databricks Assignment_22.2

```
Task 1:
A Fibonacci series (starting from 1) written in order without any spaces in between, thus
producing a sequence of digits.
Write a Scala application to find the Nth digit in the sequence.
○ Write the function using standard for loop
○ Write the function using recursion
```

```
// Write the function using standard for loop
def fib1( n : Int ) : Int = {
  var a = 0
  var b = 1
  var i = 0

  while( i < n ) {
    val c = a + b
    a = b
    b = c
    i = i + 1
  }
  return a
}
```

```
fib1: (n: Int)Int
```

```
fib1(6)
```

```
res0: Int = 8
```

```
//Write the function using recursion
def fib2( n : Int) : Int = n match {
   case 0 | 1 => n
   case _ => fib1( n-1 ) + fib1( n-2 )
}
```

```
fib2: (n: Int)Int
```

```
fib2(6)
```

```
res1: Int = 8
```

```
Task 2:
Create a calculator to work with rational numbers.
Requirements:
○ It should provide capability to add, subtract, divide and multiply rational
numbers
○ Create a method to compute GCD (this will come in handy during operations on
rational)
Add option to work with whole numbers which are also rational numbers i.e. (n/1)
- achieve the above using auxiliary constructors
- enable method overloading to enable each function to work with numbers and rational.
```

```
class rational(){
  var num =0
  var denom =1
  var value : Double = 1
  def this(num :Int , denom : Int) {
    this()
    this.num = num
    this.denom = denom
    this.value = this.num.toFloat/this.denom
  }

}
```

```
defined class rational
```

```
class calculator{
def add( a : Double , b: Double): Double =
  {
    return a+b
  }

def sub( a : Double , b: Double): Double =
  {
    return a-b
  }
def mul( a : Double , b: Double): Double =
  {
    return a*b
  }

def div( a : Double , b: Double): Double =
  {
    return a.toFloat/b
  }
  def gcd( a : Int , b: Int): Int =
  {
    var g=1
    if(a>b){
      for(i <- 2 to b+1){
        if(a%i==0 && b%i==0)
                g=i
      }

    }

    else{
        for(i <- 2 to a+1){
        if(a%i==0 && b%i==0)
                g=i
      }
    }
    return g
  }
}
```

```
defined class calculator
```

```
val cal = new calculator()
cal.add(5,6)
```

```
cal: calculator = calculator@63d8c4a7
```

```
res0: Double = 11.0
```

```
//...1/2 + 3/4
val a = new rational(1,2) // ...1/2
val b = new rational(3,4) //....3/4
cal.add(a.value,b.value)
```

```
a: rational = rational@bbe395d
b: rational = rational@7750de86
res9: Double = 1.25
```

```
cal.gcd(10,20)
```

```
res13: Int = 10
```

```
cal.div(5,2)
```

```
res11: Double = 2.5
```

```
task 3:


1.Write a simple program to show inheritance in scala.
```

```
class Employee{
    var salary:Float = 10000
}

class Programmer extends Employee{
    var bonus:Int = 5000
    println("Salary = "+salary)
    println("Bonus = "+bonus)
}

object MainObject{
    def main(args:Array[String]){
        new Programmer()
    }
}
```

```
defined class Employee
defined class Programmer
defined object MainObject
```

```
2.Write a simple program to show multiple inheritance in scala.
```

```
class A{
    var salary1 = 10000
}

class B extends A{
    var salary2 = 20000
}

class C extends B{
    def show(){
        println("salary1 = "+salary1)
        println("salary2 = "+salary2)
    }
}

object MainObject{
    def main(args:Array[String]){{
        var c = new C()
        c.show()

    }
}
}
```

```
defined class A
defined class B
defined class C
defined object MainObject
```

3Write 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.

```
class partial(){
    val  c=10
def partialadd(a : Int ,b : Int ) : Int ={

        return square(a+b+c)
}
def square(s : Int) : Int ={
        return (s*s)
}
}
```

```
defined class partial
```

```
val p = new partial()
p.partialadd(10,20)
```

```
p: partial = partial@485c1b87
res6: Int = 1600
```

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

```
def price(s : String){
  s match{
    case "Android" => println("12999")
    case "Big Data Development" => println("17999")
    case "Spark" => println("19999")
    case _ => println("we dont provide that course")

  }
}
```

```
price: (s: String)Unit
```

```
price("Android")
```

```
12999
```

```
we dont provide that course
```