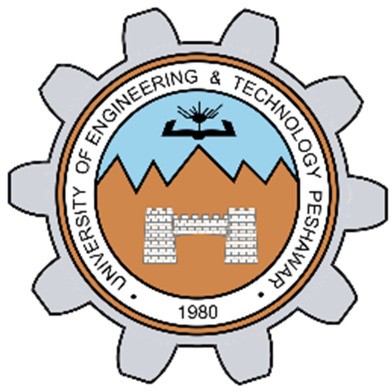
**Multiple and Multilevel Inheritance**

## LAB # 06



**Fall 2020**

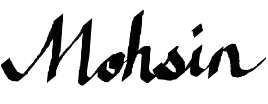
**CSE208L Object Oriented Programming Lab**

Submitted by: **Syed Mohsin Shah**

Registration No. : **19PWCSE1749**

Class Section: **A**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”



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

Submitted to:

## **Engr. Sumayyea Salahuddin**

January 12, 2021

Department of Computer Systems Engineering University of Engineering and Technology, Peshawar

# **Objectives of the Lab:**

* Understand the concept of multiple and multilevel inheritance.
* Write two level inherited classes.
* Write a class inherited from multiple base classes.
* Write multi-file programs covering inheritance.
* Use and test deep and shallow copy in a class.

Activity 1

# **Title:**

# Multi-level Inheritance

# **Problem Analysis:**

Create a class First. It contains one protected data member *f* and one public input function *f\_input()*. Use the function to take *f* from user on runtime.

Next, create a derived class Second from First class. This class also contains only one protected data member *s* and one public input function *s\_input*(). Call *f\_input*() function inside *s\_input*() and then take *s* from user on runtime.

Finally, create another derived class Third from Second class. This class contains one protected datamember *t*. It contains three public functions. An input function *t\_input()* that takes *t* from user on runtime, a max function *max*() that finds maximum of *f*, *s*, and *t* and displays the maximum, and show function that displays *f*, *s*, and *t*. Note, call *s\_input*() inside t\_input() and then take *t* from user.

Write main function to test the functionality. Create an object of Third. Call t\_input(), show(), and max() functions according to test case given in 6.4.

Note: For python, keep same name for input function i.e. *in1*() in all three classes.

# **Algorithm:**

UML diagram for the above problem is given below:

|  |
| --- |
| **First** |
| * f - int |
| * f\_input() |

* + First make class First:
    - Declare f as protected integer data member.
    - Define f\_input() method to input data into f.
  + Then make class Second:

|  |
| --- |
| **Second** |
| * f - int |
| * f\_input() |

* + - Declare s as protected integer data member.
    - Define s\_input() method to call f\_input() then input data into s.
  + Then make class Third:
    - Declare t as protected integer data member.
    - Define t\_input() method to call s\_input() then input data into s.
    - Define max() to print (t>s) && (t>f) ? t : (s>f) ? s : f

|  |
| --- |
| **Third** |
| * t - int |
| * t\_input() * max() * show() |

* + - Define show() to print f, s and t values
  + In main function, make object of Third to demonstrate the use of Multi-level
  + Inheritance.
  + Create Object t1, call t\_input() to input data into t1, then show() to print them.
  + Call max() to print the max value.

# **Flowchart:**

# **C++**

## **Source Code:**

#include <iostream>

using namespace std;

class First

{

protected:

int f;

public:

void f\_input()

{

cout<<"Enter First value: ";

cin>>f;

}

};

class Second: public First

{

protected:

int s;

public:

void s\_input()

{

f\_input();

cout<<"Enter Second value: ";

cin>>s;

}

};

class Third: public Second

{

protected:

int t;

public:

void t\_input()

{

s\_input();

cout<<"Enter Third value: ";

cin>>t;

}

void max()

{

cout<<"\nMax: ";

(t>s)&&(t>f)?cout<<t:(s>f)?cout<<s:cout<<f;

cout<<endl;

}

void show(){

cout<<"\nFirst value: "<<f;

cout<<"\nSecond value: "<<s;

cout<<"\nThird value: "<<t;

cout<<endl;

}

};

int main(){

Third t1;

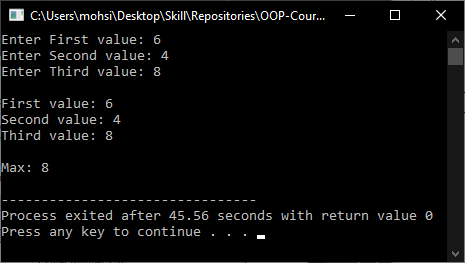
t1.t\_input();

t1.show();

t1.max();

