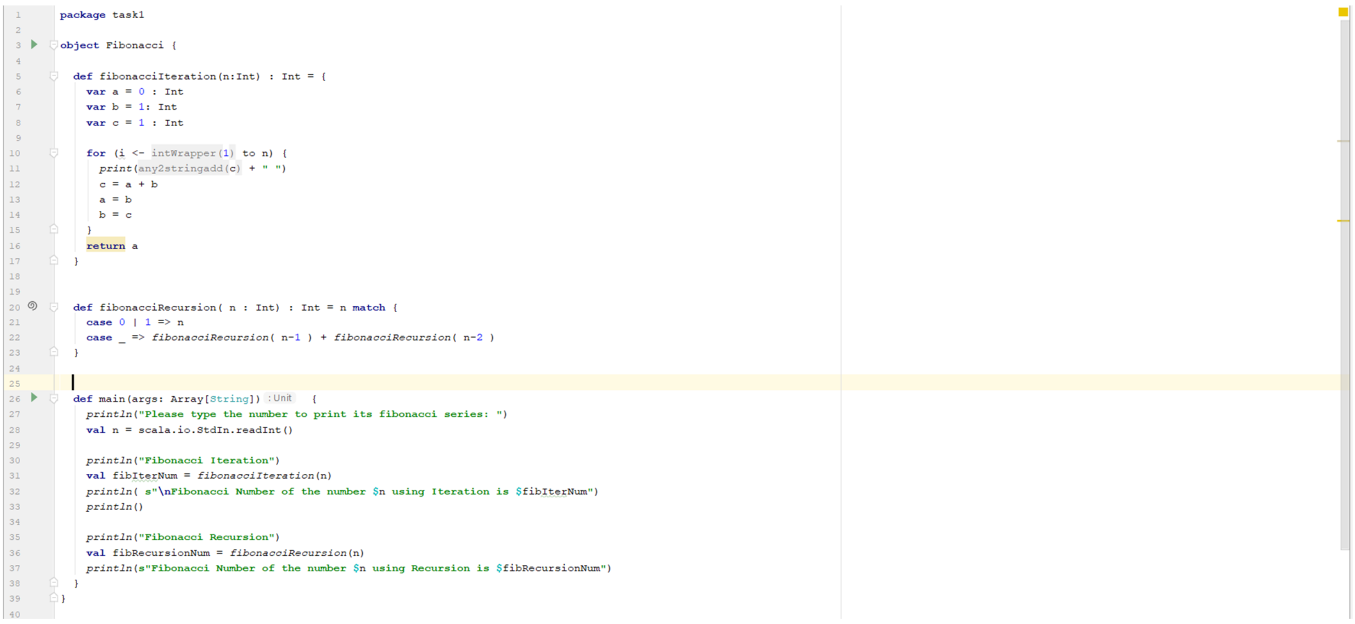
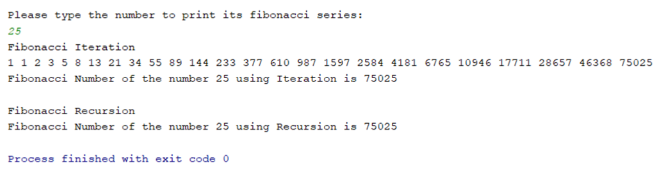
Task 1: Fibonacci Series

Code:



```scala
package task1

object Fibonacci {

  def fibonacciIteration(n:Int) : Int = {
    var a = 0 : Int
    var b = 1: Int
    var c = 1 : Int

    for (i <- intWrapper(1) to n) {
      print(any2stringadd(c) + " ")
      c = a + b
      a = b
      b = c
    }
    return a
  }


  def fibonacciRecursion( n : Int) : Int = n match {
    case 0 | 1 => n
    case _ => fibonacciRecursion( n-1 ) + fibonacciRecursion( n-2 )
  }


  def main(args: Array[String]) : Unit   {
    println("Please type the number to print its fibonacci series: ")
    val n = scala.io.StdIn.readInt()

    println("Fibonacci Iteration")
    val fibIterNum = fibonacciIteration(n)
    println( s"\nFibonacci Number of the number $n using Iteration is $fibIterNum")
    println()

    println("Fibonacci Recursion")
    val fibRecursionNum = fibonacciRecursion(n)
    println(s"Fibonacci Number of the number $n using Recursion is $fibRecursionNum")
  }
}
```

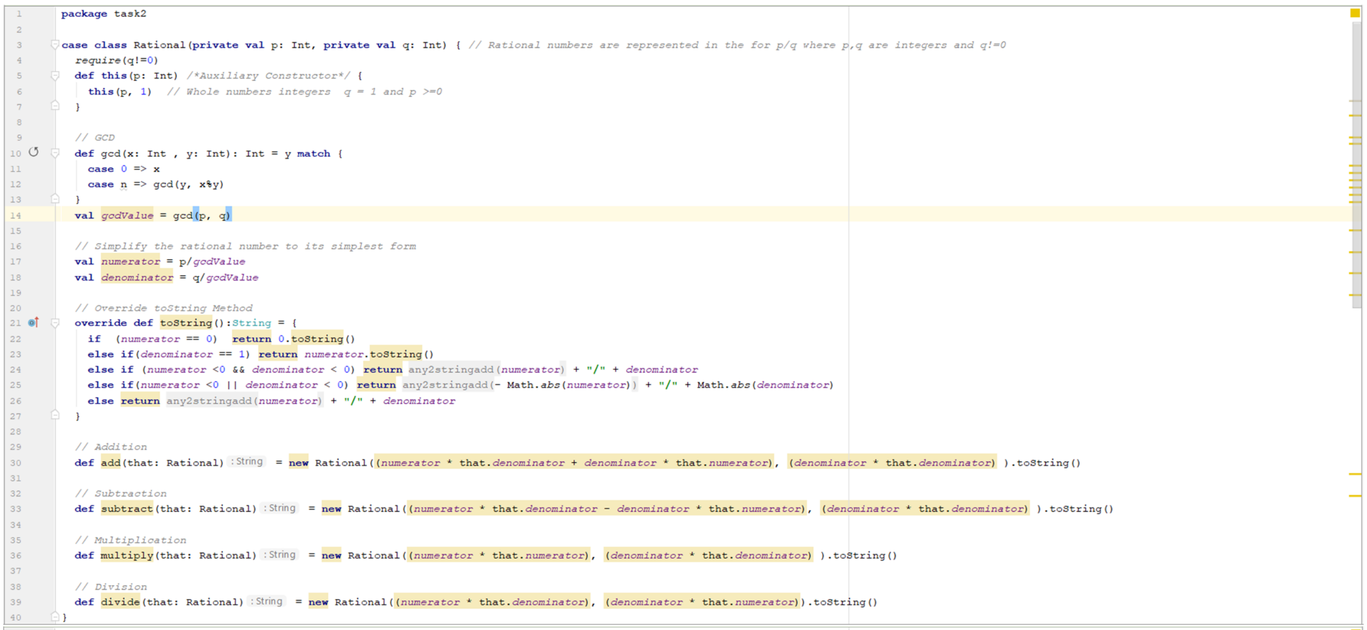
Output:



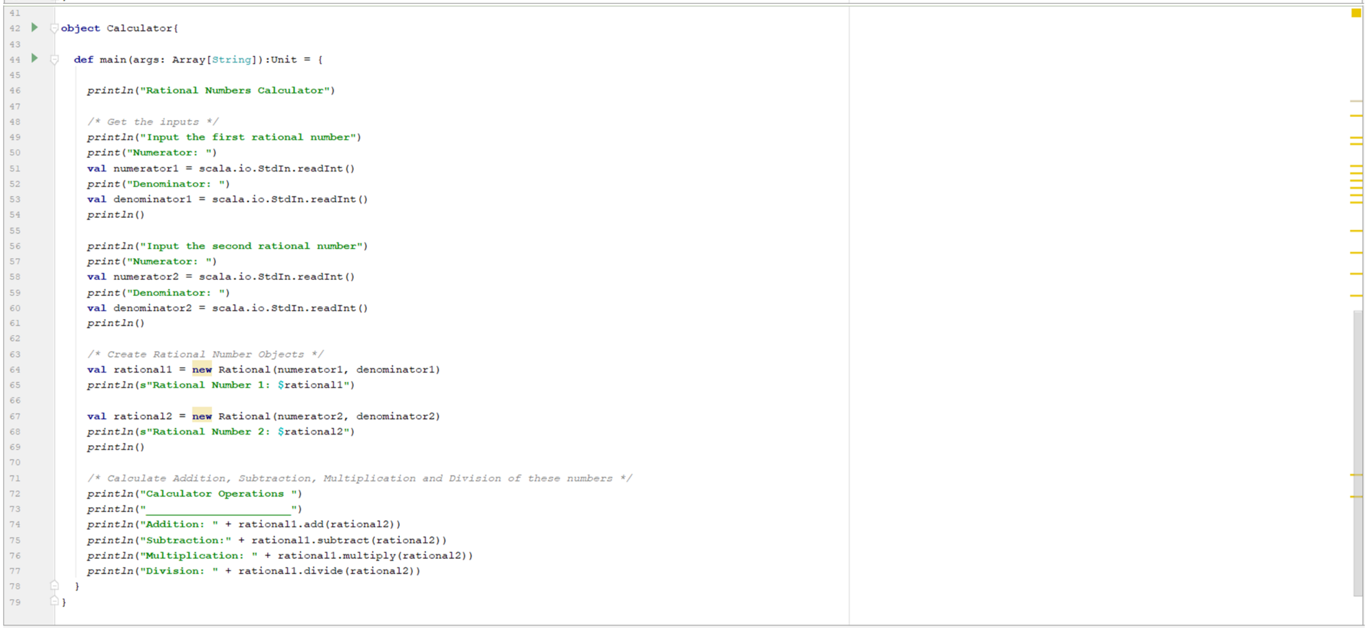
```
Please type the number to print its fibonacci series:
25
Fibonacci Iteration
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025
Fibonacci Number of the number 25 using Iteration is 75025

Fibonacci Recursion
Fibonacci Number of the number 25 using Recursion is 75025

Process finished with exit code 0
```

Task 2: Create a calculator to work with rational numbers

Code:

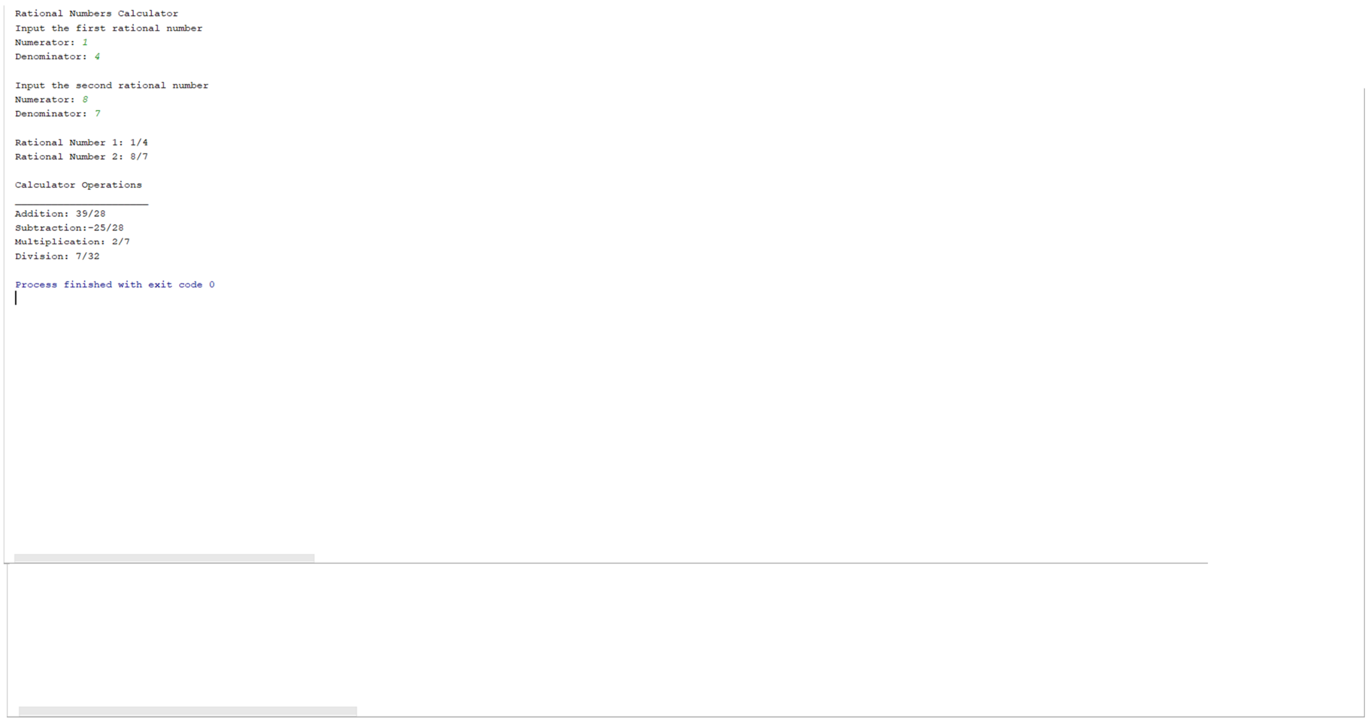


```scala
package task2

case class Rational(private val p: Int, private val q: Int) { // Rational numbers are represented in the for p/q where p,q are integers and q!=0
  require(q!=0)
  def this(p: Int) /*Auxiliary Constructor*/ {
    this(p, 1)  // Whole numbers integers  q = 1 and p >=0
  }

  // GCD
  def gcd(x: Int , y: Int): Int = y match {
    case 0 => x
    case n => gcd(y, x%y)
  }
  val gcdValue = gcd(p, q)

  // Simplify the rational number to its simplest form
  val numerator = p/gcdValue
  val denominator = q/gcdValue

  // Override toString Method
  override def toString():String = {
    if  (numerator == 0)  return 0.toString()
    else if(denominator == 1) return numerator.toString()
    else if (numerator <0 && denominator < 0) return any2stringadd(numerator) + "/" + denominator
    else if(numerator <0 || denominator < 0) return any2stringadd(- Math.abs(numerator)) + "/" + Math.abs(denominator)
    else return any2stringadd(numerator) + "/" + denominator
  }

  // Addition
  def add(that: Rational) :String  = new Rational((numerator * that.denominator + denominator * that.numerator), (denominator * that.denominator) ).toString()

  // Subtraction
  def subtract(that: Rational) :String  = new Rational((numerator * that.denominator - denominator * that.numerator), (denominator * that.denominator) ).toString()

  // Multiplication
  def multiply(that: Rational) :String  = new Rational((numerator * that.numerator), (denominator * that.denominator) ).toString()

  // Division
  def divide(that: Rational) :String  = new Rational((numerator * that.denominator), (denominator * that.numerator)).toString()
}
```

