|  |
| --- |
| **Computer Engineering Department - ITU** |
| **CE101L: Object Oriented Programming Lab** |

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 26/05/2022** |
| **Teaching Assistant: Aqsa Khalid** | **Semester: Spring 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 11B. Problem Based Learning through Objects & Classes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
| NIMRA MAQBOOL | BSCE21012 |  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Objective**

The objective of this lab is to observe the basic knowledge of programming classes in C++.

## **Equipment and Component**

|  |  |  |
| --- | --- | --- |
| **Component Description** | **Value** | **Quantity** |
| Computer | Available in lab | 1 |

## **Conduct of Lab**

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

## **Theory and Background**

**Open-ended problem** is a problem that has several or many correct answers, and several ways to the correct answer(s). The Open-Ended Approach provides students with "experience in finding something new in the process"(Shimada 1997). It is basically facilitating the development of creative problem solving skills.

Diagram

Description automatically generated

Figure 1: \*What is Open Ended Problem Solving??

**Lab Task**

**Task A: [Marks: 40]**

Define a class called Fraction. This class is used to represent a ratio of two integers. Include an additional method, equals, that takes as input another Fraction and returns true if the two fractions are identical and false if they are not. Concepts that you should use in your program.

* Menu Driven
* Default & Parameterized constructor for initializing two integers to zero and should be private and accessible through setter/getter methods.
* Overload Stream in and Stream out operator to take input two integers and display their ratio respectively.
* Overload Equal == Operator for finding two fractions identical or not.

|  |
| --- |
| **Function.h:**  // // Created by Lenovo on 5/26/2022. //  #ifndef INC\_2022\_SPRING\_CE\_OOP\_WEEK11\_LABTASK\_B\_BSCE21012\_FRACTIONS\_H #define INC\_2022\_SPRING\_CE\_OOP\_WEEK11\_LABTASK\_B\_BSCE21012\_FRACTIONS\_H #include <iostream> using namespace std;   class Fractions { private:  int num1;  int num2; //declaring  double ratio;  public:   friend Fractions operator<<(ostream &os, const Fractions &F); //declaring friend function  friend Fractions operator>>(istream &is, Fractions &F);  Fractions(){  num1=0; //declaring them equal to zero  num2=0;  ratio=0.00;  }  Fractions(int n1,int n2){  cout<<"parametrized constructor is called.."<<endl;  num1=n1; //equaling  num2=n2;  }  void setNumber1(int &number1){  num1=number1; //equaling  }  void setNumber(int &number2){  num2=number2; //equaling  }   int getNumber1(){  return num1; //equaling  }  int getNumber2(){  return num2; //equaling  }  double getRatio(){  return ratio; //equaling  }   bool operator==(Fractions &F){  F.ratio=(F.num1/F.num2); //taking ratio  ratio=(num1/num2);  if(ratio==F.ratio){ //checking that if ratios are equal or not  cout<<"\ntrue"<<endl;  return true;  }  else{  return false;  }    } }; Fractions operator<<(ostream &os, const Fractions &F){  cout<<"ratio = ";  os<<F.num1<<":"<<F.num2<<endl; //displaying  return F; } Fractions operator>>(istream &is,Fractions &F){  cout<<"ENTER NUMBER 1 = ";  is>>F.num1; //taking input  cout<<"ENTER NUMBER 2 = ";  is>>F.num2;  return F; //returning }  **Main.cpp:**  #include <iostream> #include "Fractions.h" using namespace std;  int main() {  int opt;  do{  cout<<"choose option."<<endl; //taking option  cout<<"1.take input from constructor."<<endl;  cout<<"2.take input from getter setter."<<endl;  cout<<"3.take input from stream in and stream out operator."<<endl;  cout<<"4.exit.."<<endl;  cin>>opt;  if(opt==1){  Fractions f;  int n1; //declaring  int n2;  cout<<"enter num 1 = ";  cin>>n1;  cout<<"enter num 2 = ";  cin>>n2; //taking value  Fractions F(n1,n2);  cout<<F;  cout<<"enter num 1 = ";  cin>>n1;  cout<<"enter num 2 = ";  cin>>n2;  Fractions F1(n1,n2); //calling  cout<<F1;  if(F==F1){ //checking  cout<<"BOTH RATIOS ARE EQUAL.."<<endl;  }  else{  cout<<"BOTH FRACTIONS ARE NOT EQUAL.."<<endl;  }  }  if(opt==2){  Fractions F;  int number1; //declaring  int number2;  cout<<"enter num 1 = ";  cin>>number1; //taking input  F.setNumber1(number1);  cout<<"enter num 2 = ";  cin>>number2;  F.setNumber(number2);  F.getNumber1();  F.getNumber2();  cout<<F;  Fractions F1;  cout<<"enter num 1 = ";  cin>>number1; //taking input  cout<<"enter num 2 = ";  cin>>number2;  F1.setNumber1(number1);  F1.setNumber(number2);  F1.getNumber1(); //calling  F1.getNumber2();  cout<<F1;  F1.getRatio(); //calling  F.getRatio();  if(F==F1){ //checking  cout<<"BOTH RATIOS ARE EQUAL.."<<endl;  }  else{  cout<<"BOTH FRACTIONS ARE NOT EQUAL.."<<endl;  }  }  if(opt==3){  Fractions F;  cin>>F;  cout<<F; //calling  Fractions F1; //making an object  cin>>F1; //taking input  cout<<F1;  if(F==F1){ //checking  cout<<"BOTH RATIOS ARE EQUAL.."<<endl;  }  else{  cout<<"BOTH FRACTIONS ARE NOT EQUAL.."<<endl;  }  }  if(opt==4){  cout<<"YOU CHOOSE TO EXIT.."<<endl;  exit(4);  }  }while(opt>=1 && opt<=4);    return 0; }  **Output:**  **Text  Description automatically generatedText  Description automatically generated** |

