Exposure Java	Lab 09a
The Rational Class Program I	**80, 90 & 100 Point Versions**

Assignment Purpose:

The primary purpose of this lab is to demonstrate knowledge of creating a class with object methods, instantiate multiple objects of the created class, and then call the object methods from the main program method.

Write a program with a **`Rational`** class. The purpose of the **`Rational`** class is to manipulate rational number operations. A rational number is a number that can be expressed in the form $\mathbf{A} \ / \ \mathbf{B}$ where $\mathbf{A}$ and $\mathbf{B}$ are both whole numbers (no fractions or decimals) and $\mathbf{B} \neq \mathbf{0}$.

Your main concern is to create and use the **`Rational`** class. The **`Rational`** class is quite involved and will be developed over two separate assignments. This first assignment will just get the ball rolling.

The **`main`** method is provided for you and needs to be used as shown. You are also provided with a **`getGCF`** method of the **`Rational`** class which will return the Greatest Common Factor of two integers. You will find this useful in writing other methods of the **`Rational`** class. Your mission is to complete the **`Rational`** class that is used by this program.

Lab09avst Student Version	**Do not copy this file, which is provided.**

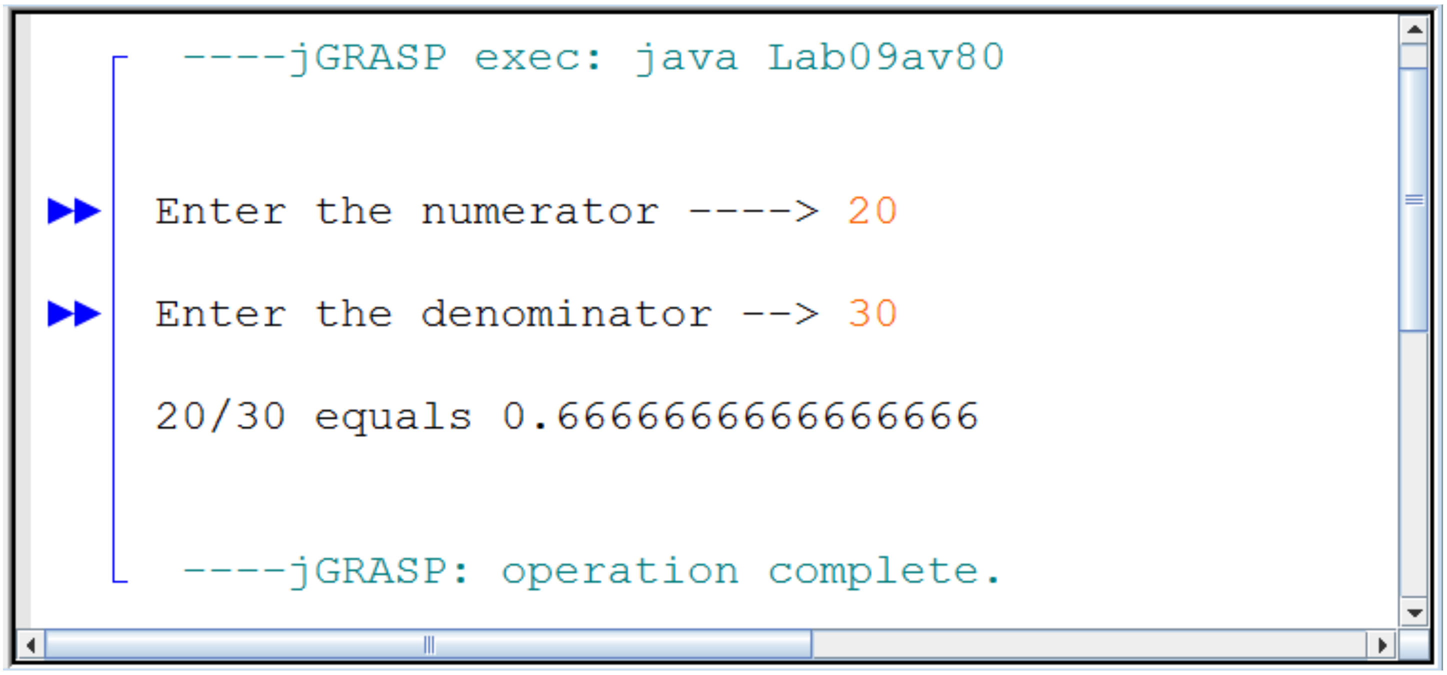
NOTE: This program will NOT compile as is. You must first write some of the methods of the <Rational> class before this program compiles.

```
// Lab09avst.java
// The Rational Class Program I
// This is the student, starting version of the TextLab05 assignment.


import java.util.Scanner;


public class Lab09avst
{
   static int num, den;   // numerator and denominator of the rational number

   public static void main (String[] args)
   {
      enterData();

      Rational r = new Rational(num,den);

      r.displayData();
   }


    public static void enterData()
   {
      Scanner input = new Scanner(System.in);
      System.out.print("\nEnter the numerator ----> ");
      num = input.nextInt();
      System.out.print("\nEnter the denominator --> ");
      den = input.nextInt();
   }
}


class Rational
{
   public void displayData()
   {
      System.out.println();
      System.out.println(getNum() + "/" + getDen() + " equals " + getDecimal());
      System.out.println();
   }

  private void getGCF(int n1,int n2)
   {
      int rem = 0;
      do
      {
         rem = n1 % n2;
         if (rem == 0)
            gcf = n2;
         else
         {
            n1 = n2;
            n2 = rem;
         }
      }
      while (rem != 0);
   }
}
```

