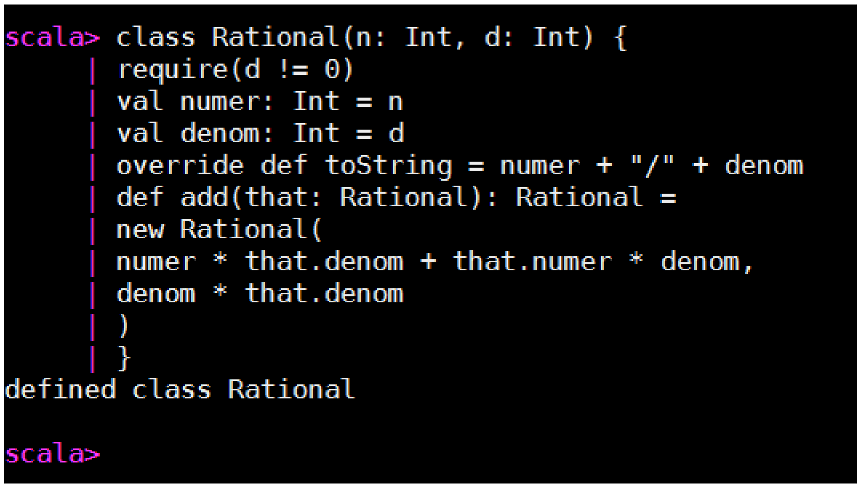
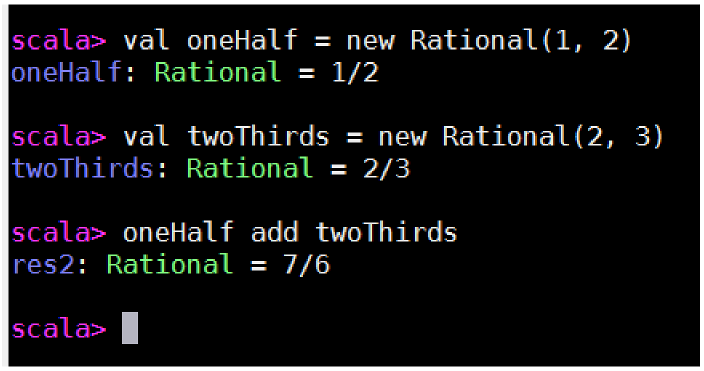
Create a calculator to work with rational numbers.
Requirements:
➢ **It should provide capability to add, subtract, divide and multiply rational Numbers**
➢ **Create a method to compute GCD (this will come in handy during operations on rational)**
Add option to work with whole numbers which are also 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.**

Basic Add for Rational Numbers:

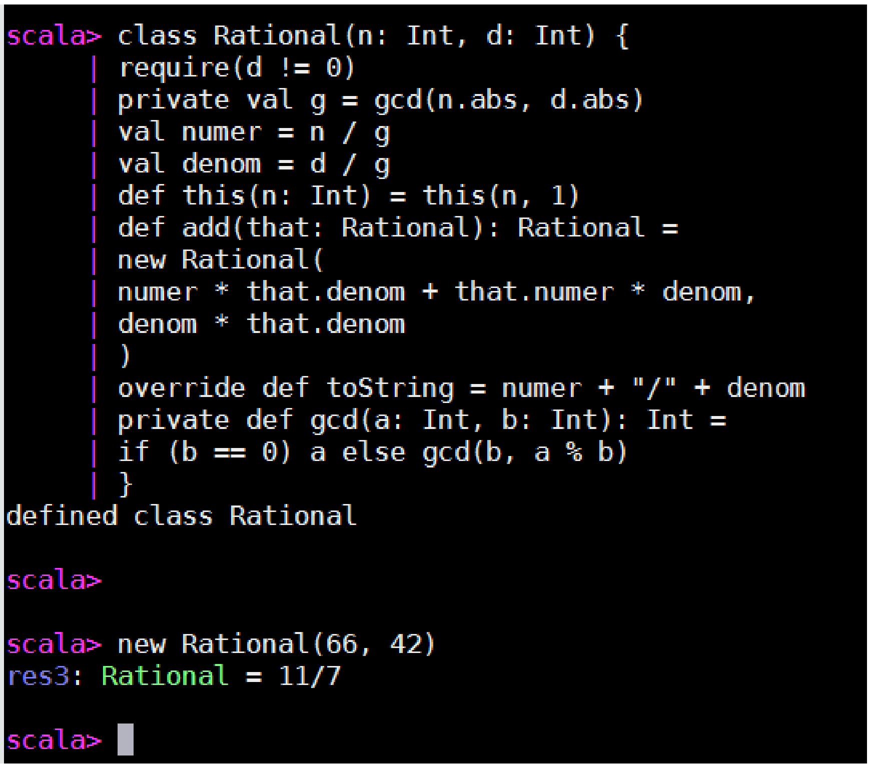


```
scala> class Rational(n: Int, d: Int) {
     | require(d != 0)
     | val numer: Int = n
     | val denom: Int = d
     | override def toString = numer + "/" + denom
     | def add(that: Rational): Rational =
     | new Rational(
     | numer * that.denom + that.numer * denom,
     | denom * that.denom
     | )
     | }
defined class Rational

scala>
```