}

## **Output:**



# **Java**

## **Source Codes:**

### **First.java:**

**package** task1;

**import** java.util.\*;

**public** **class** First

{

**protected** **int** f;

**public** **void** f\_input()

{

Scanner input = **new** Scanner(System.***in***);

System.***out***.print("Enter First value: ");

**this**.f = input.nextInt();

}

}

### **Test.java:**

**package** task1;

**import** java.util.\*;

**public** **class** Second **extends** First

{

**protected** **int** s;

**public** **void** s\_input()

{

f\_input();

Scanner input = **new** Scanner(System.***in***);

System.***out***.print("Enter Second value: ");

**this**.s = input.nextInt();

}

}

### **Third.java:**

**package** task1;

**import** java.util.\*;

**public** **class** Third **extends** Second

{

**protected** **int** t;

**public** **void** t\_input()

{

s\_input();

Scanner input = **new** Scanner(System.***in***);

System.***out***.print("Enter Third value: ");

**this**.t = input.nextInt();

}

**public** **void** max()

{

**int** max = (**this**.t>**this**.s)&&(**this**.t>**this**.f)?**this**.t:(**this**.s>**this**.f)?**this**.s:**this**.f;

System.***out***.println("\nMax: " + max);

}

**public** **void** show()

{

System.***out***.println("\nFirst value: " + **this**.f);

System.***out***.println("Second value: " + **this**.s);

System.***out***.println("Third value: " + **this**.t);

}

}

### **Main.java:**

**package** task1;

**public** **class** Main

{

**public** **static** **void** main(String[] args)

{

Third t1 = **new** Third();

t1.t\_input();

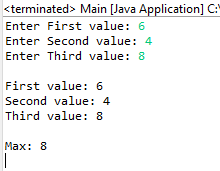
t1.show();

t1.max();

}

}

## **Output:**

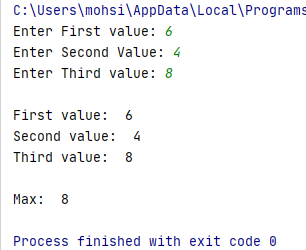


# **Python**

## **Source Code:**

class First:  
 f = 0  
  
 def f\_input(self):  
 self.f = int(input(**"Enter First value: "**))  
  
  
class Second(First):  
 s = 0  
  
 def s\_input(self):  
 self.f\_input()  
 self.s = int(input(**"Enter Second Value: "**))  
  
  
class Third(Second):  
 t = 0  
  
 def t\_input(self):  
 self.s\_input()  
 self.t = int(input(**"Enter Third value: "**))  
  
 def max(self):  
 print(**"**\n**Max: "**, self.t if (self.t > self.s) and (self.t > self.f) else self.s if (self.s > self.f) else self.f)  
  
 def show(self):  
 print(**"**\n**First value: "**, self.f)  
 print(**"Second value: "**, self.s)  
 print(**"Third value: "**, self.t)  
  
  
t1 = Third()  
  
t1.t\_input()  
  
t1.show()  
  
t1.max()

## **Output:**



Activity 2

# **Title:**

# Multiple Inheritance

# **Problem Analysis:**

Create a class base. It contains one protected data member *ba* and two public functions *input\_base()* and *show\_base*(). Use *input\_base*() to take *ba* from user on runtime while *show\_base*() to display content of *ba*.

Create another class exponent. It also contains one protected data member *exp* and two public functions *input\_exp()* and *show\_exp*(). Use *input\_exp*() to take *exp* from user on runtime while *show\_exp*() to display content of *exp*.

Next, create derived class power from base class and exponent class. This class contains one data member *po*. It contains three public functions. A constructor to initialize po with 1, an input function *in1()*, and *show1*() function. The *in1*() calls *input\_base*() and *input\_exp*() functions. The *show1*() calls *show\_base*() and *show\_exp*() functions; computes power using *ba* and *exp* and store in *po*; and displays computed power.

Write main function to test the functionality. Create an object of power. Call in1() and show1()functions according to test case given in 6.4.

