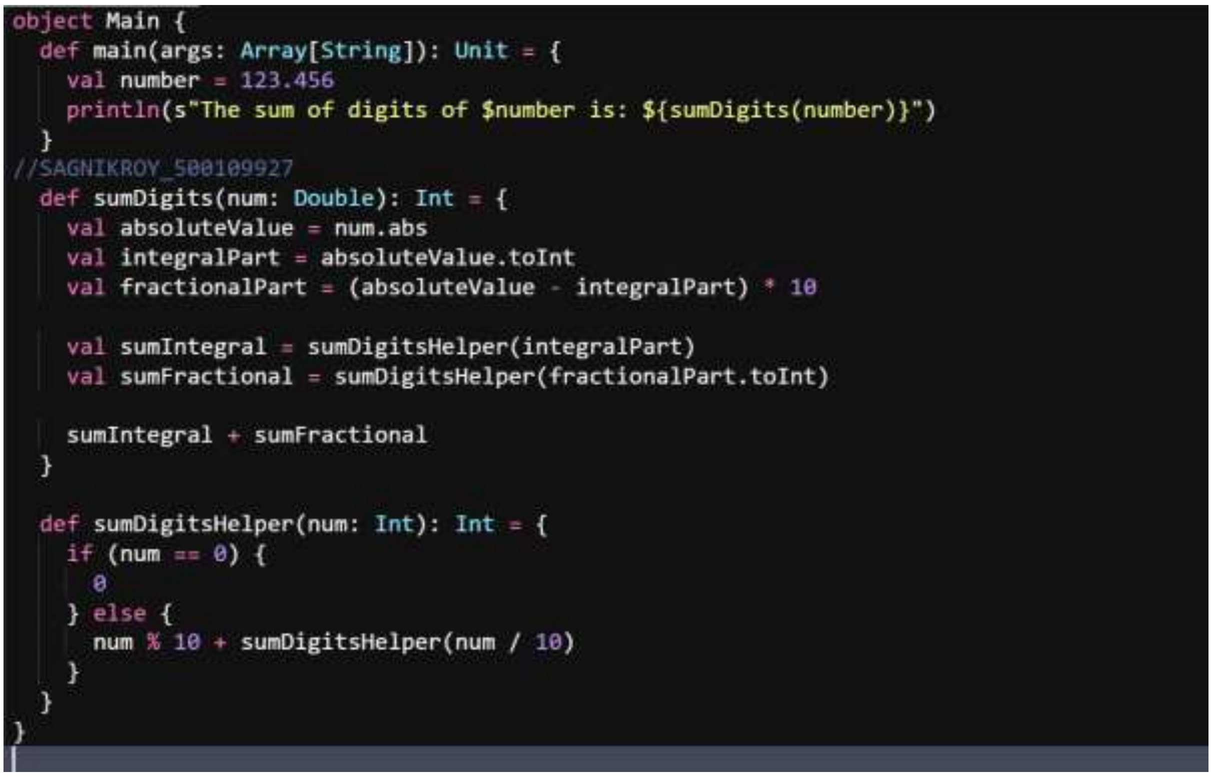


Output:

```
Account Number: 123456789
Initial Balance: 1000.0
Deposited 500.0 into account 123456789. New balance: 1500.0
Withdrew 200.0 from account 123456789. New balance: 1300.0
Invalid withdrawal amount. Amount should be positive and less than or e
Invalid deposit amount. Amount should be positive.
```