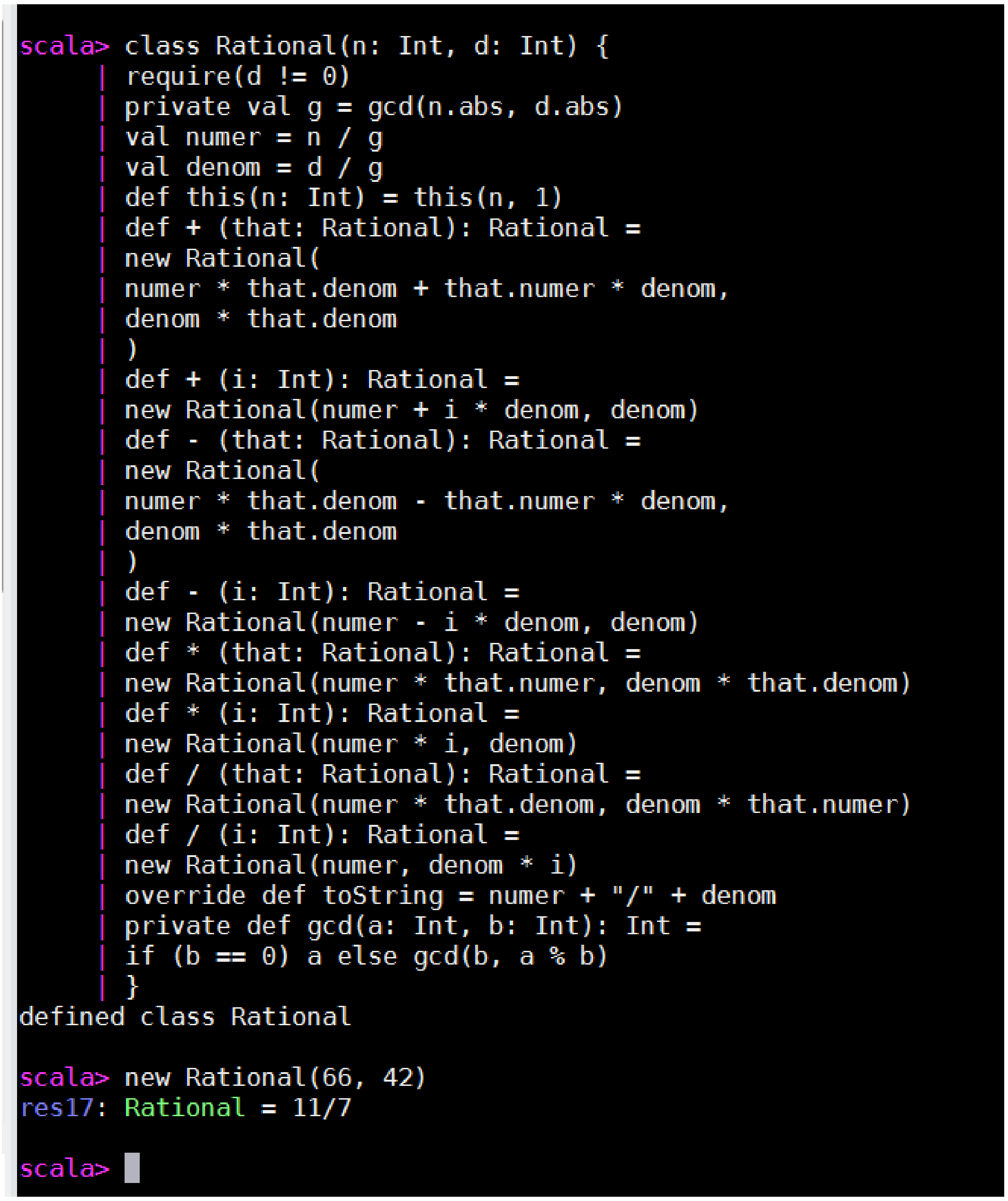


```
scala> val oneHalf = new Rational(1, 2)
oneHalf: Rational = 1/2

scala> val twoThirds = new Rational(2, 3)
twoThirds: Rational = 2/3

scala> oneHalf add twoThirds
res2: Rational = 7/6

scala>
```

