**CSE102L Computer Programming Lab**

**LAB # 5**

****

**2020**

**Submitted to:**

**Engr. Sumayya Salahuddin**

**Submitted by:**

**TAYYABA**

**Registration No :**

**19PWCSE1854**

**Semester: 3nd**

**Class Section:** **C**

“On my honor, as student of University of Engineering and Technology,

I have neither given nor received unauthorized assistance on this

academic work.”

NOV, 26 , 2020

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Objectives of the Lab:**

Objectives of the lab are to:

# Clearly understand the purpose and advantages of OOP

# Understand the concept of a Class and Objects

# Develop a basic class containing Data Members and Member Functions

# Use access specifiers to access Class Members

# Make Simple and Overloaded Constructor

# Use the Class Objects and Member Functions to provide and extract data from Object

# Practice with Classes and Objects

# Activity # 01

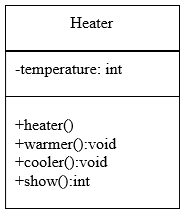
**Title:**

Make a class for heater and model it using temperature.

**Problem analysis:**

Create a class, **Heater** that contains a single integer field, **temperature**. Define a constructor that takes no parameters. The **temperature** field should be set to the value 15 in the constructor. Define the mutators **warmer** and **cooler**, whose effect is to increase or decrease the value of the temperature by 5 respectively. Define an accessor method to return the value of **temperature**. Demonstrate the use of Heater class.

**Algorithm:**

 UML diagram for the above problem is given below:

* First make class heater
* Declare temperature as private integer field
* Define no argument constructor to set value of temperature to 15
* Define warmer and cooler method to increase and decrease temperature by 5 respectively
* Define show function to return the temperature value.
* In main function, make objects of heater to demonstrate the use of heater
* Call each function one after the other and display the show function as shown in the flow chart.

**Flowchart:**



start

Private int temperature

start

temperature 15

end

start

temperature

temperature +5

end

Heater()

void warmer()

start

temperature

temperature -5

end

start

end

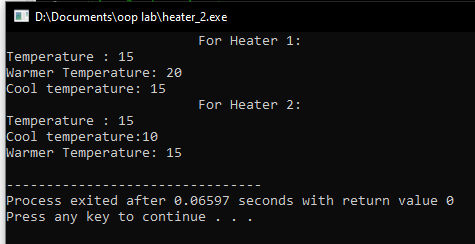
void cooler()

int show()

return temperature

end

**In C++**

**Source code: Output**

#include <iostream>

using namespace std;

//create a class

class Heater{

private:

int temperature;

public:

Heater()//constructor

{

temperature=15;

}

void warmer()//function

{

temperature+=5;

}

void cooler()//function

{

temperature-=5;

}

int show()//function

{

return temperature;

}

};

int main()

{

Heater h1;

cout<<" For Heater 1: "<<endl;

cout<<"Temperature : "<<h1.show()<<endl;

cout<<"Warmer Temperature: ";

h1.warmer();

cout<<h1.show()<<endl;

cout<<"Cool temperature: ";

h1.cooler();

cout<<h1.show()<<endl;

Heater h2;

cout<<" For Heater 2: "<<endl;

cout<<"Temperature : "<<h2.show()<<endl;

cout<<"Cool temperature:";

h2.cooler();

cout<< h2.show()<<endl;

cout<<"Warmer Temperature: ";

h2.warmer();

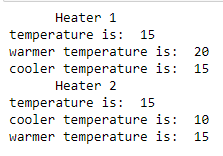
cout<< h2.show()<<endl;

return 0;

}

**In Python**

**Source code: Output:**

class Heater:

def \_\_init\_\_(self):

self.temp = 15;

def warmer(self):

self.temp+=5

def cooler(self):

self.temp-=5

def show(self):

return self.temp

print(" Heater 1")

obj1 = Heater()

print("temperature is: ", obj1.show())

obj1.warmer()

print("warmer temperature is: ", obj1.show())

obj1.cooler()

print("cooler temperature is: ", obj1.show())

print(" Heater 2")

obj2 = Heater()

print("temperature is: ", obj2.show())

obj2.cooler()

print("cooler temperature is: ", obj2.show())

obj2.warmer()

print("warmer temperature is: ", obj2.show())

**Conclusion:**

This program helps us in understanding the basic concepts of classes and objects in different languages. It acts as a base for us and helps us in preparing ourselves for the higher level of programming. We get to know about the constructor and method in OOP with the help of this program.

