|  |  |
| --- | --- |
| **AP Computer Science** | **TextLab04 Java Assignment** |
| **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 **A / B** where **A** and **B** are both whole numbers (no fractions or decimals) and **B ≠ 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.

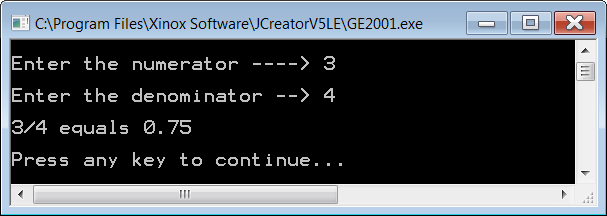
The **main** method is provided for you and needs to be used as shown. You are also provided with a **getGCF** method of the **Rational** class which will return the Greatest Common Factor of 2 integers. You will find this useful in writing other methods of the **Rational** class. Your mission is to complete the **Rational** class that is used by this program.

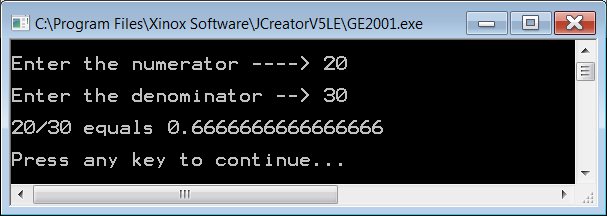
|  |  |
| --- | --- |
| **TextLab04st Student Version** | **Do not copy this file, which is provided.** |
| **NOTE: This program will NOT compile as is. You must first write some of the methods of the Rational class.** | |
| // TextLab04st.java  // The Rational Class Program I  // This is the student, starting version of the TextLab04 assignment.  import java.util.Scanner;  public class TextLab04st  {  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  {  // Rational  // getNum  // getDen  // getDecimal  // getRational  // getOriginal  // reduce  public void displayData()  {  System.out.println();  System.out.println(getNum() + "/" + getDen() + " equals " + getDecimal());  System.out.println();  }  private int getGCF(int n1,int n2)  {  int rem = 0;  int gcf = 0;  do  {  rem = n1 % n2;  if (rem == 0)  gcf = n2;  else  {  n1 = n2;  n2 = rem;  }  }  while (rem != 0);  return gcf;  }  } | |

**80 Point Version Specifics**

Your **Rational** class needs to declare two data attributes: **num** for numerator and **den** for denominator. 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. 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 (and 90) Point Version Outputs**





**90 Point Version Specifics**

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

|  |
| --- |
| ***Concatenation Hint:***  You probably know that **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 **String**s 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 (see previous page), the **displayData** method of the **Rational** class will need to be changed for the 90 point version to work properly. (See below.) Now a single call to **getRational** replaces the 2 calls to methods **getNum** and **getDen**.

|  |
| --- |
| 90 Point Version **displayData** Method |
| **public void displayData()**  **{**  **System.out.println();**  **System.out.println(getRational() + " equals " + getDecimal());**  **System.out.println();**  **}** |

**100 Point Version Specifics**

The 100-point version adds the **getOriginal** and **reduce** methods as well as **firstNum** and **firstDen** variable attributes. The constructor also 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. (See below.)

|  |
| --- |
| 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 " + getRational());**  **System.out.println();**  **}** |

**100 Point Version Outputs**