GCD With Auxiliary Constructor:


```
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 add(that: Rational): Rational =
     | new Rational(
     | numer * that.denom + that.numer * denom,
     | denom * that.denom
     | )
     | override def toString = numer + "/" + denom
     | private def gcd(a: Int, b: Int): Int =
     | if (b == 0) a else gcd(b, a % b)
     | }
defined class Rational

scala>

scala> new Rational(66, 42)
res3: Rational = 11/7

scala>
```

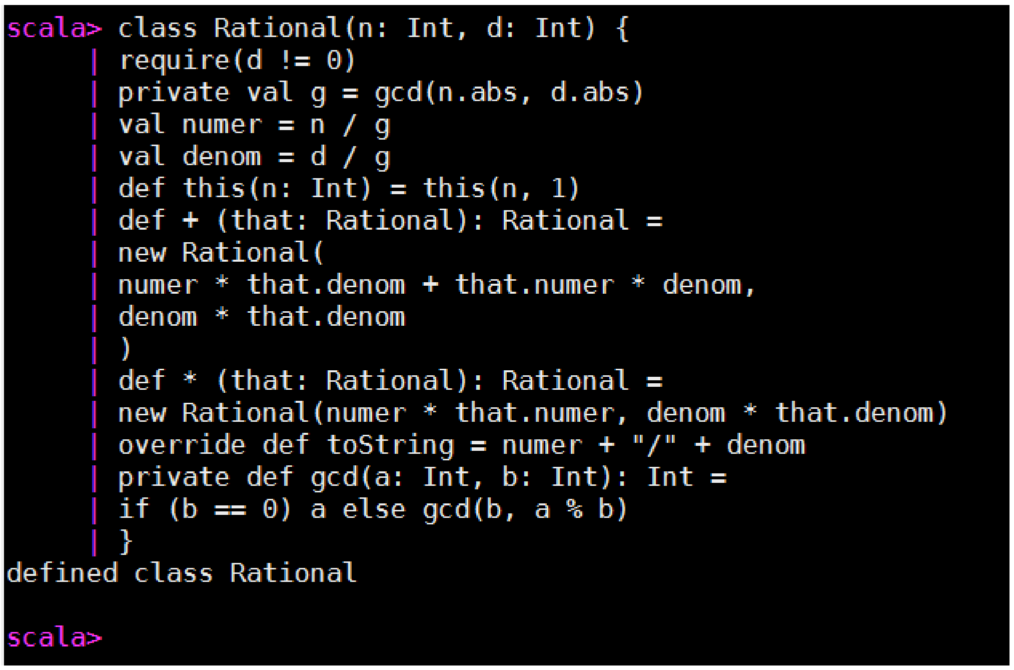
GCD with Overload:



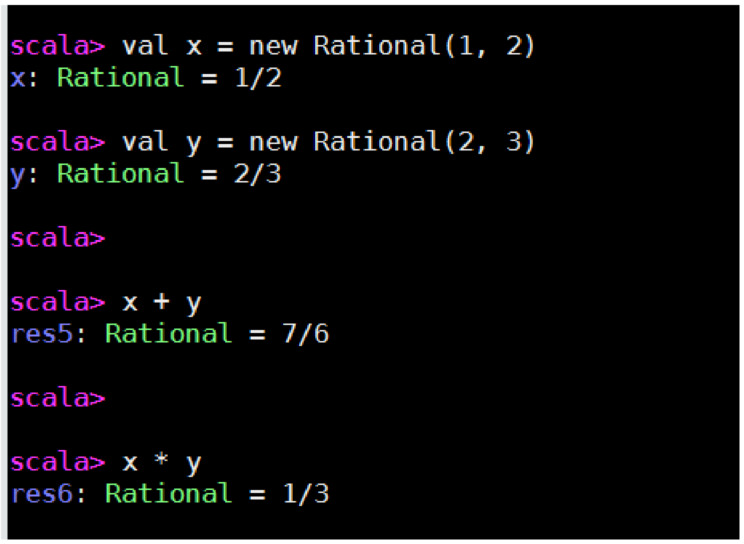
```
scala> class Rational(n: Int, d: Int) {
     | require(d != 0)
     | private val g = gcd(n.abs, d.abs)
     | val numer = n / g
     | val denom = d / g
     | def this(n: Int) = this(n, 1)
     | def + (that: Rational): Rational =
     | new Rational(
     | numer * that.denom + that.numer * denom,
     | 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 =
     | 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)
     | }
defined class Rational

scala> new Rational(66, 42)
res17: Rational = 11/7

scala>
```

ADD, SUBTRACT,MULTIPLY AND DEVIDE RATIONAL NUMBERS:

Using Auxiliary Constructor we are doing Add and Multiply:

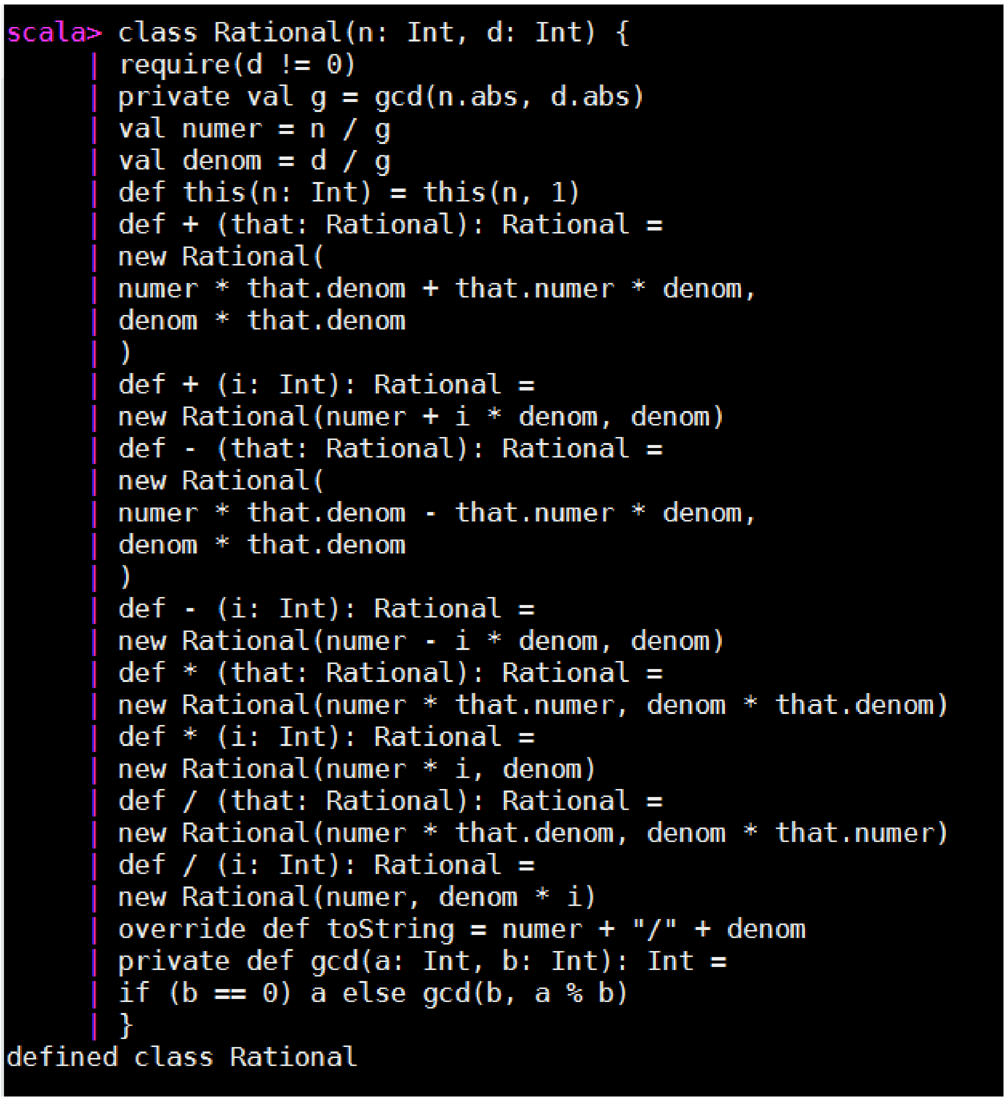


```
scala> class Rational(n: Int, d: Int) {
     | require(d != 0)
     | private val g = gcd(n.abs, d.abs)
     | val numer = n / g
     | val denom = d / g
     | def this(n: Int) = this(n, 1)
     | def + (that: Rational): Rational =
     | new Rational(
     | numer * that.denom + that.numer * denom,
     | denom * that.denom
     | )
     | def * (that: Rational): Rational =
     | new Rational(numer * that.numer, denom * that.denom)
     | override def toString = numer + "/" + denom
     | private def gcd(a: Int, b: Int): Int =
     | if (b == 0) a else gcd(b, a % b)
     | }
defined class Rational

scala>
```

