

Bigdata Assignment 5.4

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.

Solution -

```
object Rational {  
  def main(args: Array[String]) {  
  
    // We are passing numbers to Rational Class for three cases when two nos  
    // are integers , two nos are rationals and one is rational and the other is  
    // integer.  
    /*var a = new Rational(4,3)  
    var b = new Rational(10,7)  
    var a = new Rational(4,2)  
    var b = new Rational(10,5)*/  
    var a = new Rational(10,3)  
    var b = new Rational(6,3)  
  
    // Get the Result  
    var result = a+b  
    // Print the Result - don't print the denominator if it is 1  
    if (result.denom==1) println("Result of " + a.numer + " + " + b.numer + "  
    is : "+result.numer)  
    else if (a.denom!=1 && b.denom!=1)println("Result of  
    "+a.numer+"/"+a.denom+" + " + b.numer+"/"+b.denom+" is: " +  
    result.numer + "/" + result.denom)  
    else if (a.denom==1 && b.denom!=1)println("Result of
```

```

"+a.numer+"+"/"+b.denom+" is: " + result.numer + "/" +
result.denom)
else println("Result of "+a.numer+"/"+a.denom+" + "+ b.numer + "is: "
+ result.numer + "/" + result.denom)
}

}

```

```

class Rational(n: Int, d: Int) {
//GCD method
private def gcd(x: Int, y: Int): Int = {
if (x == 0) y
else if (x < 0) gcd(-x, y)
else if (y < 0) -gcd(x, -y)
else gcd(y % x, x)
}
//Auxillary Constructor
def this(x: Int){
this(x, 1)
}
private val g = gcd(n, d)
val numer: Int = n/g
val denom: Int = d/g

//Method Overloading for each type of operation
//Addition
def +(that: Rational) = new Rational(numer*that.denom +
that.numer*denom, denom*that.denom)

//Subtraction
def -(that: Rational) = new Rational(numer*that.denom -
that.numer*denom, denom*that.denom)

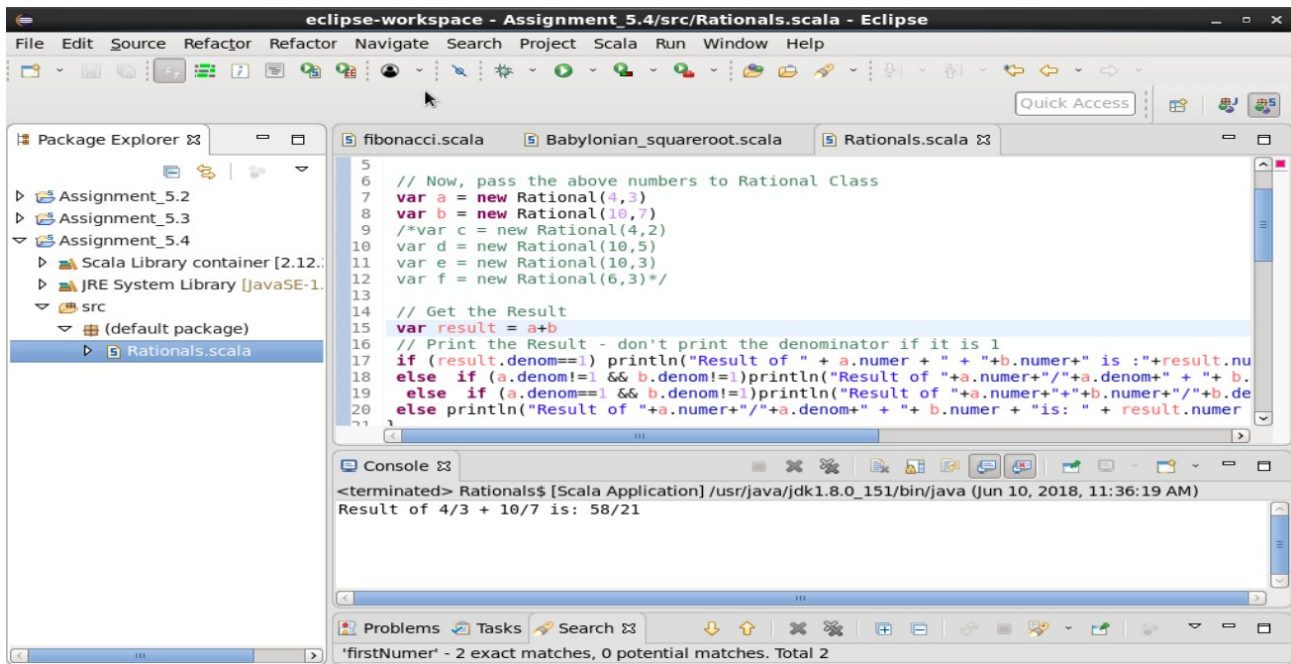
//Multiplication
def *(that: Rational) = new Rational(numer*that.numer,
denom*that.denom)

//Division
def /(that: Rational) = new Rational(numer*that.denom,
denom*that.numer)
}

```

Output -

Here input was 2 rational no($4/3$ and $10/7$) and we performed addition. The result was $58/21$.