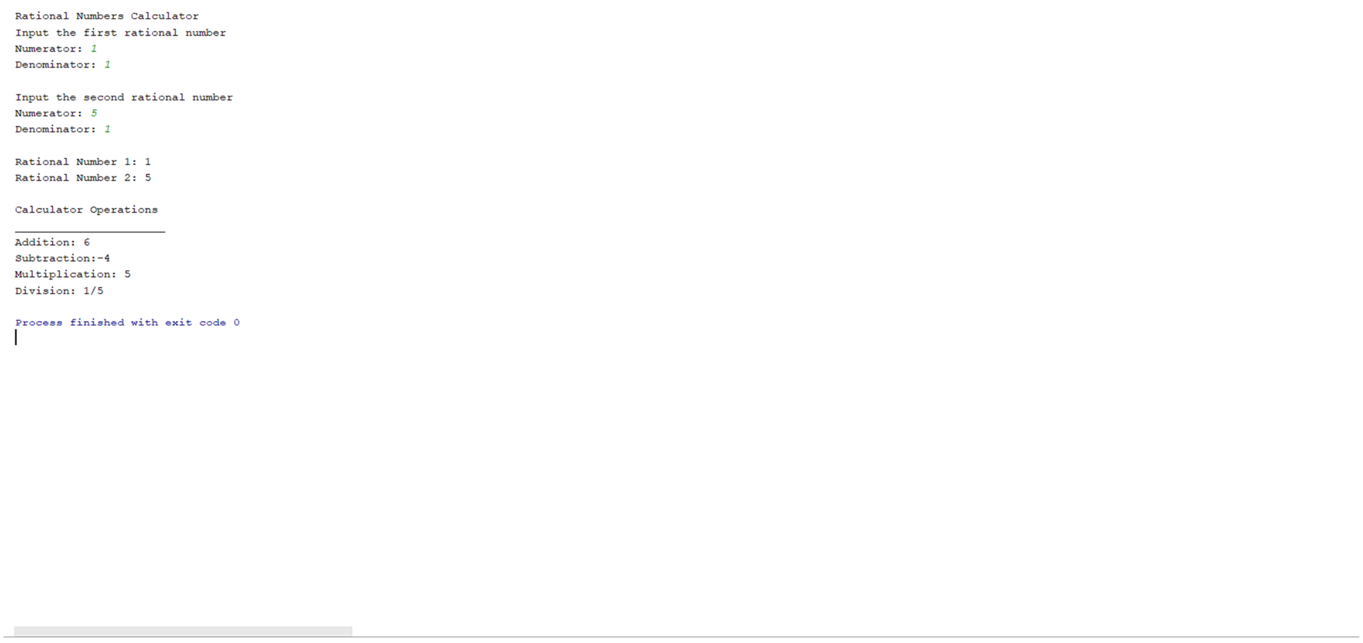


```scala

object Calculator{

  def main(args: Array[String]):Unit = {

    println("Rational Numbers Calculator")

    /* Get the inputs */
    println("Input the first rational number")
    print("Numerator: ")
    val numerator1 = scala.io.StdIn.readInt()
    print("Denominator: ")
    val denominator1 = scala.io.StdIn.readInt()
    println()

    println("Input the second rational number")
    print("Numerator: ")
    val numerator2 = scala.io.StdIn.readInt()
    print("Denominator: ")
    val denominator2 = scala.io.StdIn.readInt()
    println()

    /* Create Rational Number Objects */
    val rational1 = new Rational(numerator1, denominator1)
    println(s"Rational Number 1: $rational1")

    val rational2 = new Rational(numerator2, denominator2)
    println(s"Rational Number 2: $rational2")
    println()

    /* Calculate Addition, Subtraction, Multiplication and Division of these numbers */
    println("Calculator Operations ")
    println("______________________")
    println("Addition: " + rational1.add(rational2))
    println("Subtraction:" + rational1.subtract(rational2))
    println("Multiplication: " + rational1.multiply(rational2))
    println("Division: " + rational1.divide(rational2))
  }
}
```