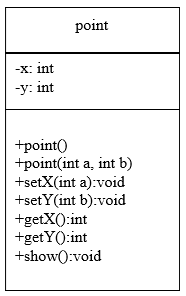
# Activity # 02

**Title:**

Make a class for point and model it using x & y coordinates.

**Problem analysis:**

Create a class called **Point** that has two data members: **x**‐ and **y**‐coordinates of the point. Provide a no argument and a 2‐argument constructor. Provide separate get and set functions for the each of the data members i.e. **getX**, **getY**, **setX**, **setY**. The getter functions should return the corresponding values to the calling function. Provide a **display** method to display the point in (x, y) format. Make appropriate function **const**.

**Algorithm:**

UML diagram for the above problem is given below:

* First make class point.
* Declare x & y co-ordinate as private integer field
* Define no argument constructor to set value of x & y to 0.
* Define argument constructor to set the parameters of x & y.
* Define setX and setY method to set the value of x & y respectively.
* Define getX and getY method to get the value of x & y.
* Define show function to display the constructor’s output.
* In main function, make objects of point to demonstrate the use of point.
* Call each function one after the other and display the show function as shown in the flow chart.



**Flowchart:**

Start

Private: int x, y

start

end

start

x=a , y=b

x=0 , y=0

end

point ()

point(int a, int b)

start

end

start

end

y=b

x=a

void setX(int a)

void setY(int b)

start

end

start

start

print x & y

return x

return y

end

end

void show()

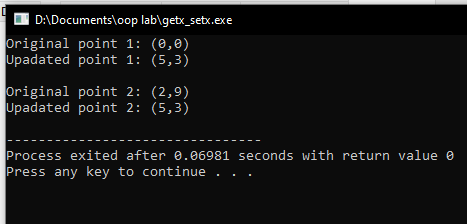
int getY()

int getX()

End

**In C++**

**Source code: Output**

#include <iostream>

using namespace std;

//create a class point

class point{

private:

int x;

int y;

public:

point()//constructor without parameters

{

x=0;

y=0;

}

point(int a, int b)//constructor with parameter

{

x=a;

y=b;

}

void setx(int a)

{

x=a;

}

void sety(int b)

{

y=b;

}

int getx()

{

return x;

}

int gety()

{

return y;

}

void show()

{

cout<<"("<<x<<","<<y<<")"<<endl;

}

};

int main()

{

point p1,p2(2,9);

//for point 1

cout<<"Original point 1: ";

p1.show();

cout<<"Upadated point 1: ";

p1.setx(5);

p1.sety(3);

cout<<"("<<p1.getx()<<","<<p1.gety()<<")"<<endl;

//for point 2

cout<<"\nOriginal point 2: ";

p2.show();

cout<<"Upadated point 2: ";

p2.setx(5);

p2.sety(3);

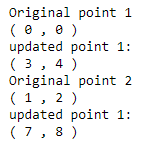
cout<<"("<<p2.getx()<<","<<p2.gety()<<")"<<endl;

return 0;

}

**In Python**

**Source code: Output:**

class Point:

def \_\_init\_\_(self, a,b):

self.x=a

self.y=b

def setX(self,p):

self.x=p

def setY(self,p):

self.y=p

def getX(self):

return self.x

def getY(self):

return self.y

def show(self):

print("(", self.x, ",", self.y, ")")

print("Original point 1")

p1 = Point(0,0)

p1.show()

print("updated point 1:")

p1.setX(3)

p1.setY(4)

print("(", p1.getX(), ",", p1.getY(), ")")

print("Original point 2")

p2 = Point(1,2)

p2.show()

print("updated point 1:")

p2.setX(7)

p2.setY(8)

print("(", p2.getX(), ",", p2.getY(), ")")

**Conclusion:**

This program helps us in understanding the basic concepts of classes and objects in different languages. It acts as a base for us and helps us in preparing ourselves for the higher level of programming. We get to know about the constructor and method in OOP with the help of this program.

