Session 16

Assignment 1 Questions

Problem Statement

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.

Program:-

```
class Rational(n: Int, d: Int) {

  require(d != 0)
  private val g = gcd(n.abs, d.abs)
```

```
val numer = n / g
val denom = d / g

def this(n: Int) = this(n, 1)

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

def + (i: Int): Rational =
  new Rational(numer + i * denom, denom)

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

def - (i: Int): Rational =
  new Rational(numer - i * denom, denom)

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

def * (i: Int): Rational =
  new Rational(numer * i, denom)

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

def / (i: Int): Rational =
  new Rational(numer, denom * i)

override def toString = numer +"/"+ denom
```

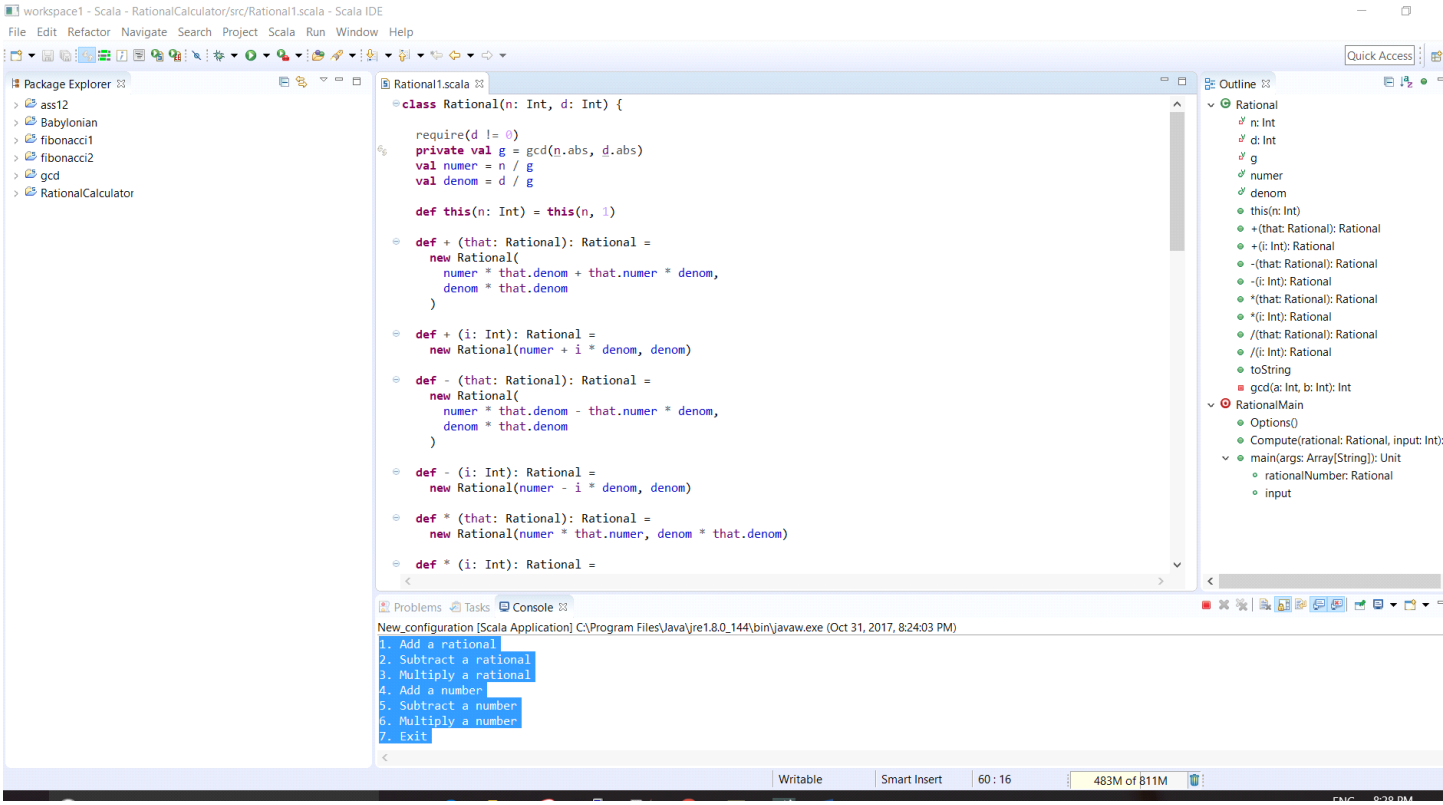
```
  private def gcd(a: Int, b: Int): Int =
    if (b == 0) a else gcd(b, a % b)
}

object RationalMain {
  def Options() = {
    println("1. Add a rational")
    println("2. Subtract a rational")
    println("3. Multiply a rational")
    println("4. Add a number")
    println("5. Subtract a number")
    println("6. Multiply a number")
    println("7. Exit")
  }

  def Compute(rational: Rational, input: Int): Rational = {

    input match {
      case 1 =>
        val p = scala.io.StdIn.readInt()
        val q = scala.io.StdIn.readInt()
        rational.+(new Rational(p, q))
      case 2 =>
        val p = scala.io.StdIn.readInt()
        val q = scala.io.StdIn.readInt()
        rational.-(new Rational(p, q))
      case 3 =>
        val p = scala.io.StdIn.readInt()
        val q = scala.io.StdIn.readInt()
        rational.*(new Rational(p, q))
      case 4 =>
        val p = scala.io.StdIn.readInt()
        rational.+(new Rational(p))
      case 5 =>
        val p = scala.io.StdIn.readInt()
        rational.-(new Rational(p))
      case 6 =>
```

