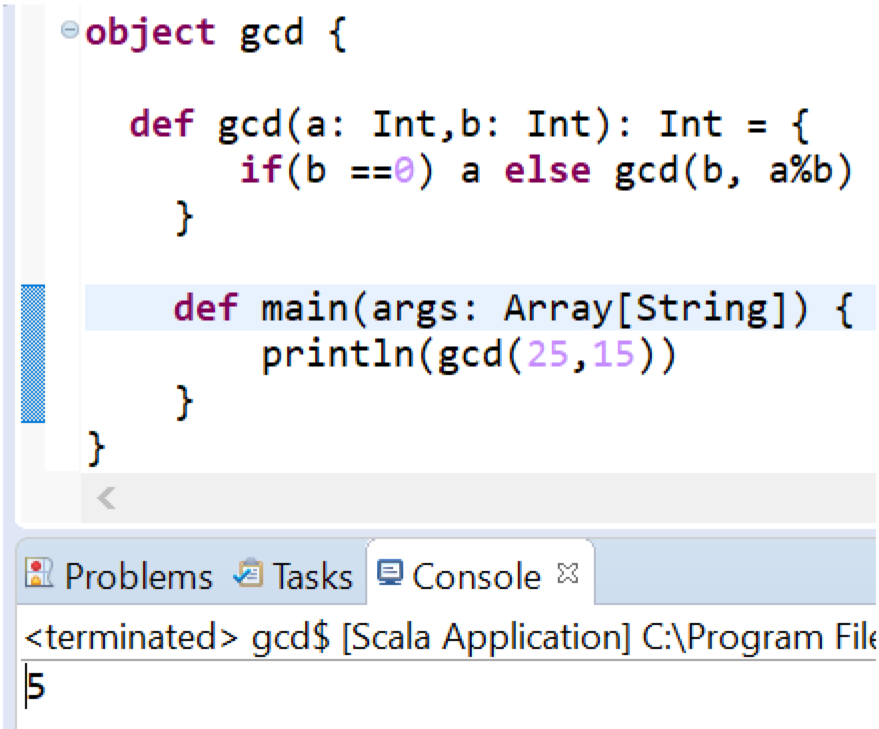
Task 1 : GCD of two numbers

```
object gcd {

  def gcd(a: Int,b: Int): Int = {
       if(b ==0) a else gcd(b, a%b)
    }

    def main(args: Array[String]) {
        println(gcd(25,15))
    }
}
```



```
object gcd {

  def gcd(a: Int,b: Int): Int = {
       if(b ==0) a else gcd(b, a%b)
    }

    def main(args: Array[String]) {
        println(gcd(25,15))
    }
}
```

```
<terminated> gcd$ [Scala Application] C:\Program File
5
```

Task 2 : Fibonacci

a)Using For loop

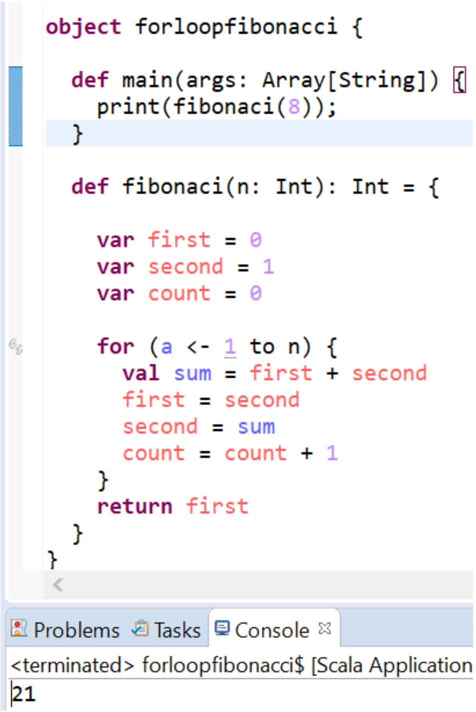
```
object forloopfibonacci {

  def main(args: Array[String]) {
    print(fibonaci(8));
  }

  def fibonaci(n: Int): Int = {

    var first = 0
    var second = 1
    var count = 0

    for (a <- 1 to n) {
      val sum = first + second
      first = second
      second = sum
      count = count + 1
    }
    return first
  }
}
```



```
object forloopfibonacci {

  def main(args: Array[String]) {
    print(fibonaci(8));
  }

  def fibonaci(n: Int): Int = {

    var first = 0
    var second = 1
    var count = 0

    for (a <- 1 to n) {
      val sum = first + second
      first = second
      second = sum
      count = count + 1
    }
    return first
  }
}
```

```
<terminated> forloopfibonacci$ [Scala Application
21
```