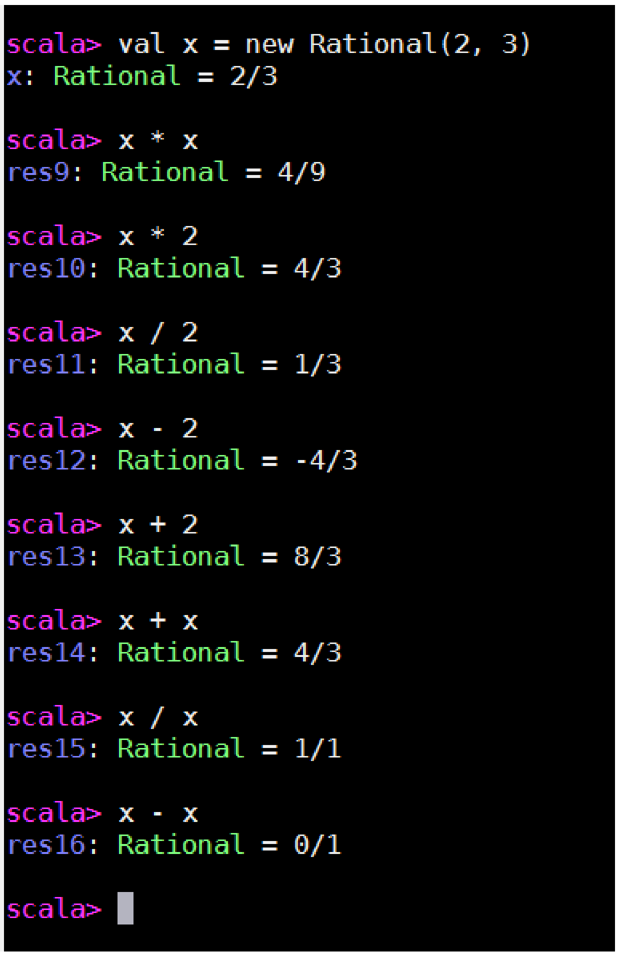


```
scala> val x = new Rational(1, 2)
x: Rational = 1/2

scala> val y = new Rational(2, 3)
y: Rational = 2/3

scala>

scala> x + y
res5: Rational = 7/6

scala>

scala> x * y
res6: Rational = 1/3
```

Using Overload and Auxiliary constructor for Add Multiple, Divide and subtract:

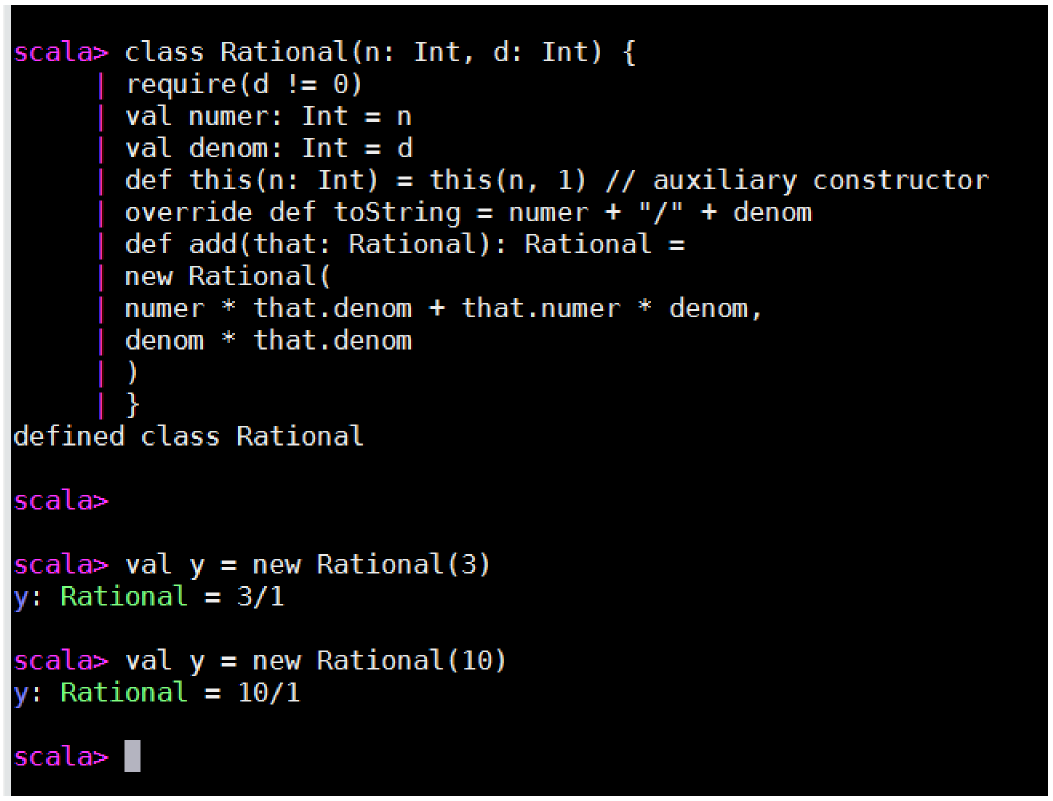


```
scala> class Rational(n: Int, d: Int) {
     | require(d != 0)
     | private val g = gcd(n.abs, d.abs)
     | val numer = n / g
     | val denom = d / g
     | def this(n: Int) = this(n, 1)
     | def + (that: Rational): Rational =
     | new Rational(
     | numer * that.denom + that.numer * denom,
     | 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 =
     | 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)
     | }
defined class Rational
```