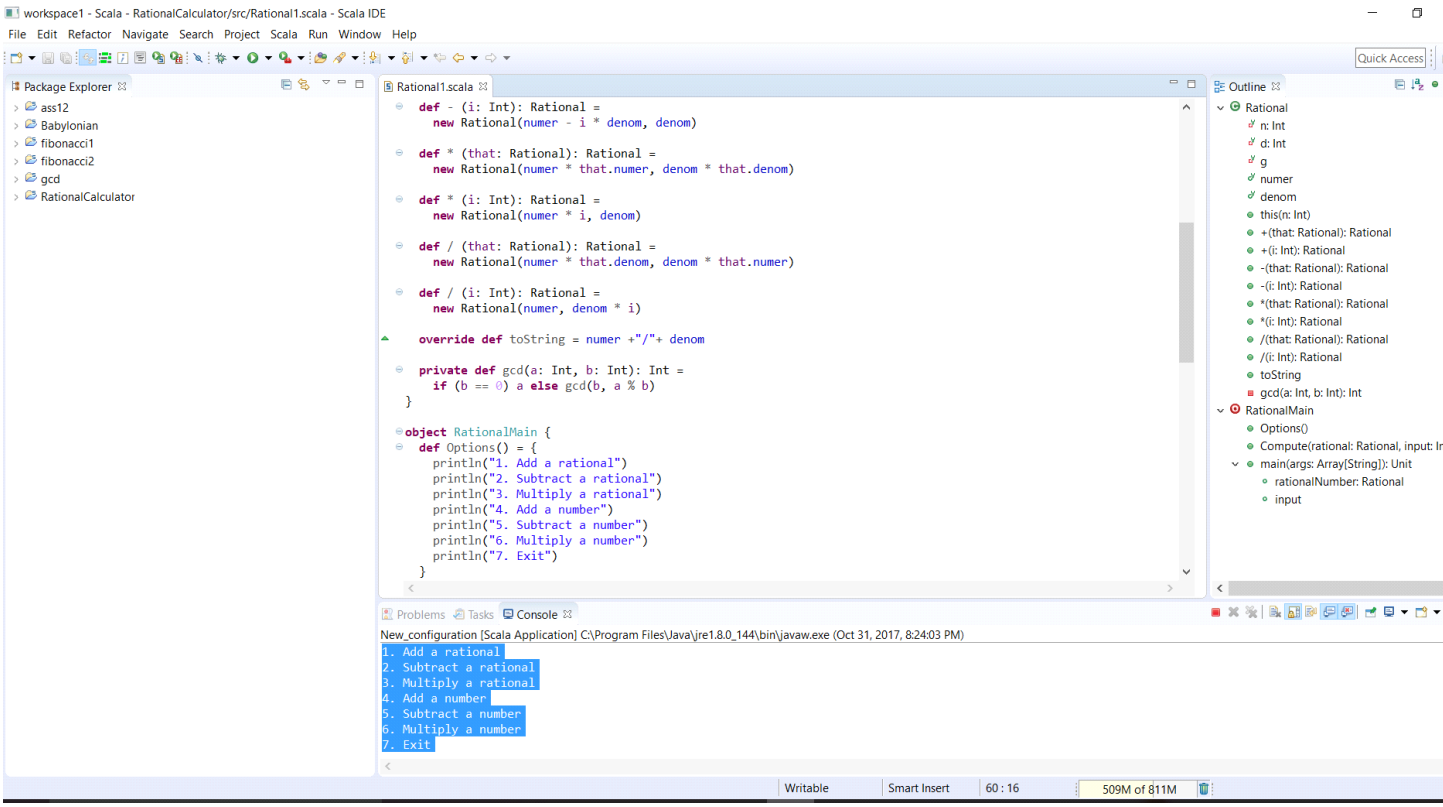
```
      val p = scala.io.StdIn.readInt()
      rational.*(new Rational(p))
    case _ =>
      rational
  }
 }

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

  var rationalNumber: Rational = new Rational(0)


  var input = 1
  do {
   Options()
   input = scala.io.StdIn.readInt()
   rationalNumber = Compute(rationalNumber, input)
   println("Output is : " + rationalNumber.toString)
  } while (input != 7)
 }
}
```