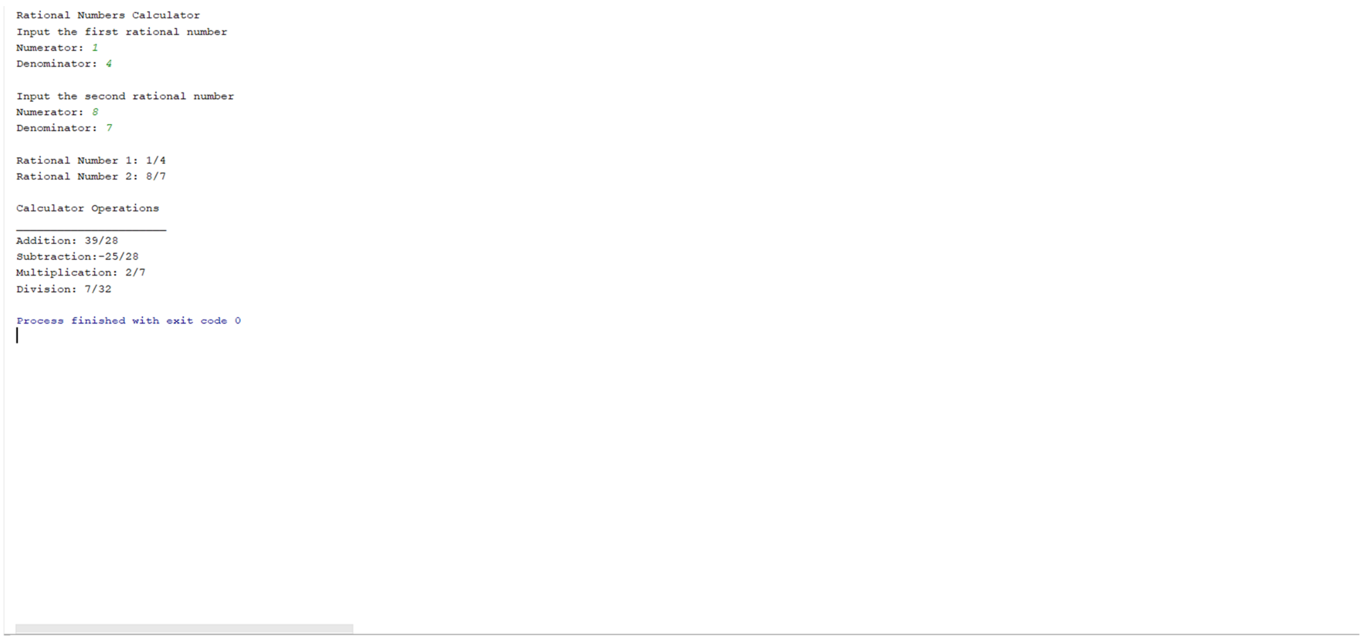
Output:



```
Rational Numbers Calculator
Input the first rational number
Numerator: 1
Denominator: 4

Input the second rational number
Numerator: 8
Denominator: 7

Rational Number 1: 1/4
Rational Number 2: 8/7

Calculator Operations
______________________
Addition: 39/28
Subtraction:-25/28
Multiplication: 2/7
Division: 7/32

Process finished with exit code 0
```



```
Rational Numbers Calculator
Input the first rational number
Numerator: 1
Denominator: 1

Input the second rational number
Numerator: 5
Denominator: 1

Rational Number 1: 1
Rational Number 2: 5

Calculator Operations
______________________
Addition: 6
Subtraction:-4
Multiplication: 5
Division: 1/5

Process finished with exit code 0
```

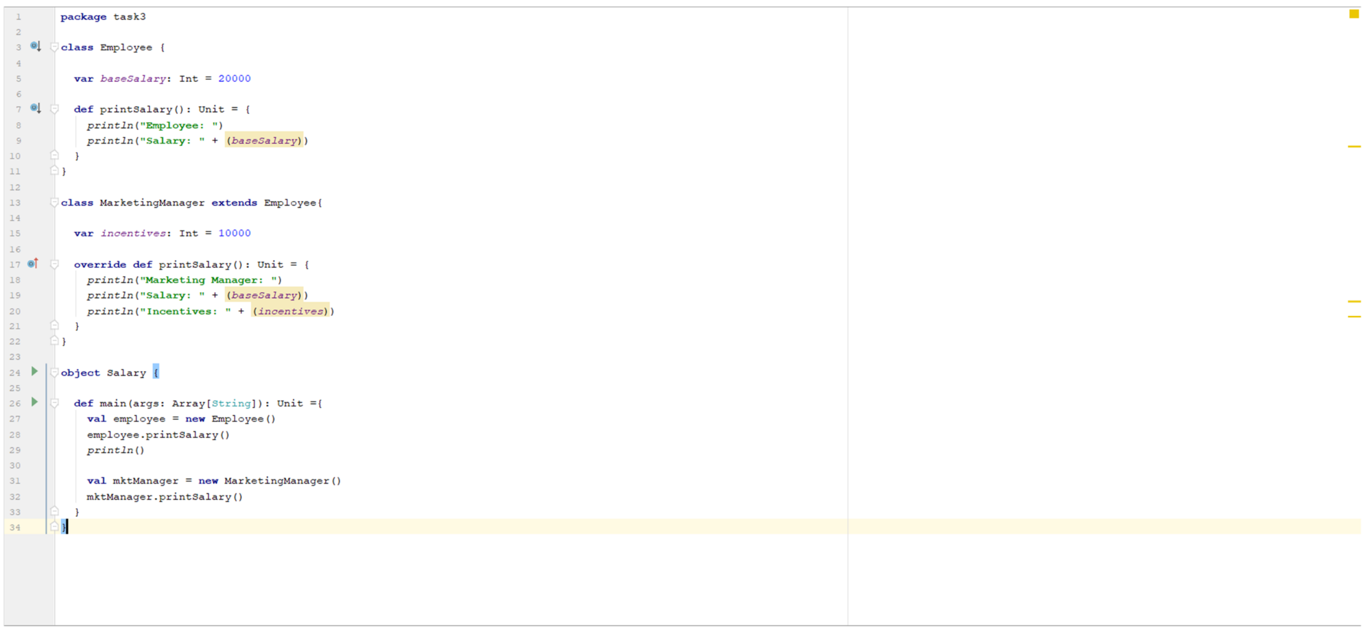


```
Rational Numbers Calculator
Input the first rational number
Numerator: 1
Denominator: 4

Input the second rational number
Numerator: 8
Denominator: 7

Rational Number 1: 1/4
Rational Number 2: 8/7

Calculator Operations
______________________
Addition: 39/28
Subtraction:-25/28
Multiplication: 2/7
Division: 7/32

Process finished with exit code 0
```

Task 3:

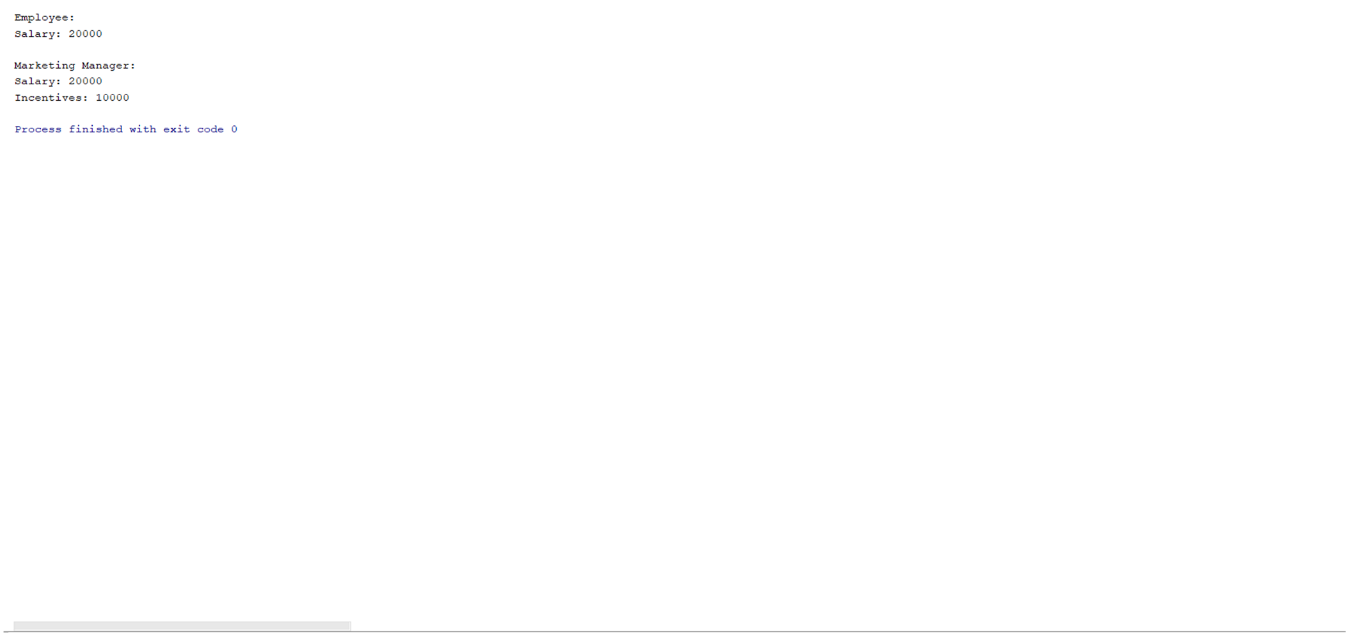
Part 1: Write a simple program to show inheritance in scala

Code:



```scala
package task3

class Employee {

  var baseSalary: Int = 20000

  def printSalary(): Unit = {
    println("Employee: ")
    println("Salary: " + (baseSalary))
  }
}

class MarketingManager extends Employee{

  var incentives: Int = 10000

  override def printSalary(): Unit = {
    println("Marketing Manager: ")
    println("Salary: " + (baseSalary))
    println("Incentives: " + (incentives))
  }
}

object Salary {

  def main(args: Array[String]): Unit ={
    val employee = new Employee()
    employee.printSalary()
    println()

    val mktManager = new MarketingManager()
    mktManager.printSalary()
  }
}
```

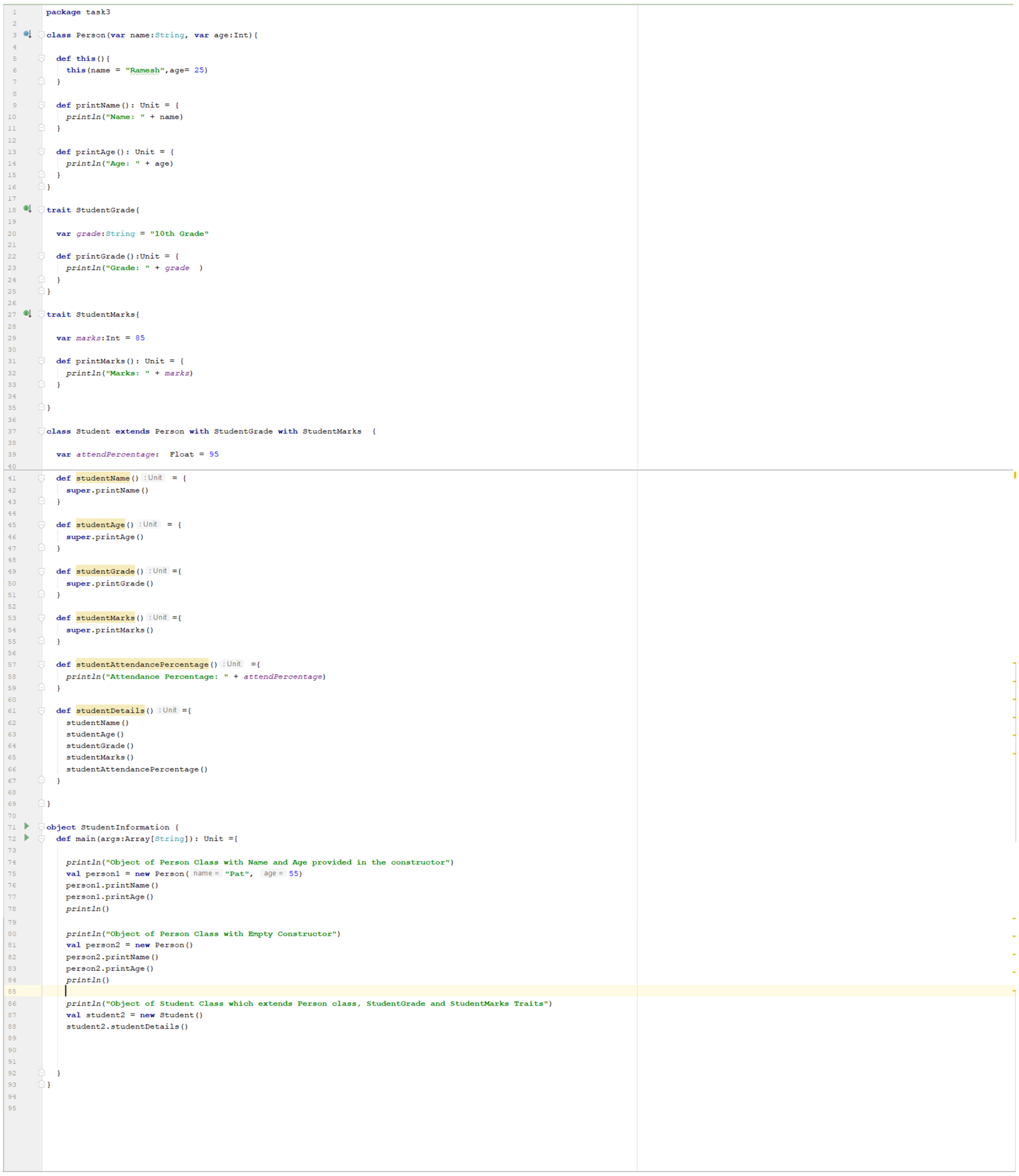
Output:



```
Employee:
Salary: 20000

Marketing Manager:
Salary: 20000
Incentives: 10000

Process finished with exit code 0
```