Note: Write code for C++ and Python for this activity. Java does not support multiple inheritance. **Algorithm:**

UML diagram for the above problem is given below:

|  |
| --- |
| **Base** |
| * ba - int |
| * input\_base() * show\_base() |

* + First make class Base:
    - Declare ba as protected integer data member.
    - Define input\_base() method to input data into ba.
    - Define show\_base() method to print value of ba.
  + Then make class Exponent:

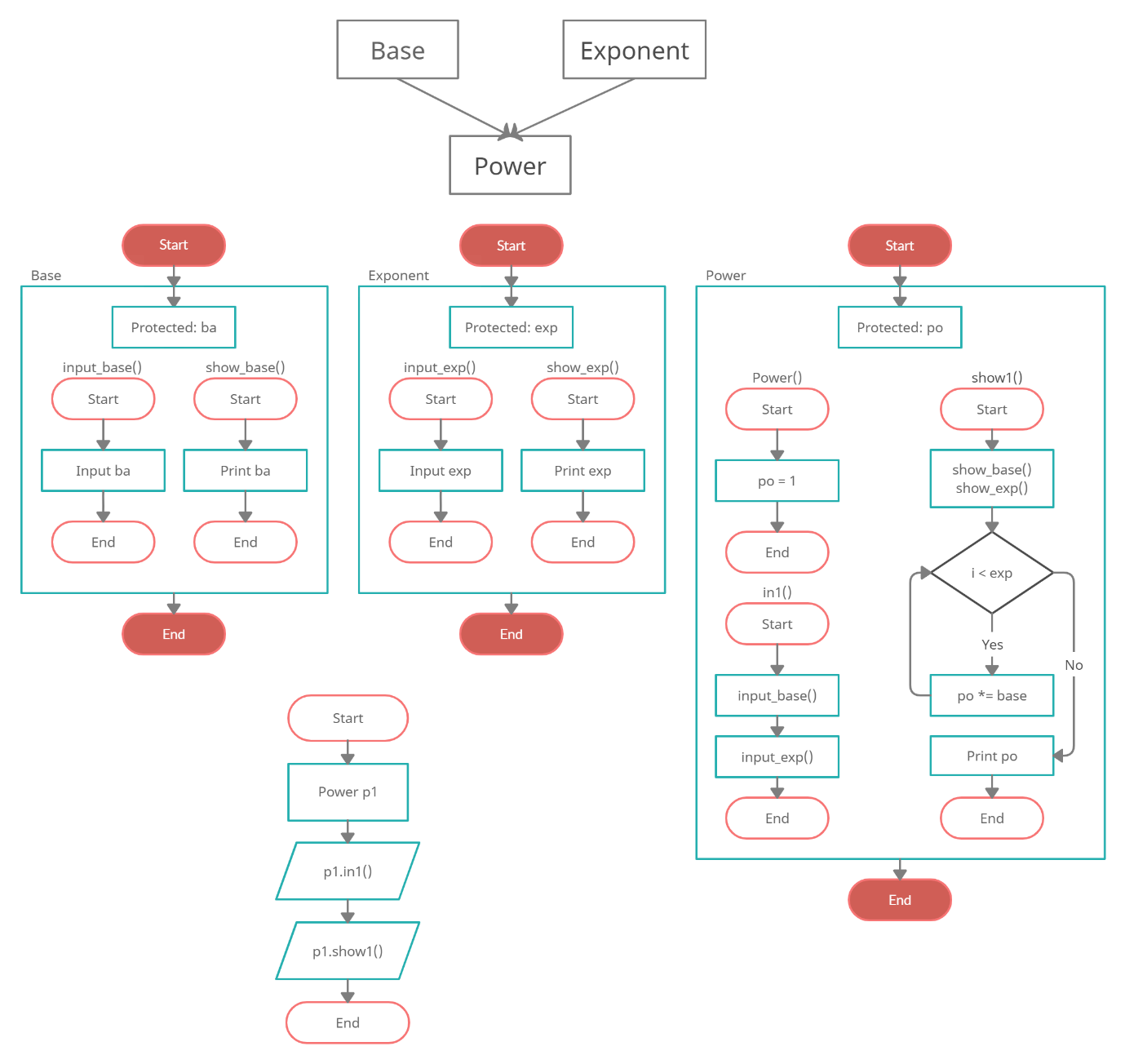
|  |
| --- |
| **Exponent** |
| * exp - int |
| * input\_exp() * show\_exp() |

* + - Declare exp as protected integer data member.
    - Define input\_exp() method to input data into exp.
    - Define show\_exp() method to print value of exp.
  + Then make class Third:
    - Declare po as protected integer data member.
    - Define parameter-less Constructor to initialize po to 1.

|  |
| --- |
| **Power** |
| * po - int |
| * Power() * in1() * show1() |

* + - Define in1() to call input\_base() and input\_exp()
    - Define show1() to:
      * Call show\_base();
      * Call show\_exp();
      * For(0 to exp): po = po \* ba
      * Print po
  + In main function, make object of Power to demonstrate the use of Multi-level Inheritance.
  + Create Object p1, call in1() to input data into p1, then show1() to print them.