b)Using recursion

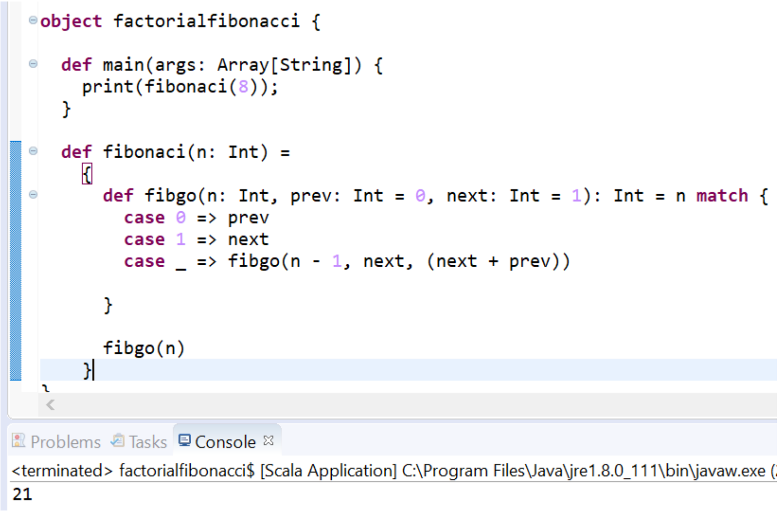
```
object factorialfibonacci {

  def main(args: Array[String]) {
    print(fibonaci(8));
  }

  def fibonaci(n: Int) =
    {
      def fibgo(n: Int, prev: Int = 0, next: Int = 1): Int = n match {
        case 0 => prev
        case 1 => next
        case _ => fibgo(n - 1, next, (next + prev))

      }

      fibgo(n)
    }
}
```



```
object factorialfibonacci {

  def main(args: Array[String]) {
    print(fibonaci(8));
  }

  def fibonaci(n: Int) =
    {
      def fibgo(n: Int, prev: Int = 0, next: Int = 1): Int = n match {
        case 0 => prev
        case 1 => next
        case _ => fibgo(n - 1, next, (next + prev))

      }

      fibgo(n)
    }
```

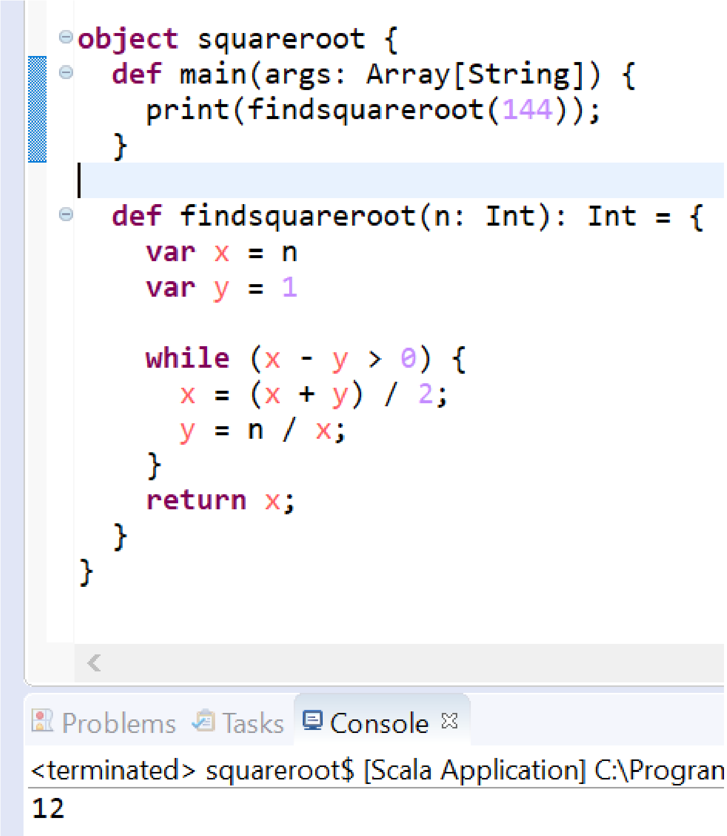
```
<terminated> factorialfibonacci$ [Scala Application] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (
21
```

Task 3: Find Square root using Babylonian method

```
object squareroot {
  def main(args: Array[String]) {
    print(findsquareroot(12));
  }

  def findsquareroot(n: Int): Int = {
    var x = n
    var y = 1

    while (x - y > 0) {
      x = (x + y) / 2;
      y = n / x;
    }
    return x;
  }
}
```



```
object squareroot {
  def main(args: Array[String]) {
    print(findsquareroot(144));
  }

  def findsquareroot(n: Int): Int = {
    var x = n
    var y = 1

    while (x - y > 0) {
      x = (x + y) / 2;
      y = n / x;
    }
    return x;
  }
}
```

```
<terminated> squareroot$ [Scala Application] C:\Progran
12
```