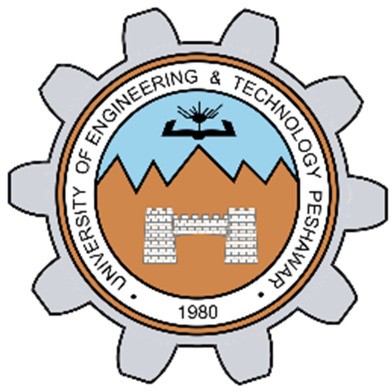
**Object Programming Essentials 3**

## LAB # 03



**Fall 2020**

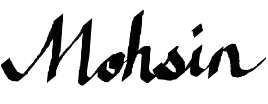
**CSE208L Object Oriented Programming Lab**

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**

December 28, 2020

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

# **Objectives of the Lab:**

* Understand and implement parameter-less and parameterized constructor in a class.
* Write a class (C++/Java) with overloaded constructors.
* Write a test program to use default copy constructor (C++).
* Understand the difference between a Shallow Copy and a Deep Copy.
* Understand the concept of dynamic memory allocation.
* Implement deep and shallow copy in a class (C++/Java).
* Use and test deep and shallow copy in a class.
* Understand and implement destructor in a class (C++/Python).

Activity

# **Title:**

# Create Employee class and create its Deep and Shallow Copy.

# **Problem Analysis:**

**C++:**

Create a class called employee. This class maintains information about name (char\*), department (char\*), salary (double), and period of service in years (double).

1. Provide a parameter-less constructor to initialize the data members to some fixed values.
2. Provide a 4-argument parameterized constructor to initialize the members to values sent from calling function. (You have to make dynamic allocation for both name and department data members in constructor.)
3. Provide a copy-constructor that performs the deep copy of the data members.
4. Provide an input function that takes all the values from user during run-time.
5. Provide a show function that shows all the information about a specific employee to user.
6. Provide a destructor to free the memory allocated to name and department in constructor.

Write all the member function outside a C++ class. Write a driver program to test the functionality of the above-mentioned class.

**Java:**

Create a class called employee. This class maintains information about name (String), department (String), salary (double), and period of service in years (double).

1. Provide a parameter-less constructor to initialize the data members to some fixed values.
2. Provide a 4-argument parameterized constructor to initialize the members to values sent from calling function.
3. Provide an input function that takes all the values from user during run-time using Scanner.
4. Provide a deep clone() that performs the deep copy of the data members.
5. Provide a show function that shows all the information about a specific employee to user.

Write a Test class to test the functionality of the above-mentioned class using Test Case given in 4.6.

Next, change the department of e1 to EE and view contents of e1 and e3.

Change the clone() function to shallow and observe the output of e1 and e3 after changing the department. Write your observation for both shallow & deep clone and compare it with

Java Example in Section 4.3.9. If the Address class is to be used in employee, how shallow & deep clone function will behave? And why?

**Python:**

Import copy module. Create a class called employee.

1. Provide a 4-argument parameterized constructor to initialize the members to values sent from calling function.
2. Provide an input function dataIn() that takes all the values from user during run-time.
3. Provide a show function that shows all the information about a specific employee to user.

Test the functionality of the above-mentioned class using Test Case given in 4.6.