# **Flowchart:**

****

# **C++**

## **Source Code:**

#include <iostream>

using namespace std;

class Base

{

protected:

int ba;

public:

void input\_base()

{

cout<<"Enter Base: ";

cin>>ba;

}

int show\_base()

{

cout<<"\nBase: "<<ba<<endl;

}

};

class Exponent

{

protected:

int exp;

public:

void input\_exp()

{

cout<<"Enter Exponent: ";

cin>>exp;

}

int show\_exp()

{

cout<<"Exponent: "<<exp<<endl;

}

};

class Power: public Base, Exponent

{

protected:

int po;

public:

Power():po(1){}

void in1()

{

input\_base();

input\_exp();

}

void show1()

{

show\_base();

show\_exp();

for(int i = 0; i < exp; i++)

po \*= ba;

cout<<"Power = "<<po<<endl;

}

};

int main(){

Power p1;

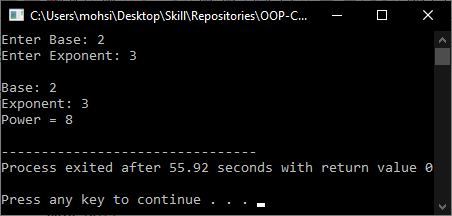
p1.in1();

p1.show1();

return 0;

}

## **Output:**

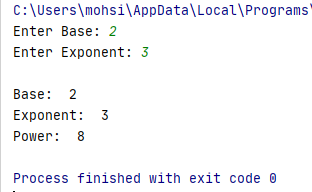


# **Python**

## **Source Code:**

class Base:  
 ba = 0  
  
 def input\_base(self):  
 self.ba = int(input(**"Enter Base: "**))  
  
 def show\_base(self):  
 print(**"**\n**Base: "**, self.ba)  
  
  
class Exponent:  
 exp = 0  
  
 def input\_exp(self):  
 self.exp = int(input(**"Enter Exponent: "**))  
  
 def show\_exp(self):  
 print(**"Exponent: "**, self.exp)  
  
  
class Power(Base, Exponent):  
 po = 0  
  
 def \_\_init\_\_(self):  
 self.po = 1  
  
 def in1(self):  
 self.input\_base()  
 self.input\_exp()  
  
 def show1(self):  
 self.show\_base()  
 self.show\_exp()  
  
 for \_ in range(self.exp):  
 self.po \*= self.ba  
  
 print(**"Power: "**, self.po)  
  
  
p1 = Power()  
  
p1.in1()  
  
p1.show1()

## **Output:**

****

Activity 3

# **Title:**

# Multi-file Programming

# **Problem Analysis:**

Redo Activity 6.3.1 and 6.3.2 using multi-file programming.

Note: In C++, create header file (\*.h) for each class and main file for main function (e.g. lab6t3.cpp).

Include the header file to access the respective class.

Note: In python, save all class in separate \*.py file (e.g. lab6t1.py containing First, Second, and Third classes and lab6t2.py containing base, exponent, and power classes )and then access using import in main python file (e.g. lab6t3.py).

Note: Since, Java is already doing multi-file programming so no need to do this activity in Java.

**Activity 1:**

# **C++**

## **Source Codes:**

### **number.h:**

#ifndef NUMBER\_H

#define NUMBER\_H

#include <iostream>

using namespace std;

class First

{

protected:

int f;

public:

void f\_input()

{

cout<<"Enter First value: ";

cin>>f;

}

};

class Second: public First

{

protected:

int s;

public:

void s\_input()

{

f\_input();

cout<<"Enter Second value: ";

cin>>s;

}

};

class Third: public Second

{

protected:

int t;

public:

void t\_input()

{

s\_input();

cout<<"Enter Third value: ";

cin>>t;

}

void max()

{

cout<<"\nMax: ";

(t>s)&&(t>f)?cout<<t:(s>f)?cout<<s:cout<<f;

cout<<endl;

}

void show(){

cout<<"\nFirst value: "<<f;

cout<<"\nSecond value: "<<s;

cout<<"\nThird value: "<<t;

cout<<endl;

}

};

#endif

### **Main.cpp:**

#include <iostream>

#include "number.h"

using namespace std;

int main(){

Third t1;

t1.t\_input();

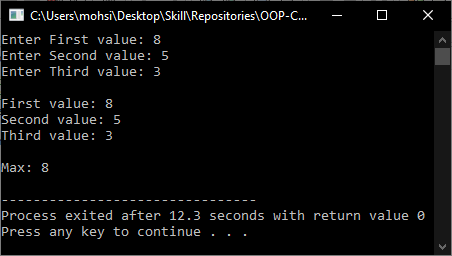
t1.show();

t1.max();

return 0;