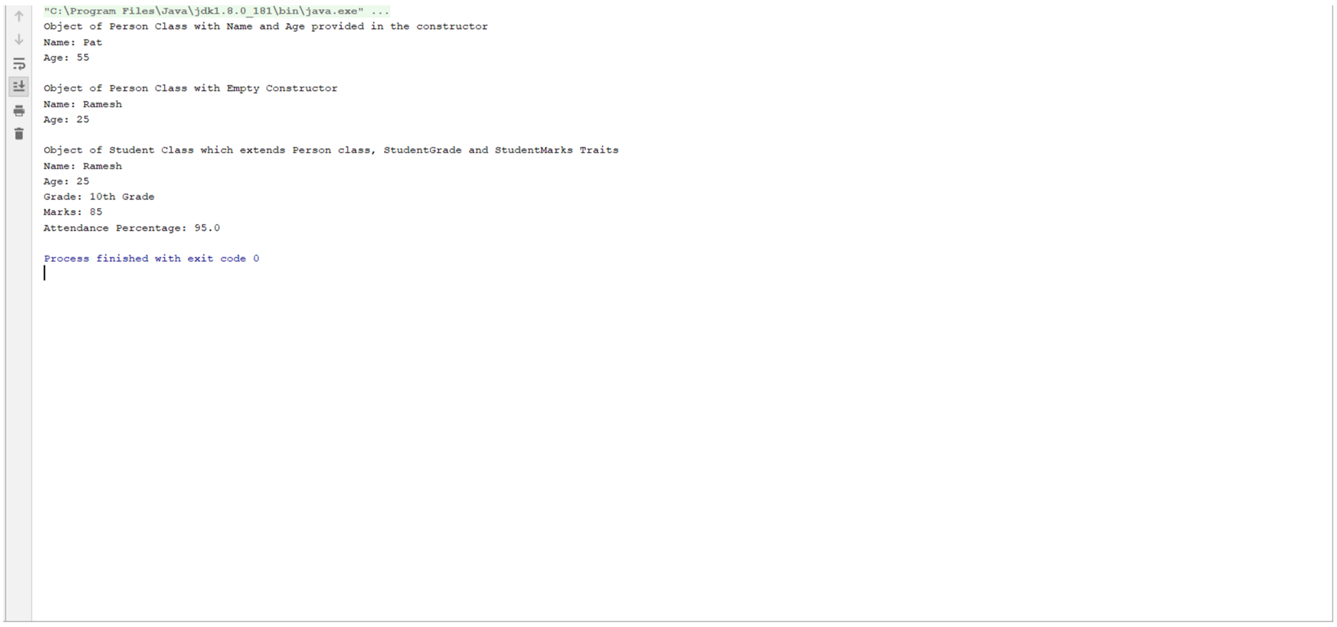
Part 2: Write a simple program to show multiple inheritance (not multi-level) in scala.

Code:



```scala
package task3

class Person(var name:String, var age:Int){

  def this(){
    this(name = "Ramesh",age= 25)
  }

  def printName(): Unit = {
    println("Name: " + name)
  }

  def printAge(): Unit = {
    println("Age: " + age)
  }
}

trait StudentGrade{

  var grade:String = "10th Grade"

  def printGrade():Unit = {
    println("Grade: " + grade   )
  }
}

trait StudentMarks{

  var marks:Int = 85

  def printMarks(): Unit = {
    println("Marks: " + marks)
  }

}

class Student extends Person with StudentGrade with StudentMarks  {

  var attendPercentage:  Float = 95

  def studentName() : Unit  = {
    super.printName()
  }

  def studentAge() : Unit  = {
    super.printAge()
  }

  def studentGrade() : Unit ={
    super.printGrade()
  }

  def studentMarks() : Unit ={
    super.printMarks()
  }

  def studentAttendancePercentage() : Unit  ={
    println("Attendance Percentage: " + attendPercentage)
  }

  def studentDetails() : Unit ={
    studentName()
    studentAge()
    studentGrade()
    studentMarks()
    studentAttendancePercentage()
  }

}

object StudentInformation {
  def main(args:Array[String]): Unit ={

    println("Object of Person Class with Name and Age provided in the constructor")
    val person1 = new Person( name = "Pat",  age = 55)
    person1.printName()
    person1.printAge()
    println()

    println("Object of Person Class with Empty Constructor")
    val person2 = new Person()
    person2.printName()
    person2.printAge()
    println()

    println("Object of Student Class which extends Person class, StudentGrade and StudentMarks Traits")
    val student2 = new Student()
    student2.studentDetails()



  }
}
```

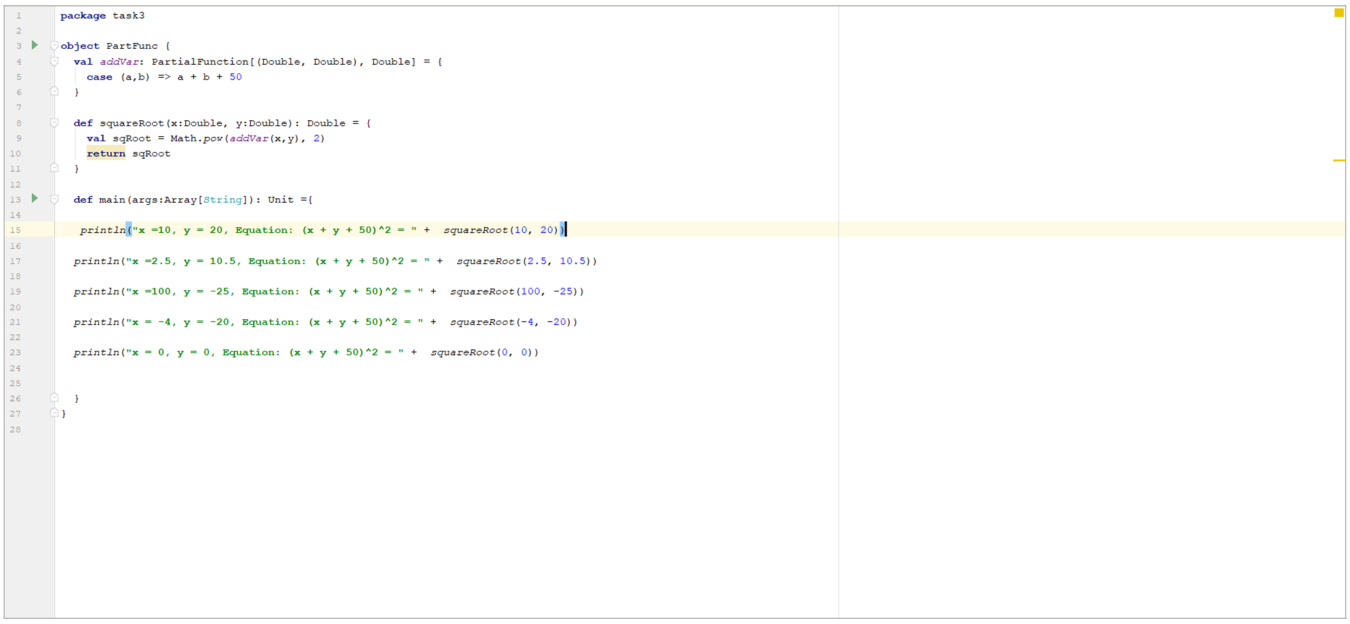
Output:



```
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
Object of Person Class with Name and Age provided in the constructor
Name: Pat
Age: 55

Object of Person Class with Empty Constructor
Name: Ramesh
Age: 25

Object of Student Class which extends Person class, StudentGrade and StudentMarks Traits
Name: Ramesh
Age: 25
Grade: 10th Grade
Marks: 85
Attendance Percentage: 95.0

Process finished with exit code 0
```