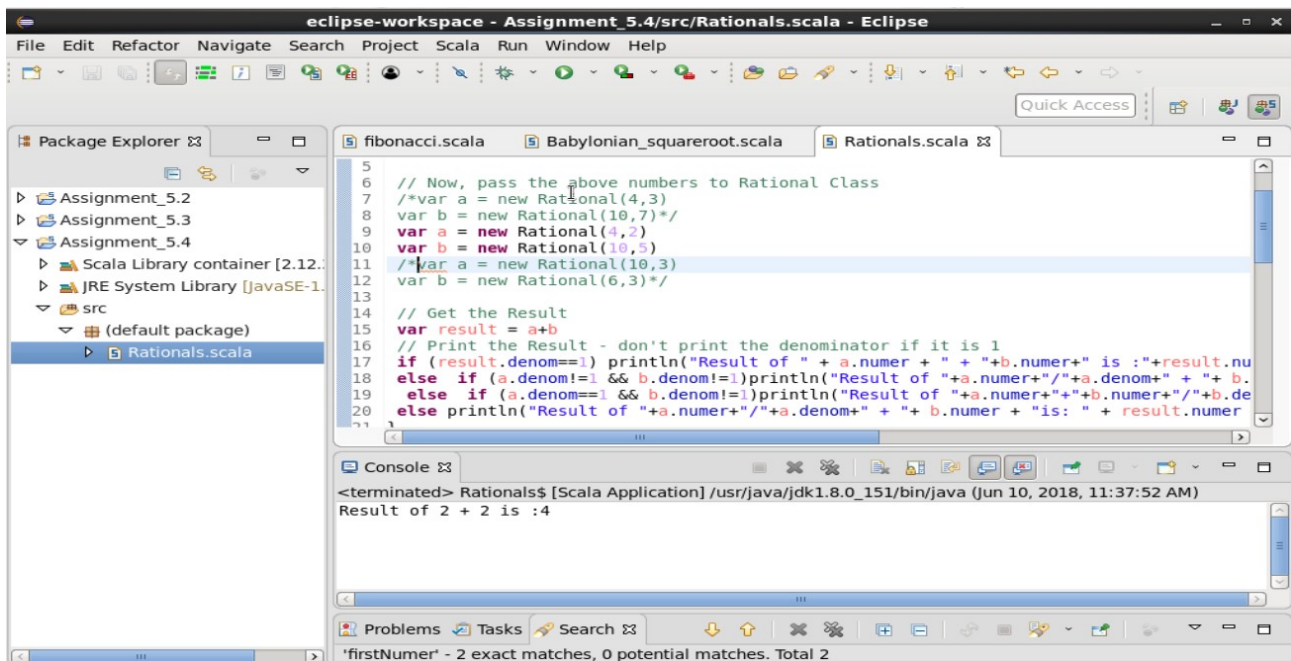
The screenshot shows the Eclipse IDE with the file `Rationals.scala` open. The code defines a `Rational` class and performs the addition of $4/3$ and $10/7$. The console output shows the result of the addition.

```
5
6 // Now, pass the above numbers to Rational Class
7 var a = new Rational(4,3)
8 var b = new Rational(10,7)
9 /*var c = new Rational(4,2)
10 var d = new Rational(10,5)
11 var e = new Rational(10,3)
12 var f = new Rational(6,3)*/
13
14 // Get the Result
15 var result = a+b
16 // Print the Result - don't print the denominator if it is 1
17 if (result.denom==1) println("Result of " + a.number + " + "+b.number+" is :"+result.nu
18 else if (a.denom!=1 && b.denom!=1)println("Result of "+a.number+"/"+a.denom+" + " + b.
19 else if (a.denom==1 && b.denom!=1)println("Result of "+a.number+"+"+b.number+"/"+b.de
20 else println("Result of "+a.number+"/"+a.denom+" + " + b.number + "is: " + result.number
```

Console Output:

```
<terminated> Rationals$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Jun 10, 2018, 11:36:19 AM)
Result of 4/3 + 10/7 is: 58/21
```

Here input was 2 integers(2 and 2) and performed addition , we got the output as 4.