80-Point Version Specifics

Your **`Rational`** class needs to declare three data attributes: **`num`** for numerator, **`den`** for denominator and **`gcf`** for greatest common factor. Only one constructor is required, which uses two parameters entered at the keyboard. The first parameter is the numerator and the second parameter is the denominator. For the 80-point version the **`Rational`** class requires three additional methods, which are **`getNum`**, **`getDen`**, and **`getDecimal`**. Method **`getNum`** returns the integer numerator, **`getDen`** returns the integer denominator and the **`getDecimal`** method returns a real number decimal value of the fraction. For example, if the numerator is 3 and the denominator is 4, **`getDecimal`** will return **`0.75`**.

80-Point Version Output



```
 ----jGRASP exec: java Lab09av80


Enter the numerator ----> 20

Enter the denominator --> 30

20/30 equals 0.6666666666666666


 ----jGRASP: operation complete.
```

90-Point Version Specifics

The 90-point version adds the **`getOriginal`** method. This method returns a **`String`** representation of the fraction. For example, if the numerator is 3 and the denominator is 4, **`getOriginal`** will return **`3/4`**.

Concatenation Hint:

`String` variables/values can be concatenated together.

```
Example:  "John" + "Smith" = "JohnSmith"
```

What you may not know is that other data types can be concatenated with **`Strings`** as well.

```
Example:  "John" + 19 = "John19"
```

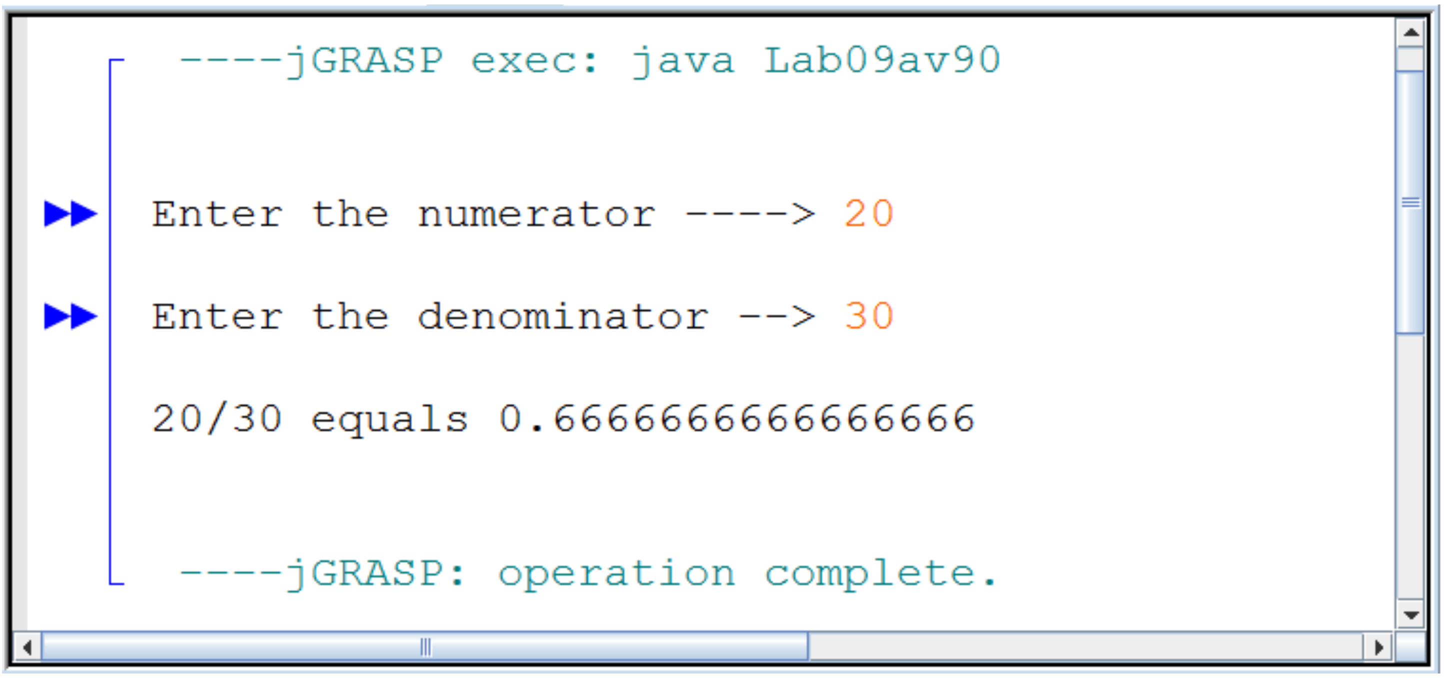
This shows an **`int`** being concatenated to the end of a **`String`**.