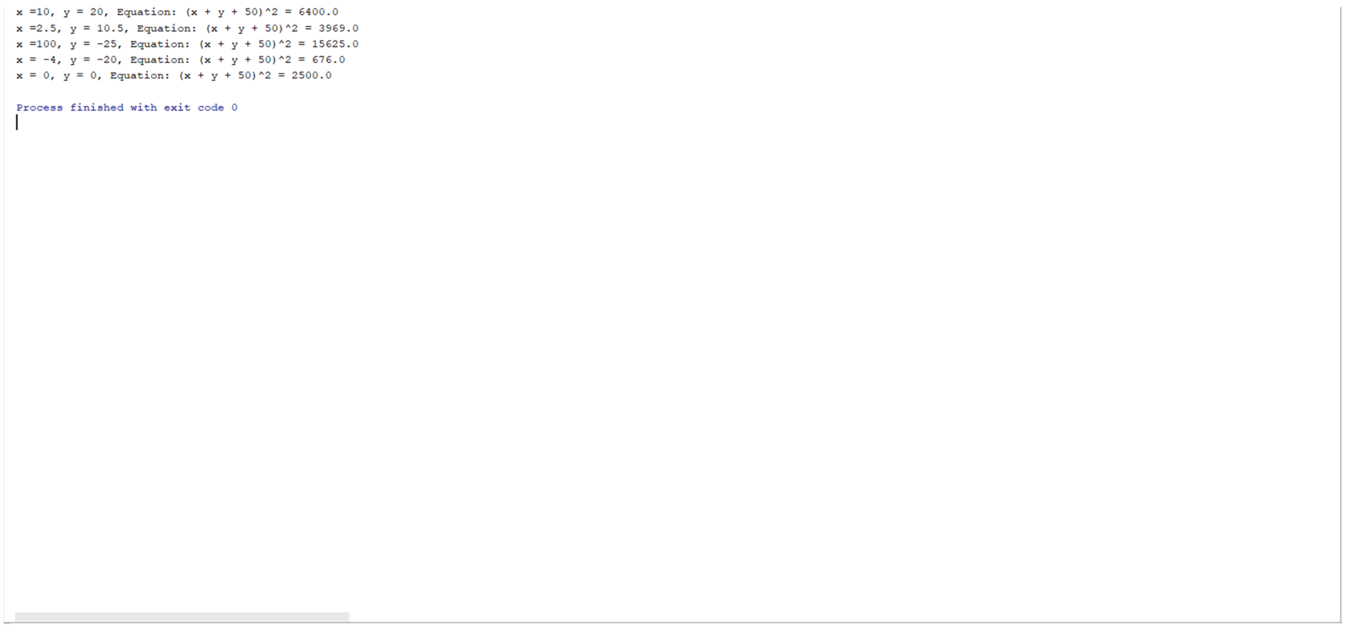
Part 3: Write a partial function to add three numbers in which one number is constant and two numbers can be passed as inputs and define another method which can take the partial function as input and squares the result

Code:



```scala
package task3

object PartFunc {
  val addVar: PartialFunction[(Double, Double), Double] = {
    case (a,b) => a + b + 50
  }

  def squareRoot(x:Double, y:Double): Double = {
    val sqRoot = Math.pow(addVar(x,y), 2)
    return sqRoot
  }

  def main(args:Array[String]): Unit ={

   println("x =10, y = 20, Equation: (x + y + 50)^2 = " +  squareRoot(10, 20))

  println("x =2.5, y = 10.5, Equation: (x + y + 50)^2 = " +  squareRoot(2.5, 10.5))

  println("x =100, y = -25, Equation: (x + y + 50)^2 = " +  squareRoot(100, -25))

  println("x = -4, y = -20, Equation: (x + y + 50)^2 = " +  squareRoot(-4, -20))

  println("x = 0, y = 0, Equation: (x + y + 50)^2 = " +  squareRoot(0, 0))


  }
}
```

