**Heaven’s Light is Our Guide**



**Rajshahi University of Engineering and Technology**

**Department of Computer Science and Engineering**

**Course No:** CSE.1204

**Course Title:** Sessional based on CSE.1203 (Object Oriented Programming)

**Lab Report No:** 09

**Lab Report On:** Interface in Java.

**Submitted By** **Submitted To**

Md. Ariful Islam Md. Asifur Rahman

Roll No: 1803046 Lecturer

Section: A Dept. of CSE,RUET

Department: CSE

**Problem No:** 01

**Problem Statement:** Implementation of **Interface** in Java.

|  |
| --- |
| multi() |

**interface** add **interface** multi **interface** grater **interface** equal

|  |
| --- |
| add() |

|  |
| --- |
| determine\_greater() |

|  |
| --- |
| determine\_equal() |

**interface** arithmetic\_operation **interface** logical\_operation

|  |
| --- |
|  |

|  |
| --- |
|  |

**interface** operation

|  |
| --- |
|  |

**class** math

|  |
| --- |
| float a;  float b; |

**class** math

|  |
| --- |
|  |

**Theory**

**Interface** is like the blue print of a class. Like a class, an interface can have methods and variables, but the methods declared in an interface are by default abstract (only method signature, no body).

* Interfaces specify what a class must do and not how.
* If a class implements an interface and does not provide method bodies for all functions specified in the interface, then the class must be declared abstract.
* It is used to achieve multiple inheritance.
* It is used to achieve loose coupling.

Syntax of interface:

**interface <interface\_name> {**

**// declare constant fields**

**//declare methods that are abstract by default**

**}**

To declare an interface,**interface** keyword is used. It is used to provide total abstraction. That means all the methods in an interface are declared with an empty body and are public and all fields are public, static and final by default. A class that implement interface must implement all the methods declared in the interface. To implement interface use **implements** keyword.

A class can implement one or more interfaces. Syntax:

**class <class\_name> implements <interface\_X>,<interface\_Y> {**

**//Body of class with overriding the methods of interfaces**

**}**

An interface can extends another interface or interfaces. Syntax:

**interface <interface\_X> extends <interface\_Y> , <interface\_Z>**

**{**

**//Body of interface X**

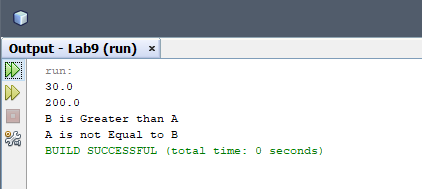
**}**

**Source Code**

1. **Lab9.java :**

|  |
| --- |
| /\*  Interface Relaeted Problem  \*/  package lab9;  interface add {  void add();  }  interface multi{  void multi();  }  interface arithmetic\_operation extends add,multi{    }  interface greater{  void determine\_greater();  }  interface equal{  void determine\_equal();  }  interface logical\_operation extends greater,equal{    }  interface operation extends arithmetic\_operation,logical\_operation{    }  class number{  private float a;  private float b;  public number(float a,float b){  this.a=a;  this.b=b;  }  public float get\_a(){  return a;  }  public float get\_b(){  return b;  }  }  class math extends number implements operation {    public math(float a,float b){  super(a,b);  }  @Override  public void add(){  System.out.println(super.get\_a()+ super.get\_b());  }  @Override  public void multi(){  System.out.println(super.get\_a()\* super.get\_b());  }  @Override  public void determine\_greater() {  if( super.get\_a()>super.get\_b()){  System.out.println("A is Greater than B");  }  else if( super.get\_a()<super.get\_b()){  System.out.println("B is Greater than A");  }  }  @Override  public void determine\_equal() {  if( super.get\_a()==super.get\_b()){  System.out.println("A is Equal to B");  }  else  System.out.println("A is not Equal to B");  }  }  public class Lab9 {  public static void main(String[] args) {  math ob=new math(10,20);  ob.add();  ob.multi();  ob.determine\_greater();  ob.determine\_equal();  }  } |

**Output**



**Conclusion :** By our Course Teachers help and my knowledge about Java, I completed the program.

**# The End #**