Even though the output of the 90-point version is identical to the 80-point version, the **`displayData`** method of the **`Rational`** class will need to be changed for the 90-point version to work properly. Now a single call to **`getOriginal`** replaces the two calls to methods **`getNum`** and **`getDen`**.

90-Point Version **`displayData`** Method

```
public void displayData()
{
     System.out.println();
     System.out.println(getOriginal() + " equals " + getDecimal());
     System.out.println();
}
```

<u>90-Point Version Output</u>



```
 ----jGRASP exec: java Lab09av90


Enter the numerator ----> 20

Enter the denominator --> 30

20/30 equals 0.6666666666666666


 ----jGRASP: operation complete.
```

100-Point Version Specifics

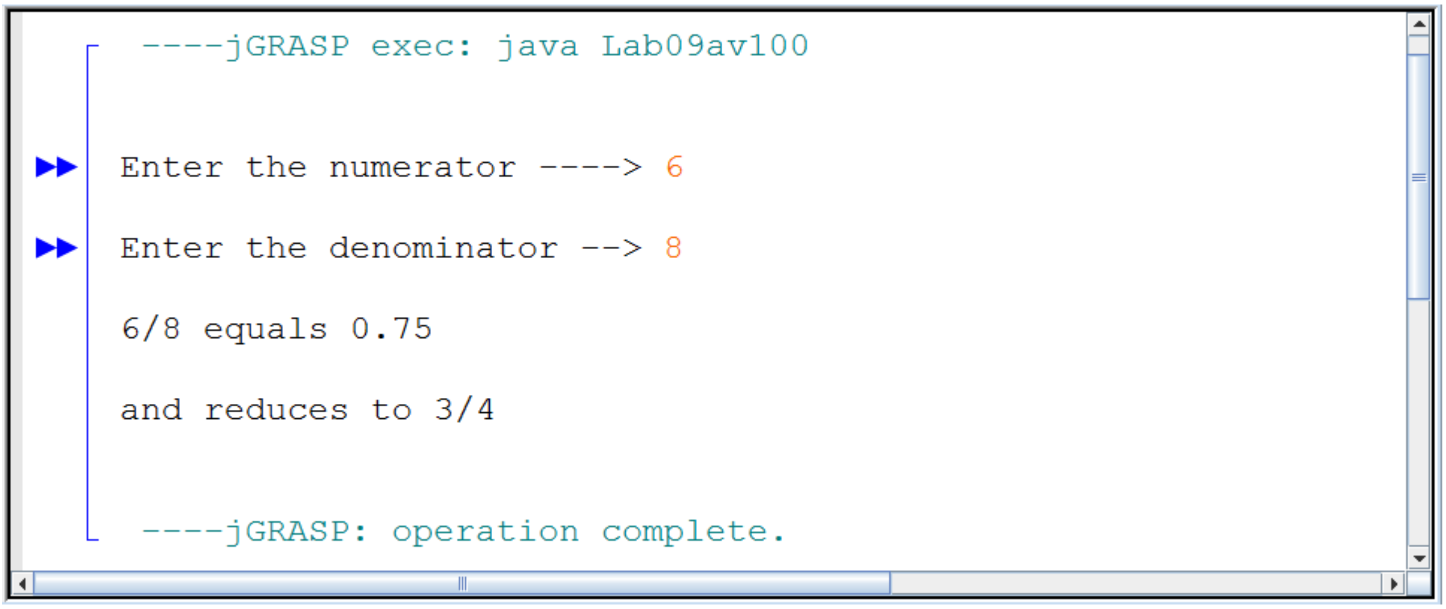
The 100-point version adds the **`reduce`** and **`getReduced`** methods as well as **`firstNum`** and **`firstDen`** variable attributes. Also change attribute **`num`** to **`reducedNum`** and attribute **`den`** to **`reducedDen`**. The constructor needs to be changed. This version of the lab assignment reduces the fraction, if possible. The output displays something like 15/20 reduces to 3/4. Without additional variables, the original values of the numerator and denominator will be lost. You need to achieve the following sequence. The **`Rational`** constructor initializes all variables and then reduces the fraction. The **`reduce`** method needs **`getGCF`** to insure maximum reduction.