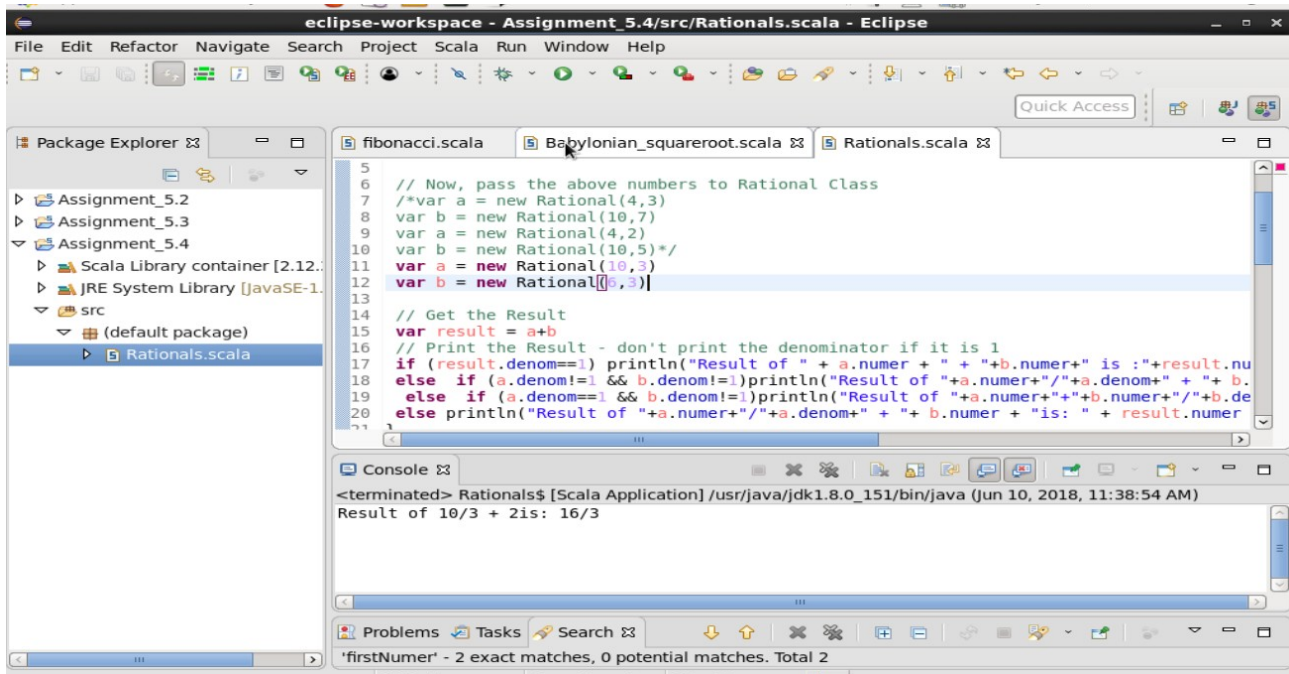
The screenshot shows the Eclipse IDE with the file `Rationals.scala` open. The code defines a `Rational` class and performs the addition of two integers, 2 and 2. The console output shows the result of the addition.

```
5
6 // Now, pass the above numbers to Rational Class
7 /*var a = new Rational(4,3)
8 var b = new Rational(10,7)*/
9 var a = new Rational(4,2)
10 var b = new Rational(10,5)
11 /*var a = new Rational(10,3)
12 var b = new Rational(6,3)*/
13
14 // Get the Result
15 var result = a+b
16 // Print the Result - don't print the denominator if it is 1
17 if (result.denom==1) println("Result of " + a.number + " + "+b.number+" is :"+result.nu
18 else if (a.denom!=1 && b.denom!=1)println("Result of "+a.number+"/"+a.denom+" + " + b.
19 else if (a.denom==1 && b.denom!=1)println("Result of "+a.number+"+"+b.number+"/"+b.de
20 else println("Result of "+a.number+"/"+a.denom+" + " + b.number + "is: " + result.number
```

Console Output:

```
<terminated> Rationals$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Jun 10, 2018, 11:37:52 AM)
Result of 2 + 2 is :4
```

Here the input is one integer 2 and one rational $10/3$, we performed addition and we got the result as $16/3$.



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure, including Assignment_5.2, Assignment_5.3, and Assignment_5.4, with the src directory expanded to show the Rational.scala file. The main editor window displays the Rational.scala file, which contains the following code:

```
5
6 // Now, pass the above numbers to Rational Class
7 /*var a = new Rational(4,3)
8 var b = new Rational(10,7)
9 var a = new Rational(4,2)
10 var b = new Rational(10,5)*/
11 var a = new Rational(10,3)
12 var b = new Rational(2,1)
13
14 // Get the Result
15 var result = a+b
16 // Print the Result - don't print the denominator if it is 1
17 if (result.denom==1) println("Result of " + a.number + " + " + b.number + " is :"+result.number)
18 else if (a.denom!=1 && b.denom!=1)println("Result of "+a.number+"/"+a.denom+" + " + b.number+"/"+b.denom+" is :"+result.number)
19 else if (a.denom==1 && b.denom!=1)println("Result of "+a.number+"/"+b.denom+" + " + b.number+"/"+b.denom+" is :"+result.number)
20 else println("Result of "+a.number+"/"+a.denom+" + " + b.number + "is: " + result.number)
21
```

The Console window at the bottom shows the output of the program:

```
<terminated> Rational$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Jun 10, 2018, 11:38:54 AM)
Result of 10/3 + 2is: 16/3
```

The Problems window at the bottom shows a search for 'firstNumer' with 2 exact matches and 0 potential matches, totaling 2 matches.