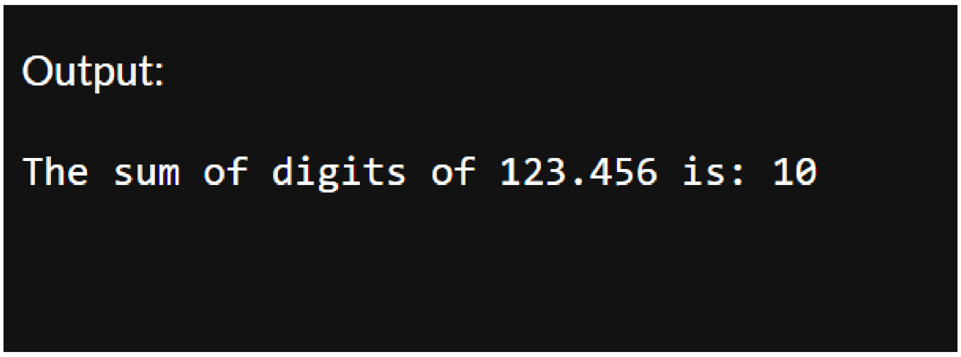
Q2. Define a recursive function sumDigits that takes a double as input and returns the sum of its digits. Write complete scala code to check the function.

Solution:



```
object Main {
  def main(args: Array[String]): Unit = {
    val number = 123.456
    println(s"The sum of digits of $number is: ${sumDigits(number)}")
  }
//SAGNIKROY_500109927
  def sumDigits(num: Double): Int = {
    val absoluteValue = num.abs
    val integralPart = absoluteValue.toInt
    val fractionalPart = (absoluteValue - integralPart) * 10

    val sumIntegral = sumDigitsHelper(integralPart)
    val sumFractional = sumDigitsHelper(fractionalPart.toInt)

    sumIntegral + sumFractional
  }

  def sumDigitsHelper(num: Int): Int = {
    if (num == 0) {
      0
    } else {
      num % 10 + sumDigitsHelper(num / 10)
    }
  }
}
```

Output:

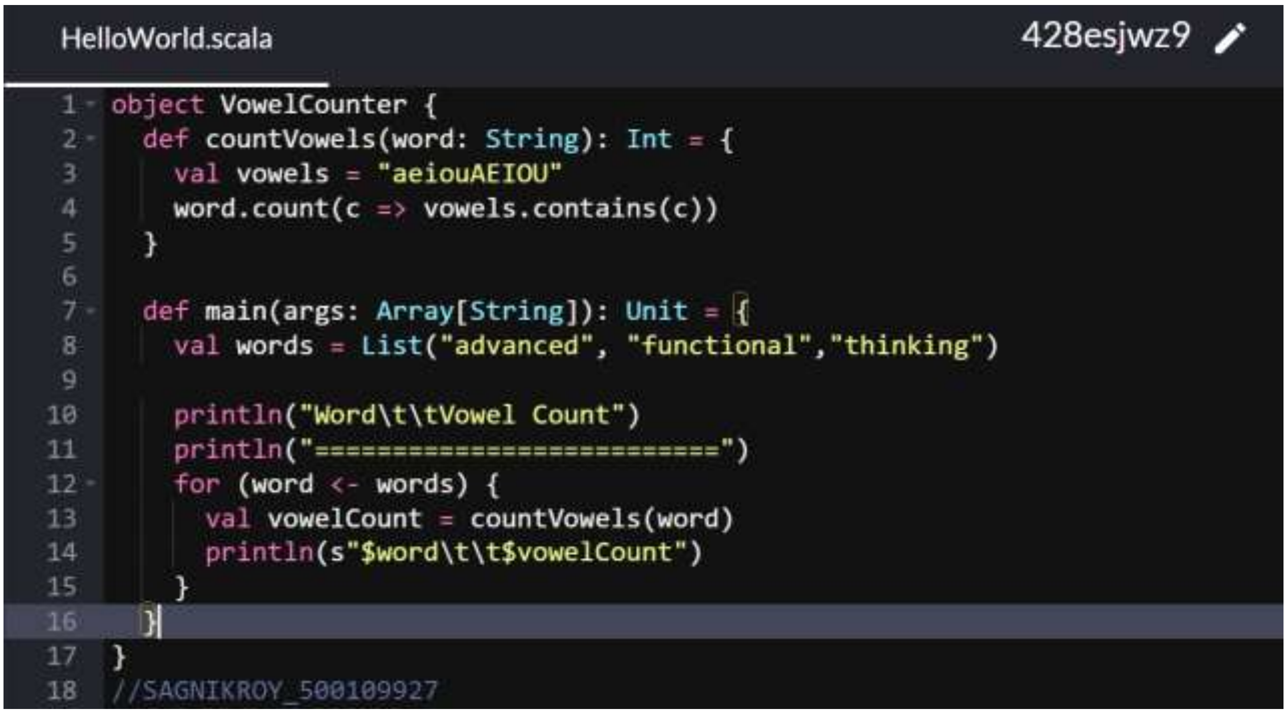


Output:

```
The sum of digits of 123.456 is: 10
```

Q3. Create a list named words containing some strings in Scala. Iterate over the elements of the list words, compute number of vowels in the string using a user defined function and print the results.

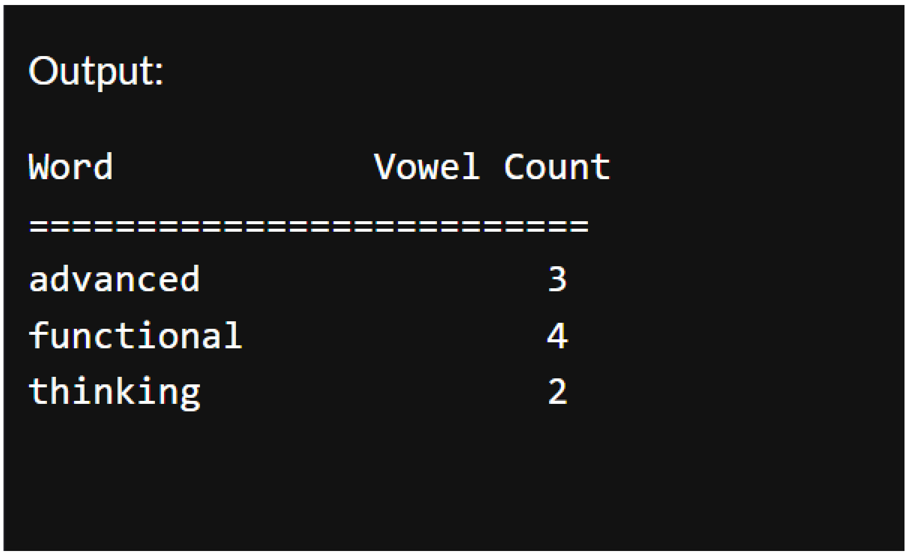
Solution:



HelloWorld.scala 428esjwz9

```scala
object VowelCounter {
  def countVowels(word: String): Int = {
    val vowels = "aeiouAEIOU"
    word.count(c => vowels.contains(c))
  }

  def main(args: Array[String]): Unit = {
    val words = List("advanced", "functional","thinking")

    println("Word\t\tVowel Count")
    println("=========================")
    for (word <- words) {
      val vowelCount = countVowels(word)
      println(s"$word\t\t$vowelCount")
    }
  }
}
//SAGNIKROY_500109927
```