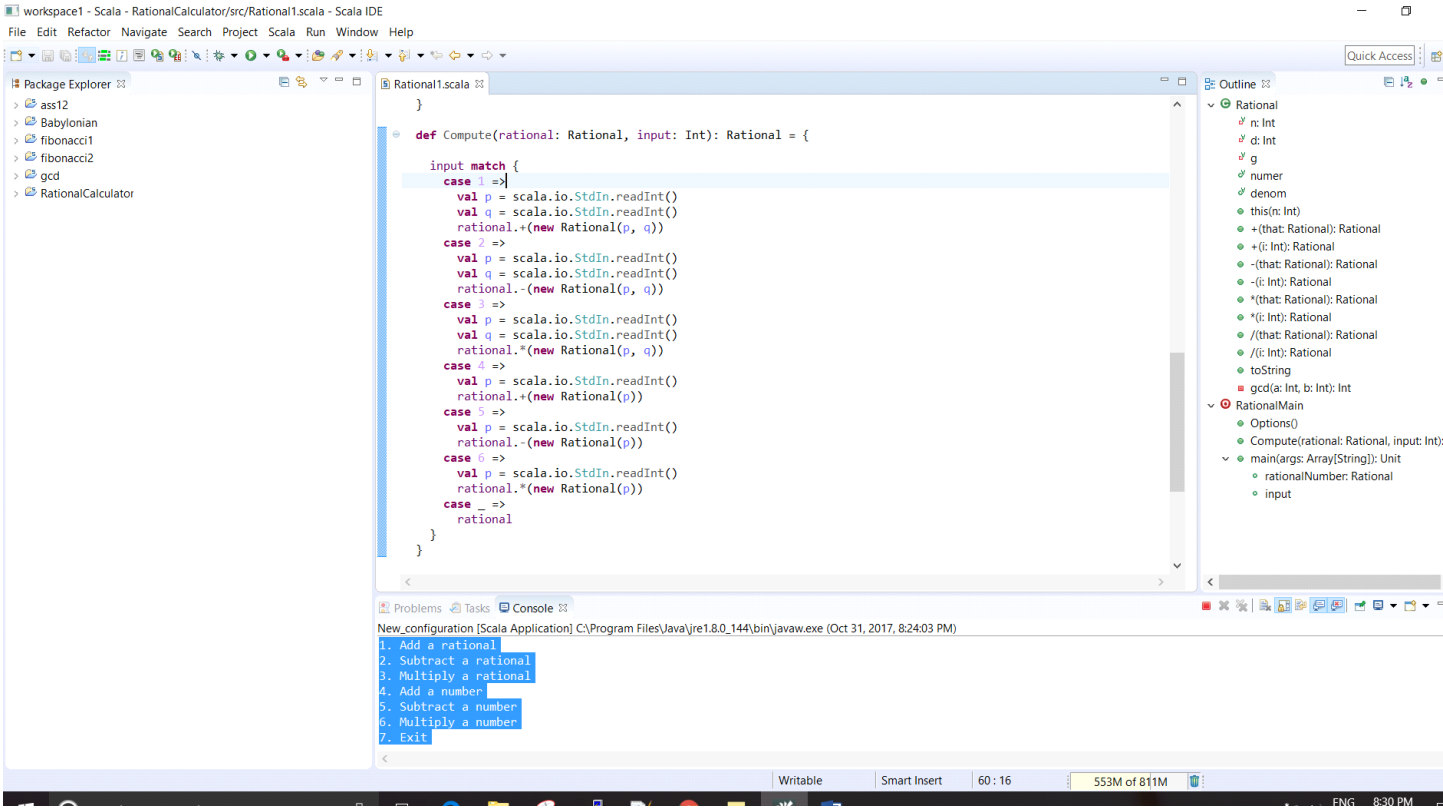
Output: -

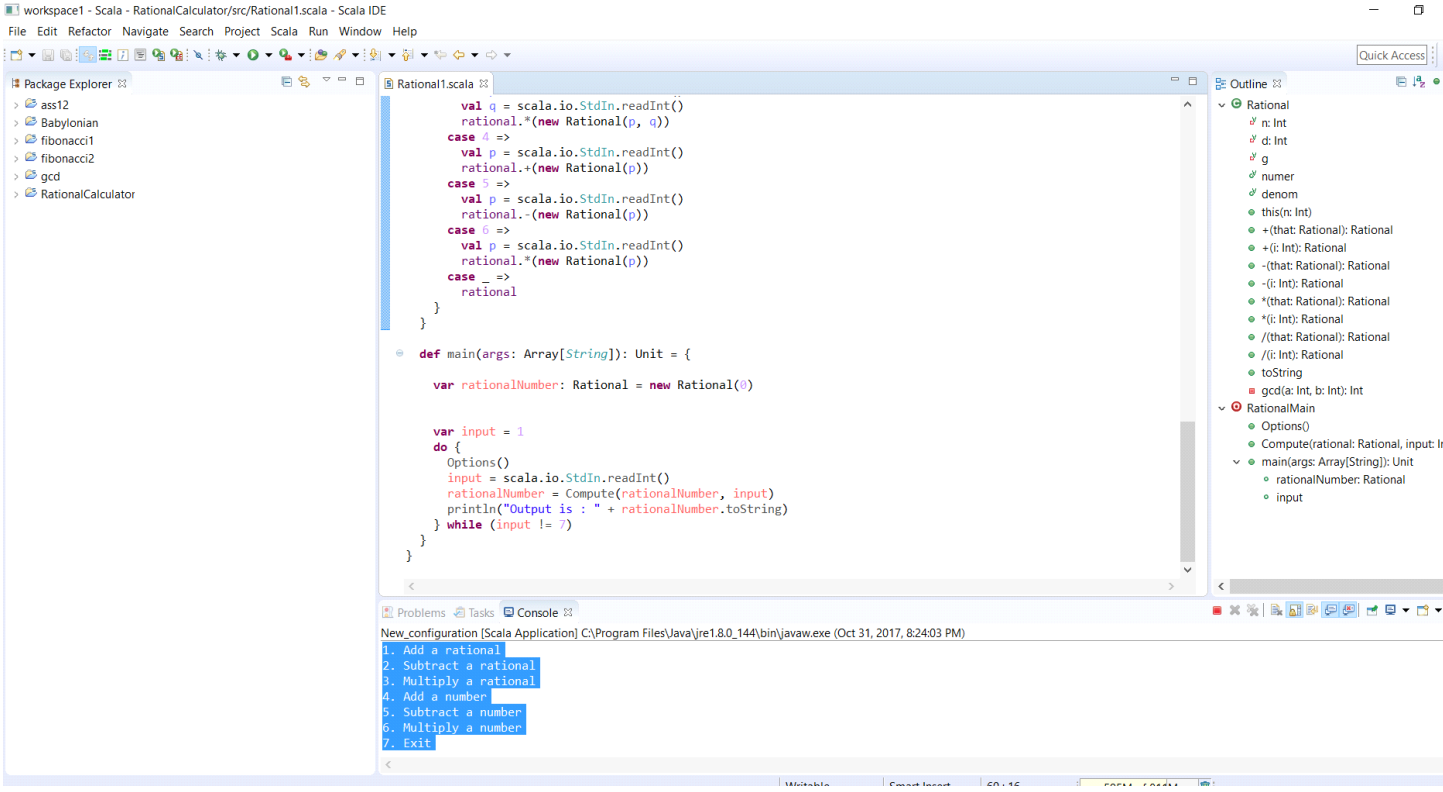
1. Add a rational
2. Subtract a rational
3. Multiply a rational
4. Add a number
5. Subtract a number
6. Multiply a number
7. Exit