}

## **Output:**



# **Python**

## **Source Code:**

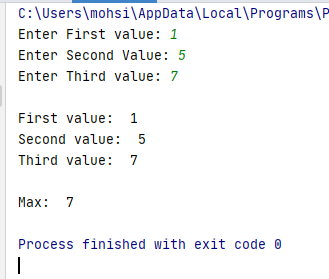
### **Number.py:**

class First:  
 f = 0  
  
 def f\_input(self):  
 self.f = int(input(**"Enter First value: "**))  
  
  
class Second(First):  
 s = 0  
  
 def s\_input(self):  
 self.f\_input()  
 self.s = int(input(**"Enter Second Value: "**))  
  
  
class Third(Second):  
 t = 0  
  
 def t\_input(self):  
 self.s\_input()  
 self.t = int(input(**"Enter Third value: "**))  
  
 def max(self):  
 print(**"**\n**Max: "**, self.t if (self.t > self.s) and (self.t > self.f) else self.s if (self.s > self.f) else self.f)  
  
 def show(self):  
 print(**"**\n**First value: "**, self.f)  
 print(**"Second value: "**, self.s)  
 print(**"Third value: "**, self.t)

### **Main.py:**

from Task3.Task1.Number import Third  
  
t1 = Third()  
  
t1.t\_input()  
  
t1.show()  
  
t1.max()

## **Output:**

****

**Activity 2:**

# **C++**

## **Source Codes:**

### **power.h:**

#ifndef POWER\_H

#define POWER\_H

#include <iostream>

using namespace std;

class Base

{

protected:

int ba;

public:

void input\_base()

{

cout<<"Enter Base: ";

cin>>ba;

}

int show\_base()

{

cout<<"\nBase: "<<ba<<endl;

}

};

class Exponent

{

protected:

int exp;

public:

void input\_exp()

{

cout<<"Enter Exponent: ";

cin>>exp;

}

int show\_exp()

{

cout<<"Exponent: "<<exp<<endl;

}

};

class Power: public Base, Exponent

{

protected:

int po;

public:

Power():po(1){}

void in1()

{

input\_base();

input\_exp();

}

void show1()

{

show\_base();

show\_exp();

for(int i = 0; i < exp; i++)

po \*= ba;

cout<<"Power = "<<po<<endl;

}

};

#endif

### **Main.cpp:**

#include <iostream>

#include "power.h"

using namespace std;

int main(){

Power p1;

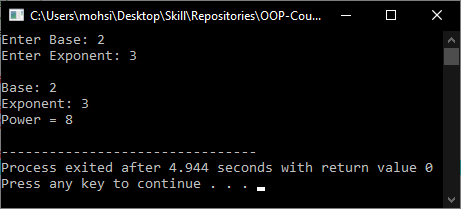
p1.in1();

p1.show1();

return 0;

}

## **Output:**



# 

# **Python**

## **Source Code:**

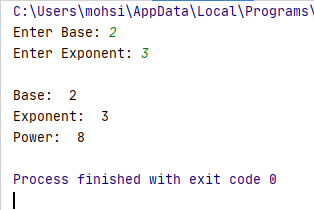
### **Power.py:**

class Base:  
 ba = 0  
  
 def input\_base(self):  
 self.ba = int(input(**"Enter Base: "**))  
  
 def show\_base(self):  
 print(**"**\n**Base: "**, self.ba)  
  
  
class Exponent:  
 exp = 0  
  
 def input\_exp(self):  
 self.exp = int(input(**"Enter Exponent: "**))  
  
 def show\_exp(self):  
 print(**"Exponent: "**, self.exp)  
  
  
class Power(Base, Exponent):  
 po = 0  
  
 def \_\_init\_\_(self):  
 self.po = 1  
  
 def in1(self):  
 self.input\_base()  
 self.input\_exp()  
  
 def show1(self):  
 self.show\_base()  
 self.show\_exp()  
  
 for \_ in range(self.exp):  
 self.po \*= self.ba  
  
 print(**"Power: "**, self.po)

### **Main.py:**

from Task3.Task2.Power import Power  
  
p1 = Power()  
  
p1.in1()  
  
p1.show1()

## **Output:**

****

**Conclusion:**

These programs helps us in building the fundamental concepts of Multi-level, Multiple Inheritance and Multi-file programming in various languages. It teaches us the prerequisite fundamentals for higher level programming. We learn the various concepts about the Inheritance and Multi-file programming in OOP with the help of these programs.