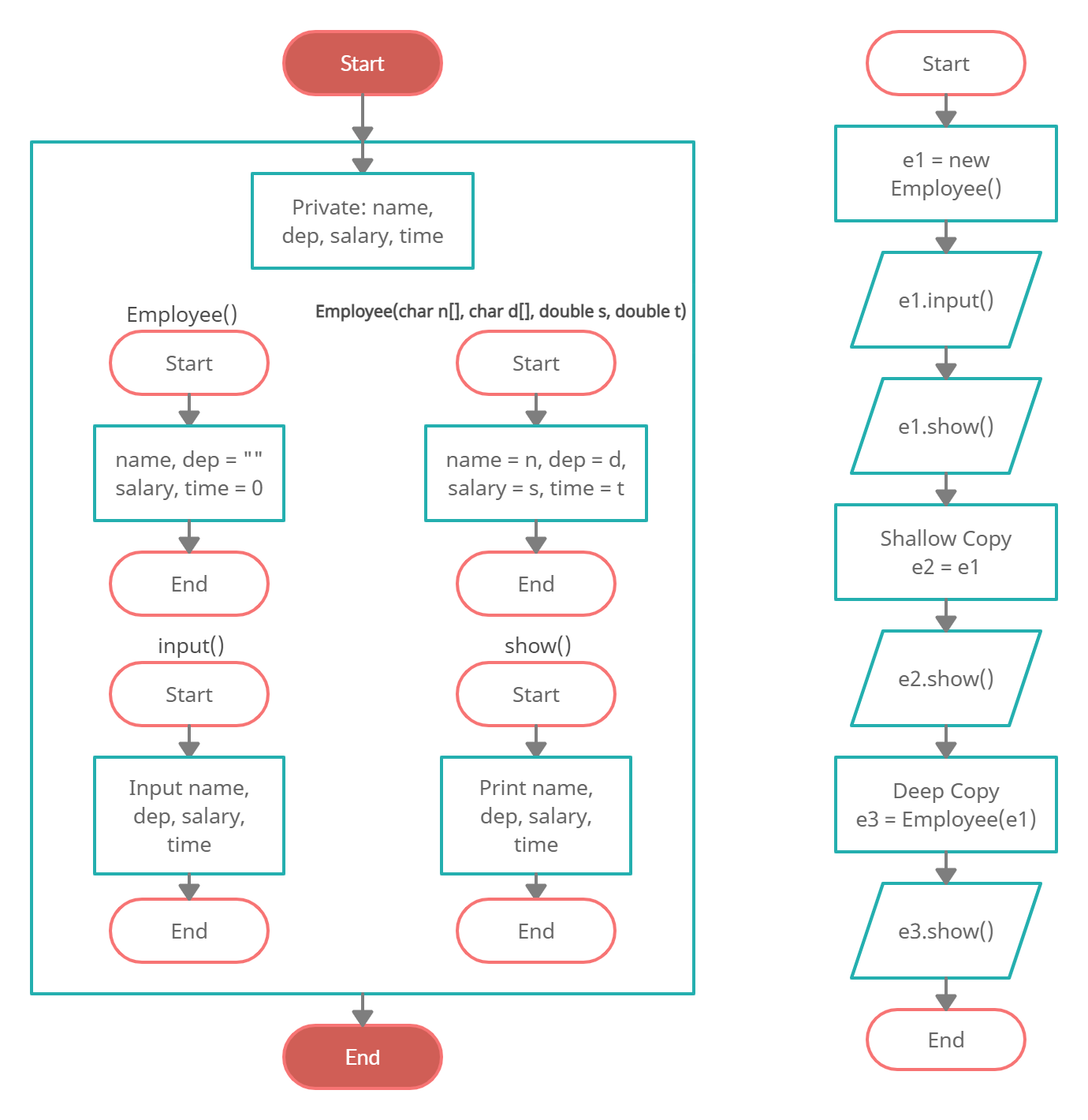
# **Algorithm:**

UML diagram for the above problem is given below:

|  |
| --- |
| **Employee** |
| * name: char * department: char * salary: double * serviceTime: double |
| * Employee() * Employee(char n[], char d[], double s, double t) * input(): void * show(): void * Employee(const Employee &e) * ~Employee() |

* + First make class Employee:
    - Declare name, department, salary and serviceTime as private data members.
    - Define no argument constructor and also a parameterized constructor.
    - Define show() method to print data in employee.
    - Define input() method to input data into employee.
    - Define a copy constructor.
    - Define Destructor.
  + In main function, make objects of Employee to demonstrate the use of Employee.
  + Create Object e1, call input() to input data into e1, then show() to print them.
  + Do Shallow Copy: Create Object e2 assigned to e1, then call its show() to print data.
  + Do Deep Copy: Create Object e3 with e1 as parameter, then use show() to display data.

# **Flowchart:**



Employee

# **C++**

## **Source Code:**

#include<iostream>

#include<string.h>

using namespace std;

class Employee

{

private:

char\* name;

char\* department;

double salary;

double serviceTime;

public:

Employee():name(0),department(0)

{

salary = 0;

serviceTime = 0;

}

Employee(char n[], char dep[], double sal, double t):name(n),department(dep)

{

salary = sal;

serviceTime = t;

}

//Deep Copy Constructor

Employee(const Employee &e)

{

int length = strlen(e.name);

name = new char[length+1];

strcpy(name, e.name);

length = strlen(e.department);

department = new char[length+1];

strcpy(department, e.department);

salary = e.salary;

serviceTime = e.serviceTime;

}

void input()

{

string d,n;

cout<<"Enter name: ";

getline(cin, n);

name = new char[n.length() + 1];

strcpy(name, n.c\_str());

cout<<"Enter department: ";

cin>>d;

department = new char[d.length() + 1];

strcpy(department, d.c\_str());

cout<<"Enter salary:";

cin>>salary;

cout<<"Enter time of service:";

cin>>serviceTime;

}

void show()

{

cout<<"\nDetails:\nName: "<<name<<endl;

cout<<"Enter department: "<<department<<endl;

cout<<"Enter salary:"<<salary<<endl;

cout<<"Enter time of service:"<<serviceTime<<endl;

}

~Employee()

{

delete[] name, department, salary, serviceTime;

}

};

int main()

{

Employee e1;

e1.input();

e1.show();

Employee e2 = e1;

e2.show();

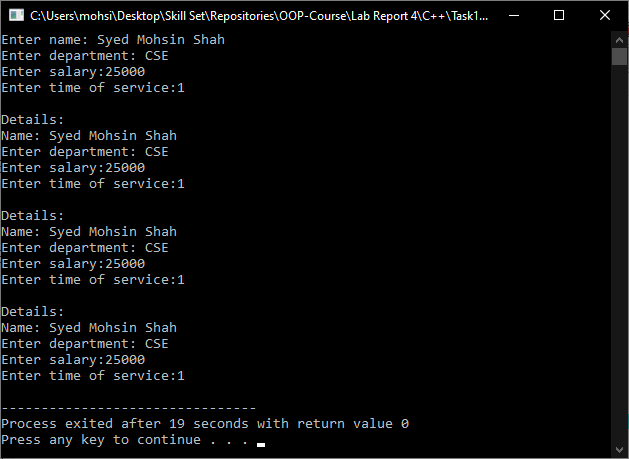
Employee e3(e1);

e3.show();

return 0;