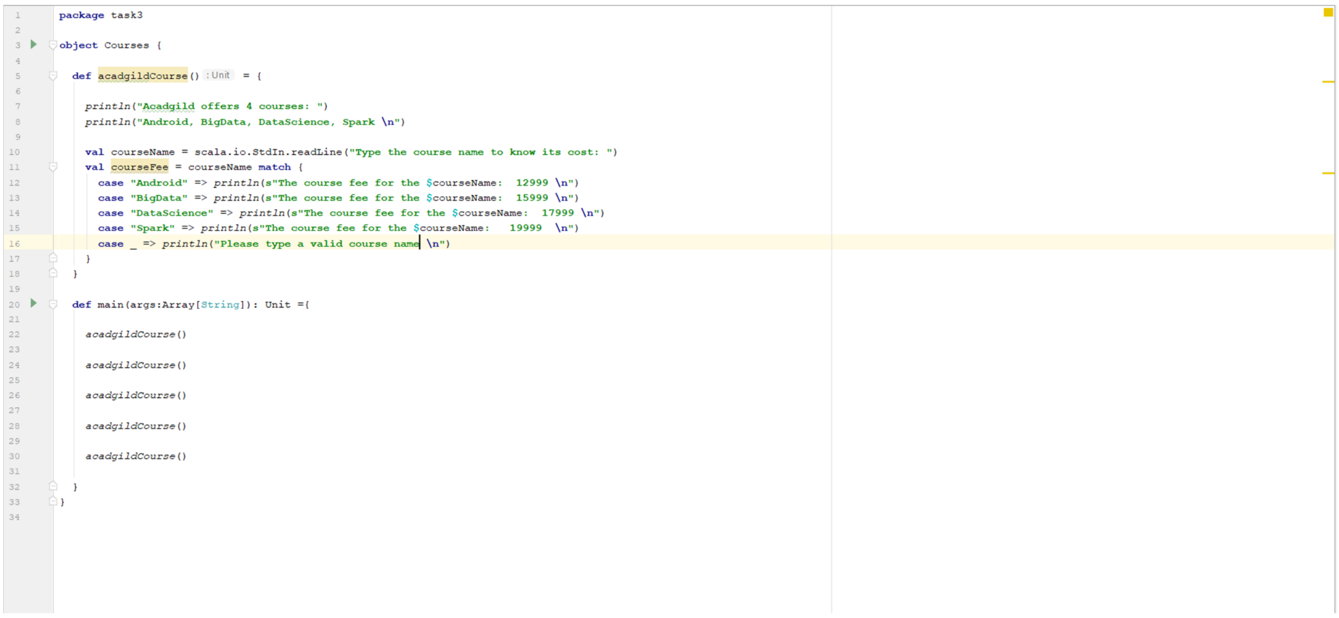
Output:



```
x =10, y = 20, Equation: (x + y + 50)^2 = 6400.0
x =2.5, y = 10.5, Equation: (x + y + 50)^2 = 3969.0
x =100, y = -25, Equation: (x + y + 50)^2 = 15625.0
x = -4, y = -20, Equation: (x + y + 50)^2 = 676.0
x = 0, y = 0, Equation: (x + y + 50)^2 = 2500.0

Process finished with exit code 0
```

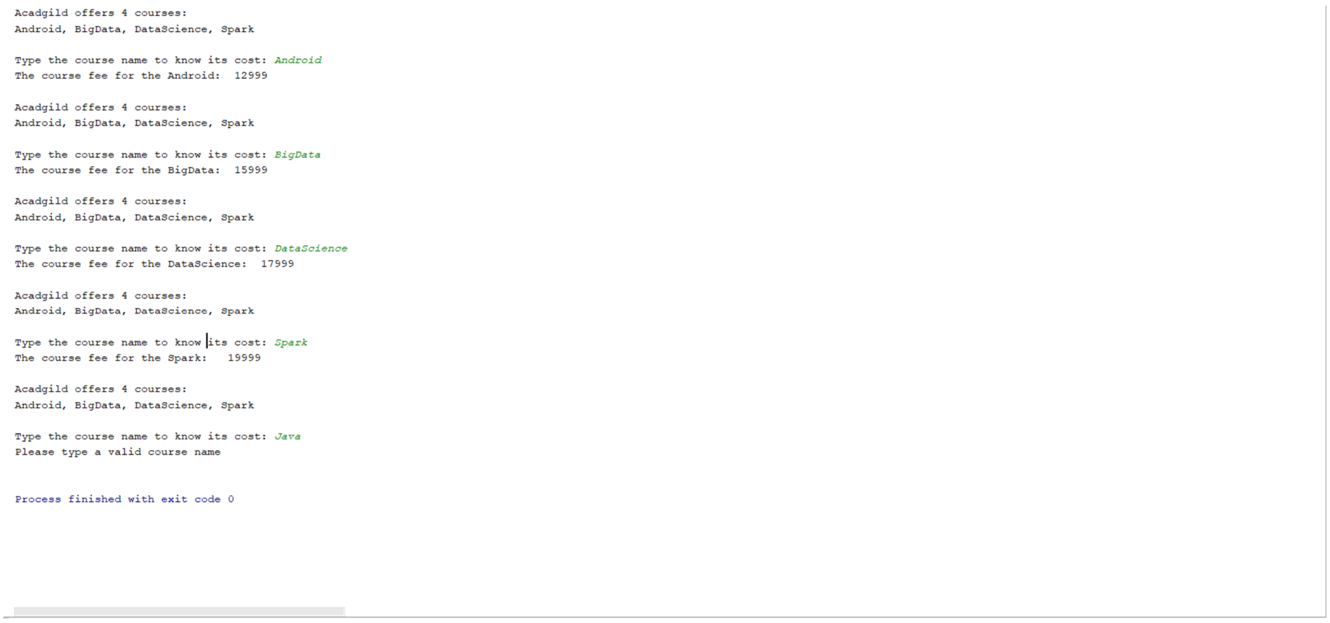
Part 4: Write a program to print the prices of 4 courses of Acadgild: Android-12999,Big Data Development-17999,Big Data Development-17999,Spark-19999 using match and add a default condition if the user enters any other course

Code:



```scala
package task3

object Courses {

  def acadgildCourse() = {

    println("Acadgild offers 4 courses: ")
    println("Android, BigData, DataScience, Spark \n")

    val courseName = scala.io.StdIn.readLine("Type the course name to know its cost: ")
    val courseFee = courseName match {
      case "Android" => println(s"The course fee for the $courseName:  12999 \n")
      case "BigData" => println(s"The course fee for the $courseName:  15999 \n")
      case "DataScience" => println(s"The course fee for the $courseName:  17999 \n")
      case "Spark" => println(s"The course fee for the $courseName:   19999  \n")
      case _ => println("Please type a valid course name \n")
    }
  }

  def main(args:Array[String]): Unit ={

    acadgildCourse()

    acadgildCourse()

    acadgildCourse()

    acadgildCourse()

    acadgildCourse()

  }
}
```

Output:



```
Acadgild offers 4 courses:
Android, BigData, DataScience, Spark

Type the course name to know its cost: Android
The course fee for the Android:  12999

Acadgild offers 4 courses:
Android, BigData, DataScience, Spark

Type the course name to know its cost: BigData
The course fee for the BigData:  15999

Acadgild offers 4 courses:
Android, BigData, DataScience, Spark

Type the course name to know its cost: DataScience
The course fee for the DataScience:  17999

Acadgild offers 4 courses:
Android, BigData, DataScience, Spark

Type the course name to know its cost: Spark
The course fee for the Spark:   19999

Acadgild offers 4 courses:
Android, BigData, DataScience, Spark

Type the course name to know its cost: Java
Please type a valid course name


Process finished with exit code 0
```