Screenshot: -


workspace1 - Scala - RationalCalculator/src/Rational1.scala - Scala IDE
File Edit Refactor Navigate Search Project Scala Run Window Help
Quick Access
Package Explorer
ass12
Babylonian
fibonacci1
fibonacci2
gcd
RationalCalculator
Rational1.scala
class Rational(n: Int, d: Int) {
require(d != 0)
private val g = gcd(n.abs, d.abs)
val numer = n / g
val denom = d / g
def this(n: Int) = this(n, 1)
def + (that: Rational): Rational =
new Rational(
numer * that.denom + that.numer * denom,
denom * that.denom
)
def + (i: Int): Rational =
new Rational(numer + i * denom, denom)
def - (that: Rational): Rational =
new Rational(
numer * that.denom - that.numer * denom,
denom * that.denom
)
def - (i: Int): Rational =
new Rational(numer - i * denom, denom)
def * (that: Rational): Rational =
new Rational(numer * that.numer, denom * that.denom)
def * (i: Int): Rational =
Outline
Rational
n: Int
d: Int
g
numer
denom
this(n: Int)
+(that: Rational): Rational
+(i: Int): Rational
-(that: Rational): Rational
-(i: Int): Rational
*(that: Rational): Rational
*(i: Int): Rational
/(that: Rational): Rational
/(i: Int): Rational
toString
gcd(a: Int, b: Int): Int
RationalMain
Options()
Compute(rational: Rational, input: Int):
main(args: Array[String]): Unit
rationalNumber: Rational
input
Problems Tasks Console
New_configuration [Scala Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (Oct 31, 2017, 8:24:03 PM)
1. Add a rational
2. Subtract a rational
3. Multiply a rational
4. Add a number
5. Subtract a number
6. Multiply a number
7. Exit
Writable
Smart Insert
60 : 16
483M of 811M


workspace1 - Scala - RationalCalculator/src/Rational1.scala - Scala IDE
File Edit Refactor Navigate Search Project Scala Run Window Help
Quick Access
Package Explorer
ass12
Babylonian
fibonacci1
fibonacci2
gcd
RationalCalculator
Rational1.scala
def - (i: Int): Rational =
new Rational(numer - i * denom, denom)
def * (that: Rational): Rational =
new Rational(numer * that.numer, denom * that.denom)
def * (i: Int): Rational =
new Rational(numer * i, denom)
def / (that: Rational): Rational =
new Rational(numer * that.denom, denom * that.numer)
def / (i: Int): Rational =
new Rational(numer, denom * i)
override def toString = numer +"/"+ denom
private def gcd(a: Int, b: Int): Int =
if (b == 0) a else gcd(b, a % b)
}
object RationalMain {
def Options() = {
println("1. Add a rational")
println("2. Subtract a rational")
println("3. Multiply a rational")
println("4. Add a number")
println("5. Subtract a number")
println("6. Multiply a number")
println("7. Exit")
}
Outline
Rational
n: Int
d: Int
g
numer
denom
this(n: Int)
+(that: Rational): Rational
+(i: Int): Rational
-(that: Rational): Rational
-(i: Int): Rational
*(that: Rational): Rational
*(i: Int): Rational
/(that: Rational): Rational
/(i: Int): Rational
toString
gcd(a: Int, b: Int): Int
RationalMain
Options()
Compute(rational: Rational, input: Ir
main(args: Array[String]): Unit
rationalNumber: Rational
input
Problems Tasks Console
New_configuration [Scala Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (Oct 31, 2017, 8:24:03 PM)
1. Add a rational
2. Subtract a rational
3. Multiply a rational
4. Add a number
5. Subtract a number
6. Multiply a number
7. Exit
Writable
Smart Insert
60 : 16
509M of 811M