}

## **Output:**



# **Java**

## **Source Code:**

### **Employee.java:**

**package** task1;

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

**public** **class** Employee **implements** Cloneable

{

**private** String name;

**private** String department;

**private** **double** salary;

**private** **double** serviceTime;

**public** Employee()

{

name = **null**;

department = **null**;

salary = 0;

serviceTime = 0;

}

**public** Employee(String name, String department, **double** salary, **double** serviceTime)

{

**this**.name = name;

**this**.department = department;

**this**.salary = salary;

**this**.serviceTime = serviceTime;

}

**public** **void** input()

{

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

System.***out***.print("Enter Name: ");

**this**.name = input.nextLine();

System.***out***.print("Enter Department: ");

**this**.department = input.nextLine();

System.***out***.print("Enter Salary: ");

**this**.salary = input.nextDouble();

System.***out***.print("Enter Time of Service: ");

**this**.serviceTime = input.nextDouble();

}

//Deep Clone

@Override

**public** Object clone() **throws** CloneNotSupportedException

{

Employee EmpClone = (Employee) **super**.clone();

**return** EmpClone;

}

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

{

System.***out***.println("\nDetails:\nName: " + **this**.name);

System.***out***.println("Department: " + **this**.department);

System.***out***.println("Salary: " + **this**.salary);

System.***out***.println("Time of Service: " + **this**.serviceTime);

}

}

### **Test.java:**

**package** task1;

**public** **class** Test

{

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

{

Employee e1 = **new** Employee();

e1.input();

e1.show();

Employee e2 = e1;

e2.show();

Employee e3 = **null**;

**try**

{

e3 = (Employee) e1.clone();

e3.show();

}

**catch** (CloneNotSupportedException err)

{

err.printStackTrace();

}

/\*

e1.input(); //To change CSE to EE

e1.show();

//This is Shallow Copied Object

e2.show();

//This is Deep Copied Object

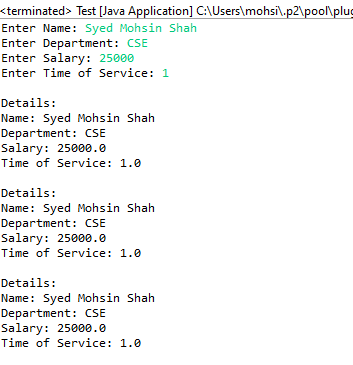
e3.show();

\*/

}

}

## **Output:**



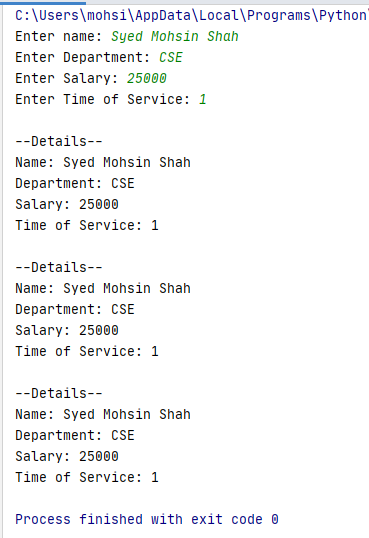
Values of **shallow copy** changes because it creates a new object and then inserts **references** to the original object. Values of **deep copy** remain because it creates a new object and then, recursively, inserts **copies** into it of the original object.

# **Python**

## **Source Code:**

import copy  
  
  
class Employee:  
 def \_\_init\_\_(self, name, department, salary, serviceTime):  
 self.name = name  
 self.department = department  
 self.salary = salary  
 self.serviceTime = serviceTime  
  
 def dataIn(self):  
 self.name = input(**"Enter name: "**)  
 self.department = input(**"Enter Department: "**)  
 self.salary = input(**"Enter Salary: "**)  
 self.serviceTime = input(**"Enter Time of Service: "**)  
  
 def show(self):  
 print(**"**\n**--Details--"**)  
 print(**"Name: "** + self.name)  
 print(**"Department: "** + self.department)  
 print(**"Salary: "** + str(self.salary))  
 print(**"Time of Service: "** + str(self.serviceTime))  
  
  
e1 = Employee(**""**, **""**, 0, 0)  
  
e1.dataIn()  
e1.show()  
  
e2 = e1  
e2.show()  
  
e3 = copy.deepcopy(e1)  
e3.show()

## **Output:**



# **Conclusion:**

This program helps us in building the fundamental concepts of Dynamic memory allocation, Destructor and Copy Constructor in various languages. It teaches us the prerequisite fundamentals for higher level programming. We learn the various concepts about the constructor and method in OOP with the help of this program.