As with the 90-point version, the **`displayData`** method of the **`Rational`** class will need to be changed again for this program to work properly.

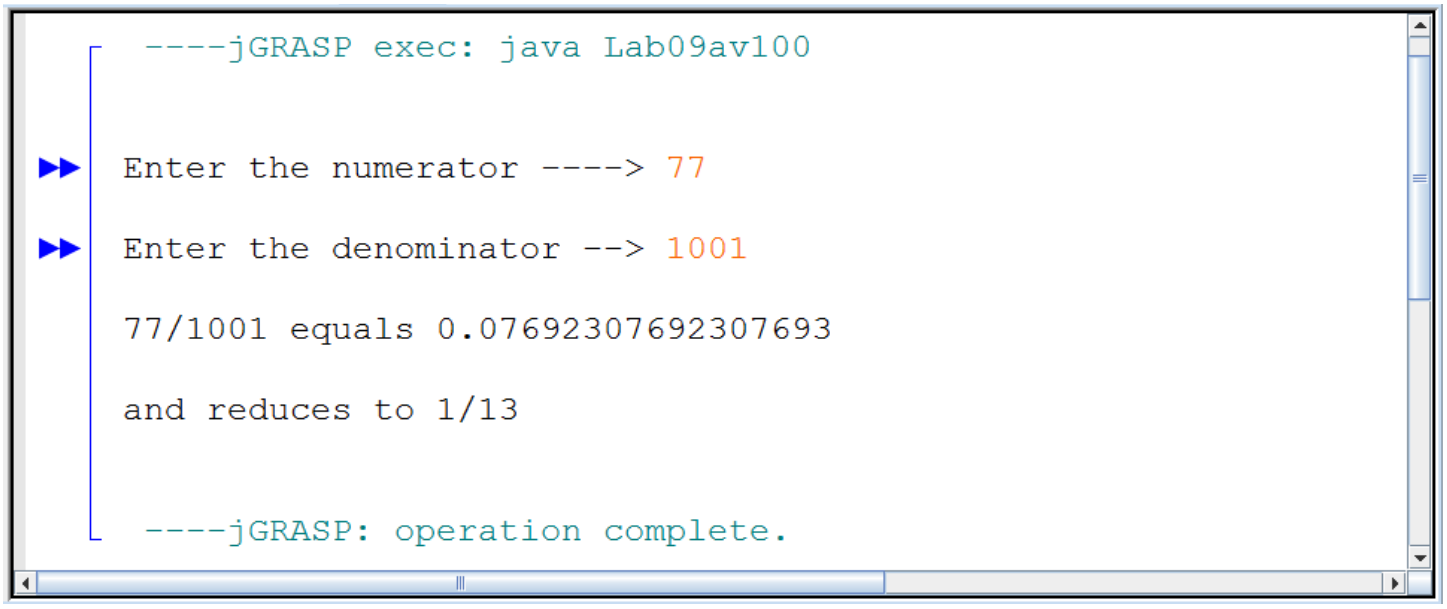
100-Point Version **displayData** Method

```
public void displayData()
{
  System.out.println();
  System.out.println(getOriginal() + " equals " + getDecimal());
  System.out.println();
  System.out.println("and reduces to " + getReduced());
  System.out.println();
}
```

100-Point Version Outputs



```
 ----jGRASP exec: java Lab09av100

Enter the numerator ----> 6

Enter the denominator --> 8

6/8 equals 0.75

and reduces to 3/4


 ----jGRASP: operation complete.
```



```
 ----jGRASP exec: java Lab09av100

Enter the numerator ----> 77

Enter the denominator --> 1001

77/1001 equals 0.07692307692307693

and reduces to 1/13


 ----jGRASP: operation complete.
```