workspace1 - Scala - RationalCalculator/src/Rational1.scala - Scala IDE
File Edit Refactor Navigate Search Project Scala Run Window Help
Quick Access
Package Explorer
ass12
Babylonian
fibonacci1
fibonacci2
gcd
RationalCalculator
Rational1.scala
}
def Compute(rational: Rational, input: Int): Rational = {
input match {
case 1 =>
val p = scala.io.StdIn.readInt()
val q = scala.io.StdIn.readInt()
rational.+(new Rational(p, q))
case 2 =>
val p = scala.io.StdIn.readInt()
val q = scala.io.StdIn.readInt()
rational.-(new Rational(p, q))
case 3 =>
val p = scala.io.StdIn.readInt()
val q = scala.io.StdIn.readInt()
rational.*(new Rational(p, q))
case 4 =>
val p = scala.io.StdIn.readInt()
rational.+(new Rational(p))
case 5 =>
val p = scala.io.StdIn.readInt()
rational.-(new Rational(p))
case 6 =>
val p = scala.io.StdIn.readInt()
rational.*(new Rational(p))
case _ =>
rational
}
}
Outline
Rational
n: Int
d: Int
g
numer
denom
this(n: Int)
+(that: Rational): Rational
+(i: Int): Rational
-(that: Rational): Rational
-(i: Int): Rational
*(that: Rational): Rational
*(i: Int): Rational
/(that: Rational): Rational
/(i: Int): Rational
toString
gcd(a: Int, b: Int): Int
RationalMain
Options()
Compute(rational: Rational, input: Int):
main(args: Array[String]): Unit
rationalNumber: Rational
input
Problems Tasks Console
New_configuration [Scala Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (Oct 31, 2017, 8:24:03 PM)
1. Add a rational
2. Subtract a rational
3. Multiply a rational
4. Add a number
5. Subtract a number
6. Multiply a number
7. Exit
Writable
Smart Insert
60 : 16
553M of 811M
ENG
8:30 PM


workspace1 - Scala - RationalCalculator/src/Rational1.scala - Scala IDE
File Edit Refactor Navigate Search Project Scala Run Window Help
Quick Access
Package Explorer
ass12
Babylonian
fibonacci1
fibonacci2
gcd
RationalCalculator
Rational1.scala
val q = scala.io.StdIn.readInt()
rational.*(new Rational(p, q))
case 4 =>
val p = scala.io.StdIn.readInt()
rational.+(new Rational(p))
case 5 =>
val p = scala.io.StdIn.readInt()
rational.-(new Rational(p))
case 6 =>
val p = scala.io.StdIn.readInt()
rational.*(new Rational(p))
case _ =>
rational
}
}
def main(args: Array[String]): Unit = {
var rationalNumber: Rational = new Rational(0)
var input = 1
do {
Options()
input = scala.io.StdIn.readInt()
rationalNumber = Compute(rationalNumber, input)
println("Output is : " + rationalNumber.toString)
} while (input != 7)
}
}
Outline
Rational
n: Int
d: Int
g
numer
denom
this(n: Int)
+(that: Rational): Rational
+(i: Int): Rational
-(that: Rational): Rational
-(i: Int): Rational
*(that: Rational): Rational
*(i: Int): Rational
/(that: Rational): Rational
/(i: Int): Rational
toString
gcd(a: Int, b: Int): Int
RationalMain
Options()
Compute(rational: Rational, input: I
main(args: Array[String]): Unit
rationalNumber: Rational
input
Problems Tasks Console
New_configuration [Scala Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (Oct 31, 2017, 8:24:03 PM)
1. Add a rational
2. Subtract a rational
3. Multiply a rational
4. Add a number
5. Subtract a number
6. Multiply a number
7. Exit