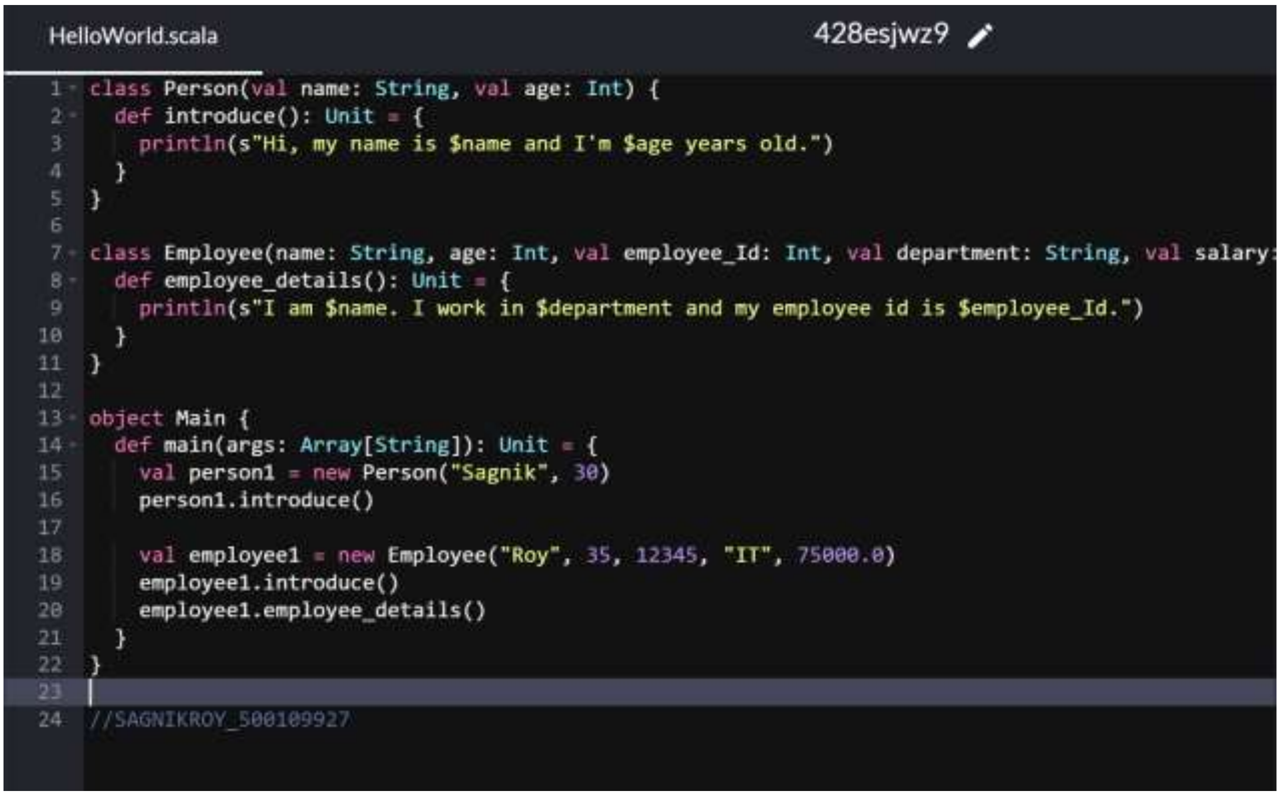
Output:



```
Output:

Word            Vowel Count
==========================
advanced              3
functional            4
thinking              2
```

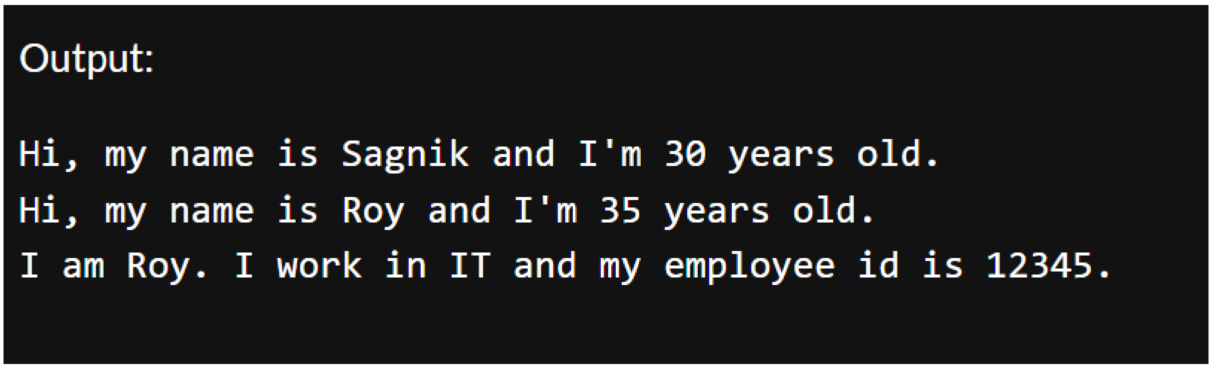
Q4. Define a class Person with attributes names and age. Create a Class Employee that inherits Person class and has additional attributes employee_Id, department, and salary. Person class has a method ‘introduce’ with default arguments that prints "Hi, my name is [name] and I'm [age] years old. Similarly, Employee class has a method ‘employee_details’ with default argumets that prints, “I am [name]. I work in [department] and my employee id is [employee_Id]." Create an instance of both the classes and call the methods created.

