Task 1
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.

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

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

```
scala> var rational = new Rational(10,20)
rational: Rational = 1/2

scala> rational.+(1)
res0: Rational = 3/2

scala> rational.-(1)
res1: Rational = -1/2

scala> rational.-(3/2)
res2: Rational = -1/2

scala> rational.-(1/2)
res3: Rational = 1/2

scala> rational.*(1/2)
res4: Rational = 0/1

scala> rational.*(5/2)
res5: Rational = 1/1
```