# Activity # 03

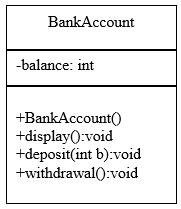
**Title:**

Make a class for Bank Acount and model it using balance.

**Problem analysis:**

Create a class called ***Bank Account*** that models a checking account at a bank. The program creates an account with an opening balance, displays the balance, makes a deposit and a withdrawal, and then displays the new balance. Note in withdrawal function, if balance is below Rs. 500 then display message showing insufficient balance otherwise allow withdrawal.

**Algorithm:**

 UML diagram for the above problem is given below:

* First make class Bank Account.
* Declare balance as private integer field.
* Define no argument constructor to set value of balance to 1000.
* Define deposit and withdrawal method to increase and decrease balance by 500 respectively
* Define display function to display the output
* In main function, make objects of bank account to demonstrate the use of bank account.
* Call each function one after the other and display the output as shown in the flow chart.

**Flowchart:**



Start

Private int balance

start

balance = 1000

end

BankAccount()

start

end

Void display ()

print “current balance”, balance

start

balance += 500

end

Void deposit()

start

balance += 500

end

void withdrawal()

Balance<=500

print “insufficient balance”

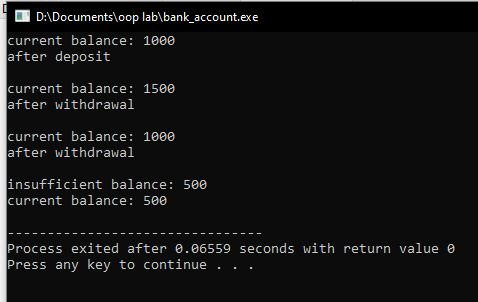
True

False

End

**In C++**

**Source code: Output**

#include <iostream>

using namespace std;

//create class for bank account

class BankAccount{

private:

int balance;

public:

BankAccount()//constructor without parameter

{

balance=1000;

}

void display()

{

cout<<"current balance: "<<balance<<endl;

}

void deposit()

{

balance+=500;

}

void withdrawal()

{

balance-=500;

if(balance<=500)

cout<<"insufficient balance: "<<balance<<endl;;

}

};

int main()

{

BankAccount obj;//create object for class

obj.display();

obj.deposit();

cout<<"after deposit "<<endl;;

obj.display();

obj.withdrawal();

cout<<"after withdrawal"<<endl;

obj.display();

cout<<"after withdrawal"<<endl;

obj.withdrawal();

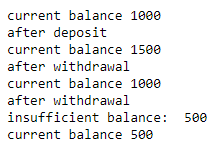
obj.display();

return 0;

}

**In Python**

**Source code: Output:**

class BankAccount:

def \_\_init\_\_(self):

self.balance=1000

def display(self):

print("current balance", self.balance)

def deposit(self):

self.balance+=500

def withdrawal(self):

self.balance-=500

if self.balance <=500:

print("insufficient balance: ",self.balance)

obj = BankAccount()

obj.display()

obj.deposit()

print("after deposit ")

obj.display()

obj.withdrawal()

print("after withdrawal")

obj.display()

print("after withdrawal")

obj.withdrawal()

obj.display()

**Conclusion:**

This program helps us in understanding the basic concepts of classes and objects in different languages. It acts as a base for us and helps us in preparing ourselves for the higher level of programming. We get to know about the constructor and method in OOP with the help of this program.