

***C# Basics***

**Lab Guides**

|  |  |
| --- | --- |
| Document Code | 25e-BM/HR/HDCV/FSOFT |
| Version | 1.1 |
| Effective Date | 20/11/2012 |

**Hanoi, 06/2019**

RECORD OF CHANGES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Effective Date | Change Description | Reason | Reviewer | Approver |
|  | 01/Oct/2018 | Create new | Draft |  |  |
|  | 01/Jun/2019 | Update template | Fsoft template |  |  |
| 3 | 15/Apr/2019 | Review content | Review | TuTB |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Contents

[Lab 9: Work with Abstraction and Encapsulation 4](#_Toc17708878)

[Objectives: 4](#_Toc17708879)

[Prerequisites: 4](#_Toc17708880)

[Problem Description: 4](#_Toc17708881)

[Guidelines: 4](#_Toc17708882)

[Step 1: Create project named **Fraction** in Visual Studio 4](#_Toc17708883)

[Step 2: Add class named **Fraction** then declare Fraction’s properties: 4](#_Toc17708884)

[Step 3: Create 3 constructors 4](#_Toc17708885)

[Step 4: Create **Print** method to returns the description of the fraction as a String 5](#_Toc17708886)

[Step 5: Create method 6](#_Toc17708887)

[Step 6: Create method GetNormalForm to normally a fraction 7](#_Toc17708888)

[Step 7: Create method Add, Subtract to implement add, subtract 2 fractions 7](#_Toc17708889)

[Step 8: In Program.cs file, write code to initial objects in various ways 8](#_Toc17708890)

|  |  |
| --- | --- |
|  | **CODE: Net.S.L009**  **TYPE: SHORT**  **LOC: 100**  **DURATION: 30 MINUTES** |

# Lab 9: Work with Abstraction and Encapsulation

Objectives:

* Understand the application of Abstraction and Encapsulation in C #.

Prerequisites:

* Download and installs Visual Studio (included .net Framework)

Problem Description:

* Create a Fraction class with 2 properties: Numerator and Denominator
* Create 3 constructors for Fraction class
  + No parameter: assign default value for Numerator = 0, Denominator = 1
  + With 1 parameter: assign passed value to Numerator, assign Denominator as 1
  + With 2 parameters: assign passed values to Numerator and Denominator
* Create method GetGreatestCommonDivisor(int, int) to get Greatest Common Divisor of 2 numbers
* Create method GetLeastCommonMultiple(int, int) to get Least Common Multiple of 2 numbers
* Create method GetNormalForm to normally fraction
* Create method Add, Subtract to implement add, subtract 2 fractions

Guidelines:

### Step 1: Create project named **Fraction** in Visual Studio

### Step 2: Add class named **Fraction** then declare Fraction’s properties:

public class Fraction

{

public int Numerator { get; set; }

public int Denominator { get; set; }

}

### Step 3: Create 3 constructors

public Fraction()

{

Numerator = 0;

Denominator = 1;

}

public Fraction(int integer)

{

Numerator = integer;

Denominator = 1;

}

public Fraction(int numerator, int denominator)

{

if (denominator >= 0)

{

this.Numerator = numerator;

this.Denominator = denominator;

}

else

{

this.Numerator = -numerator;

this.Denominator = -denominator;

}

}

### Step 4: Create **Print** method to returns the description of the fraction as a String

public void Print()

{

if (Denominator == 0)

{

Console.WriteLine("Error when print a fraction with denomination equal 0");

}

else if (Denominator == 1)

{

Console.WriteLine(string.Format("{0}", this.Numerator));

}

else

{

Console.WriteLine(string.Format("{0}/{1}", this.Numerator, this.Denominator));

} }

### Step 5: Create method

Create method GetGreatestCommonDivisor(int, int) to get Greatest Common Divisor of 2 numbers

private int GetGreatestCommonDivisor(int firstNumber, int secondNumber)

{

if (firstNumber == 0 || secondNumber == 0)

{

return 1;

}

var uFirstNumber = Math.Abs(firstNumber);

var uSecondNumber = Math.Abs(secondNumber);

while (uFirstNumber != uSecondNumber)

{

if (uFirstNumber > uSecondNumber)

{

uFirstNumber = uFirstNumber - uSecondNumber;

}

else

{

uSecondNumber = uSecondNumber - uFirstNumber;

}

}

return uSecondNumber;

}

Create method GetLeastCommonMultiple(int, int) to get Least Common Multiple of 2 numbers

private int GetLeastCommonMultiple(int firstNumber, int secondNumber)

{

var greatestCommonDivisor = this.GetGreatestCommonDivisor(firstNumber, secondNumber);

var leastCommonMultiple = (firstNumber \* secondNumber) / greatestCommonDivisor;

return Math.Abs(leastCommonMultiple);

}

### Step 6: Create method GetNormalForm to normally a fraction

public void GetNormalForm()

{

var greatestCommonDivisor = this.GetGreatestCommonDivisor(this.Numerator, this.Denominator);

if (greatestCommonDivisor > 1)

{

this.Numerator = this.Numerator / greatestCommonDivisor;

this.Denominator = this.Denominator / greatestCommonDivisor;

}

if (this.Denominator < 0)

{

this.Numerator = -this.Numerator;

this.Denominator = -this.Denominator;

}

}

### Step 7: Create method Add, Subtract to implement add, subtract 2 fractions

public void Add(Fraction secondFraction)

{

var numerator = this.Numerator \* secondFraction.Denominator

+ this.Denominator \* secondFraction.Numerator;

var denominator = this.Denominator \* secondFraction.Denominator;

this.Numerator = numerator;

this.Denominator = denominator;

this.GetNormalForm();

}

public void Subtract(Fraction secondFraction)

{

var numerator = this.Numerator \* secondFraction.Denominator

- this.Denominator \* secondFraction.Numerator;

var denominator = this.Denominator \* secondFraction.Denominator;

this.Numerator = numerator;

this.Denominator = denominator;

this.GetNormalForm();

}

### Step 8: In Program.cs file, write code to initial objects in various ways

static void Main(string[] args)

{

Console.WriteLine("Init fraction: 1/2");

Fraction fraction = new Fraction(1, 2);

Console.WriteLine("Add new value 1/3");

Console.Write("The fraction value is: ");

fraction.Add(new Fraction(1, 3));

fraction.Print();

Console.WriteLine("Subtract value 1/4");

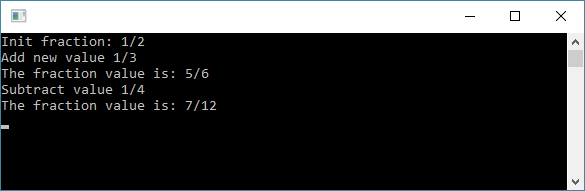
Console.Write("The fraction value is: ");

fraction.Subtract(new Fraction(1, 4));

fraction.Print();

Console.ReadKey(); }

Outputs

****