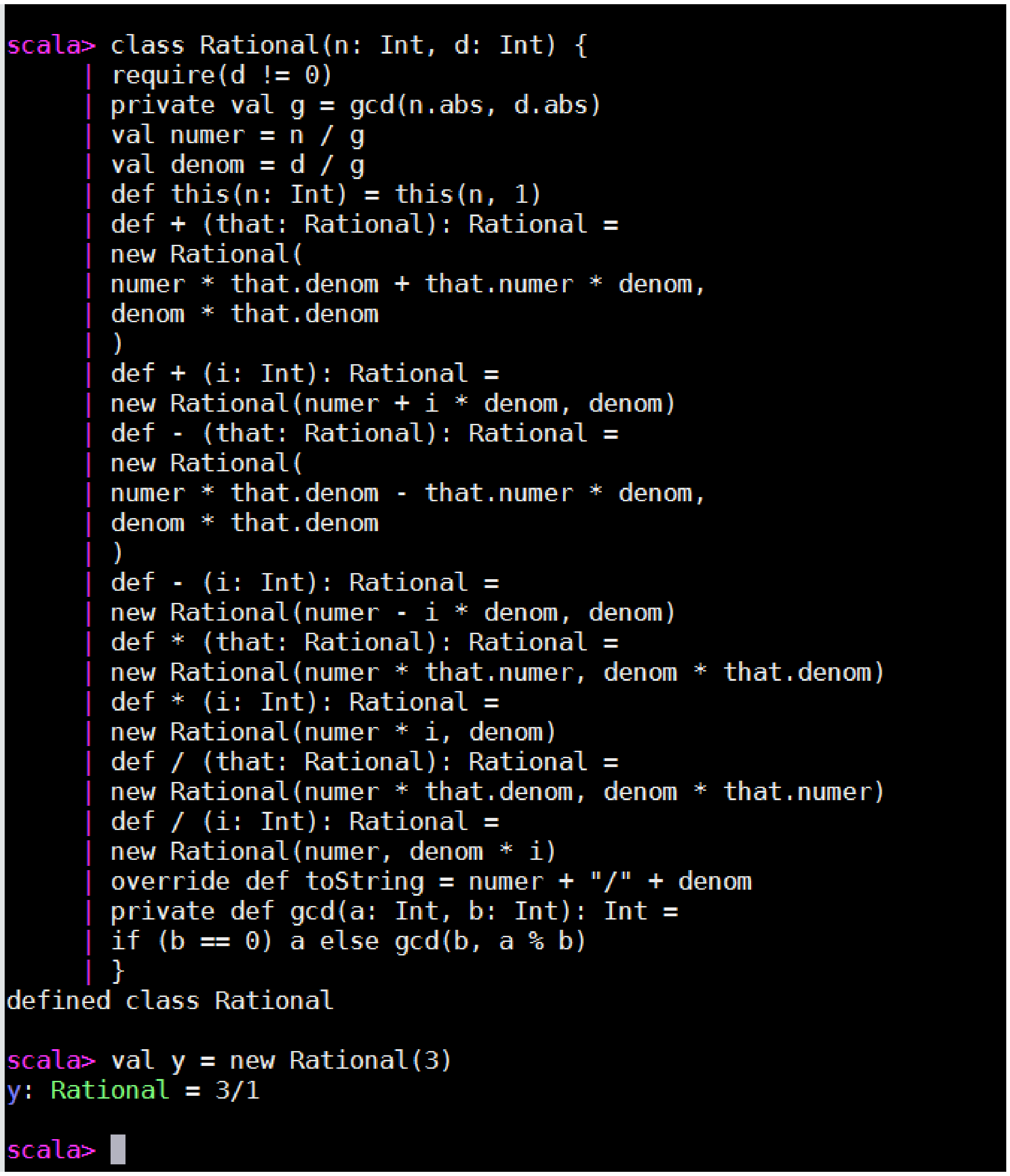


```
scala> val x = new Rational(2, 3)
x: Rational = 2/3

scala> x * x
res9: Rational = 4/9

scala> x * 2
res10: Rational = 4/3

scala> x / 2
res11: Rational = 1/3

scala> x - 2
res12: Rational = -4/3

scala> x + 2
res13: Rational = 8/3

scala> x + x
res14: Rational = 4/3

scala> x / x
res15: Rational = 1/1

scala> x - x
res16: Rational = 0/1

scala>
```

WHOLE Numbers use as Rational Numbers:



```
scala> class Rational(n: Int, d: Int) {
     | require(d != 0)
     | val numer: Int = n
     | val denom: Int = d
     | def this(n: Int) = this(n, 1) // auxiliary constructor
     | override def toString = numer + "/" + denom
     | def add(that: Rational): Rational =
     | new Rational(
     | numer * that.denom + that.numer * denom,
     | denom * that.denom
     | )
     | }
defined class Rational

scala>

scala> val y = new Rational(3)
y: Rational = 3/1

scala> val y = new Rational(10)
y: Rational = 10/1

scala>
```



```
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 =
     | 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)
     | }
defined class Rational

scala> val y = new Rational(3)
y: Rational = 3/1

scala> 
```