Solution:


HelloWorld.scala 428esjwz9

```scala
class Person(val name: String, val age: Int) {
  def introduce(): Unit = {
    println(s"Hi, my name is $name and I'm $age years old.")
  }
}

class Employee(name: String, age: Int, val employee_Id: Int, val department: String, val salary:
  def employee_details(): Unit = {
    println(s"I am $name. I work in $department and my employee id is $employee_Id.")
  }
}

object Main {
  def main(args: Array[String]): Unit = {
    val person1 = new Person("Sagnik", 30)
    person1.introduce()

    val employee1 = new Employee("Roy", 35, 12345, "IT", 75000.0)
    employee1.introduce()
    employee1.employee_details()
  }
}

//SAGNIKROY_500109927
```

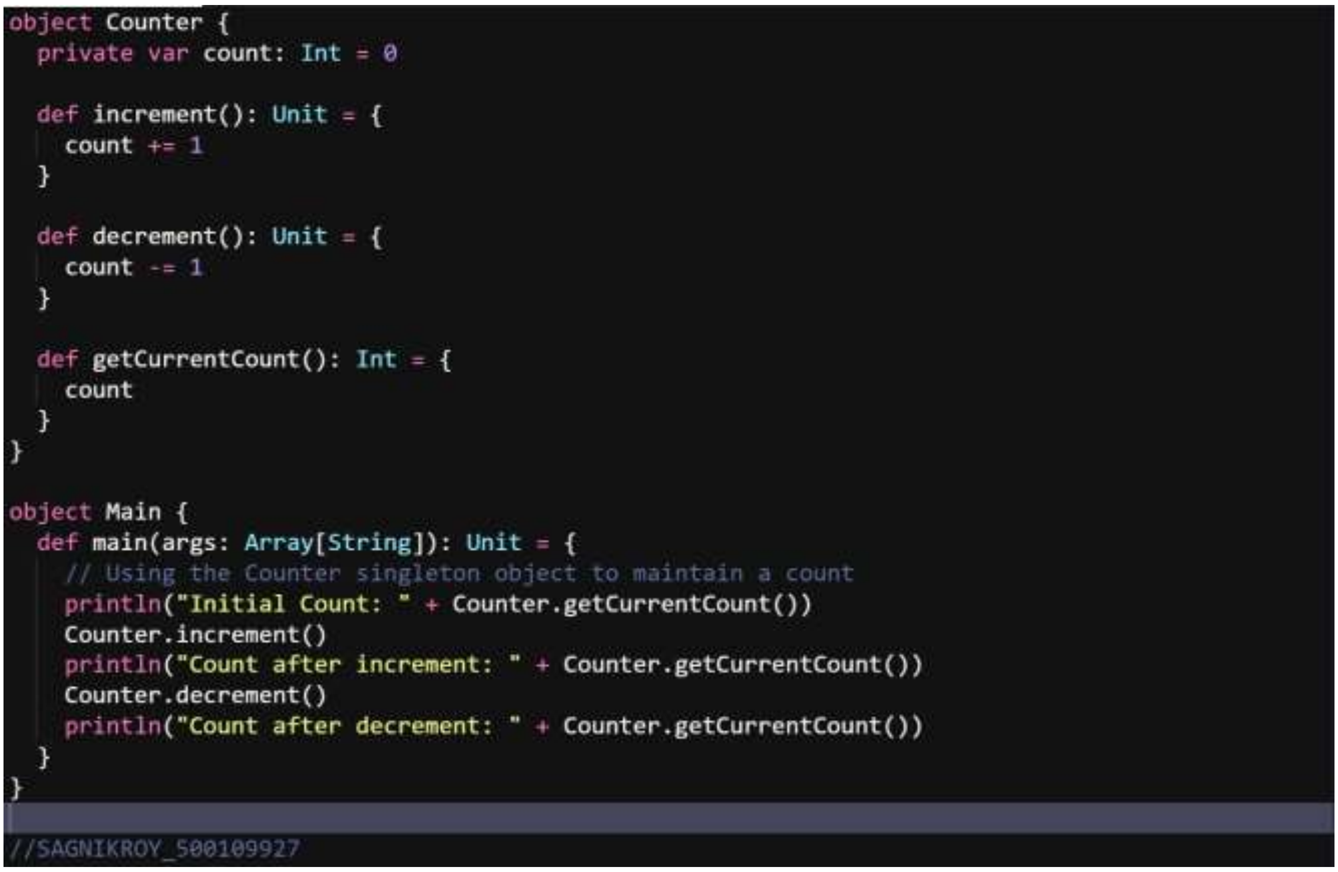
Output:


Output:

```
Hi, my name is Sagnik and I'm 30 years old.
Hi, my name is Roy and I'm 35 years old.
I am Roy. I work in IT and my employee id is 12345.
```

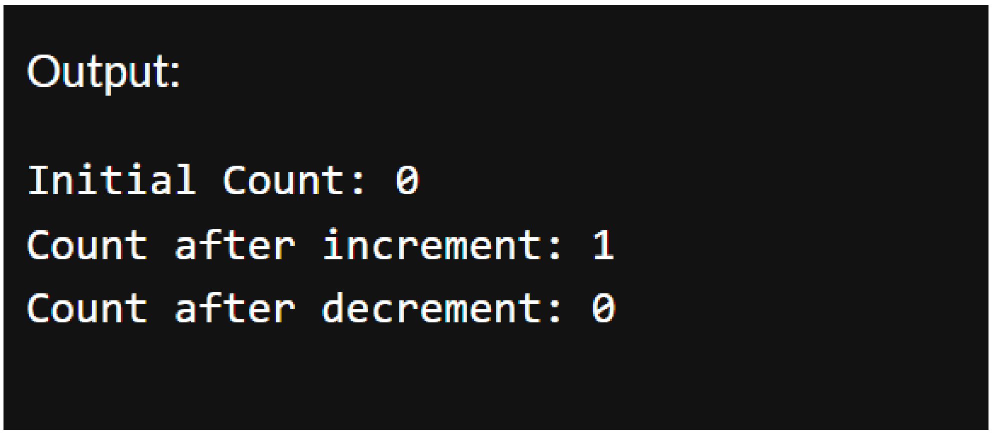
Q5. Implement a Scala Singleton object named Counter that keeps track of a variable count. Provide methods to increment, decrement, and retrieve the current count value. Demonstrate how to use this Singleton object to maintain a count in your application.

Solution:



```scala
object Counter {
  private var count: Int = 0

  def increment(): Unit = {
    count += 1
  }

  def decrement(): Unit = {
    count -= 1
  }

  def getCurrentCount(): Int = {
    count
  }
}

object Main {
  def main(args: Array[String]): Unit = {
    // Using the Counter singleton object to maintain a count
    println("Initial Count: " + Counter.getCurrentCount())
    Counter.increment()
    println("Count after increment: " + Counter.getCurrentCount())
    Counter.decrement()
    println("Count after decrement: " + Counter.getCurrentCount())
  }
}

//SAGNIKROY_500109927
```

Output:



Output:

```
Initial Count: 0
Count after increment: 1
Count after decrement: 0
```