#### **Assessment Rubric for Lab**

**Method for assessment:**

Lab reports and instructor observation during lab sessions. Outcome assessed:

a. Ability to conduct experiments, as well as to analyze and interpret data (P) b. Ability to function on multi-disciplinary teams (A)

c. Ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (P)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Performance metric** | **Task** | **CLO** | **Description** | **Max marks** | **Exceeds expectation** | **Meets expectation** | **Does not meet expectation** | **Obtained marks** |
| 1. Realization of experiment (a) | 1 | 1 | Functionality | 40 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed (35-40) | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed (20-34) | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed (0-19) |  |
| 2. Teamwork (b) | 1 | 3 | Group Performance | 5 | Actively engages and cooperates with other group member(s) in effective manner (4-5) | Cooperates with other group member(s) in a reasonable manner but conduct can be improved (2-3) | Distracts or discourages other group members from conducting the experiment (0-1) |  |
| 3. Conducting experiment (a, c) | 1 | 1 | On Spot Changes | 10 | Able to make changes (8-10) | Partially able to make changes (5-7) | Unable to make changes (0-4) |  |
| 1 | 1 | Viva | 10 | Answered all questions (8-10) | Few incorrect answers (5-7) | Unable to answer all questions (0-4) |  |
| 4. Laboratory safety and disciplinary rules (a) | 1 | 3 | Code commenting | 5 | Comments are added and does help the reader to understand the code (4-5) | Comments are added and does not help the reader to understand the code (2-3) | Comments are not added (0-1) |  |
| 5. Data collection (c) | 1 | 3 | Code Structure | 5 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap (4-5) | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables (2-3) | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy (0-1) |  |
| 6. Data analysis (a, c) | 1 | 4 | Algorithm | 20 | Solution is efficient, easy to understand, and maintain (15-20) | A logical solution that is easy to follow but it is not the most efficient (6-14) | A difficult and inefficient solution (0-5) |  |
| 7. Computer use (c) | 1 | 2 | Documentation & GitHub Submissions | 5 | Timely (4-5) | Late (2-3) | Not done (0-1) |  |
|  | Max Marks (total): | | | 100 | Obtained Marks (total): | | |  |

Lab Engineer Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_