

MIT WORLD PEACE UNIVERSITY

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

TO DEMONSTRATE THE USE OF OBJECTS, CLASSES,
CONSTRUCTORS AND DESTRUCTORS USING C++
AND JAVA.

PRACTICAL REPORT

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 34

September 1, 2022

Contents

1	Aim and Objectives	1
2	Problem Statement	1
3	Theory	1
4	Algorithm	1
5	Platform	1
6	Input	2
7	Output	2
8	Conclusion	2
9	Code	2
9.1	Java Implementation	2
9.1.1	Java Input	5
9.1.2	Java Output	6
9.2	C++ Implementation	7
9.2.1	C++ Input	10
9.2.2	C++ Output	11
10	FAQs	13

1 Aim and Objectives

To demonstrate the use of objects, classes, constructors and destructors using C++ and JAVA.

1. To study various OOP concepts
2. To acquaint with the use of objects and classes in C++ and Java.
3. To learn implementation of constructor, destructors and dynamic memory allocation

2 Problem Statement

Develop an object-oriented program to create a database of employee information system containing the following information: Employee Name, Employee number, qualification, address, contact number, salary details (basic, DA, TA, Net salary), etc. Construct the database with suitable member functions for initializing and destroying the data viz. constructor, default constructor, Copy constructor, destructor. Use dynamic memory allocation concept while creating and destroying the object of a class. Use static data member concept wherever required. Accept and display the information of Employees.

3 Theory

4 Algorithm

1. Start
2. Create Employee Class
3. Declare appropriate Data Members and define the member functions
4. Accept multiple employee's Data using an Array of Objects
5. Assign some Basic employees using default, copy and parameterized constructors
6. Show usage of Default Constructor and display the Data
7. Show usage of Parameterized constructor and display that data
8. Show usage of Copy Constructor and display that Data.
9. Destroy the objects if possible
10. End

5 Platform

Operating System: Arch Linux x86-64

IDEs or Text Editors Used: Visual Studio Code

Compilers : g++ and gcc on linux for C++, and javac, with JDK 18.0.2 for Java

6 Input

1. Number of Employees for enrollment
2. Employee ID
3. Employee Name
4. Employee Position
5. Employee Address
6. Employee Salary

7 Output

Employee Data should be displayed by use of member functions.

8 Conclusion

Thus, learned to use objects, classes, constructor and destructor and implemented solution of the given problem statement using C++ and Java.

9 Code

9.1 Java Implementation

```
1 package assignment_1;
2 import java.util.Scanner;
3
4 public class Employee {
5
6     // create an object of Scanner
7     Scanner input = new Scanner(System.in);
8
9     int emp_id;
10    int age, basic_sal, da, ta;
11    String address_city, position, name;
12    static int ssn;
13
14    Employee()
15    {
16        System.out.println("Default Constructor was called");
17    }
18
19    // Parameterized Constructor
20    Employee(int e, int a, int b, int d,
21            int t, String add, String pos, String nam)
22    {
23        System.out.println("Parameterized constructor was called");
24        emp_id = e;
25        age = a;
26        basic_sal = b;
27        da = d;
```

```
28         ta = t;
29         address_city = add;
30         position = pos;
31         name = nam;
32     }
33
34     // Copy Constructor
35     Employee(Employee E)
36     {
37         System.out.println("Copy constructor was called");
38         emp_id = E.emp_id;
39         age = E.age;
40         basic_sal = E.basic_sal;
41         da = E.da;
42         ta = E.ta;
43         address_city = E.address_city;
44         position = E.position;
45         name = E.name;
46     }
47
48     double calc_gross_sal()
49     {
50         return basic_sal + da + ta - (0.15 * basic_sal);
51     }
52
53     void display()
54     {
55         ssn = ssn + 1;
56         System.out.println("Employee ssn is: " + ssn);
57         System.out.println("Employee ID is : " + emp_id);
58         System.out.println("Employee Name: " + name);
59         System.out.println("Employee Age: " + age);
60         System.out.println("Employee Position: " + position);
61         System.out.println("Employee basic Salary: " + basic_sal);
62         System.out.println("Employee DA: " + da);
63         System.out.println("Employee TA: " + ta);
64         System.out.println("Employee Gross Salary: " + calc_gross_sal() );
65         System.out.println("Employee Address City: " + address_city);
66         System.out.println("\n");
67     }
68
69     void accept()
70     {
71         System.out.println("Enter the age :");
72         age = input.nextInt();
73         System.out.println("Employee ID is: ");
74         emp_id = input.nextInt();
75         System.out.println("Employee Name: ");
76         name = input.next();
77         System.out.println("Employee Age: ");
78         age = input.nextInt();
79         System.out.println("Employee Position: ");
80         position = input.next();
81         System.out.println("Employee basic Salary: ");
82         basic_sal = input.nextInt();
83         System.out.println("Employee DA: ");
84         da = input.nextInt();
85         System.out.println("Employee TA: ");
86         ta = input.nextInt();
```

```
87         System.out.println("Employee Address City: ");
88         address_city = input.next();
89     }
90
91     // @Override
92     // protected void finalize() throws Throwable
93     // {
94     //     try
95     //     {
96     //         input.close();
97     //     }
98     //     catch(Throwable t)
99     //     {
100        //         throw t;
101        //     }
102        //     finally
103        //     {
104        //         super.finalize();
105        //     }
106        // }
107
108 }
```

Listing 1: Employee.java

```
1 package assignment_1;
2 import java.util.Scanner;
3
4
5 public class Source {
6
7     static Scanner input = new Scanner(System.in);
8
9     public static void main(String[] args) {
10         System.out.println("This is the first Assignment");
11         int count = 0;
12
13         // Default Constructor
14         Employee CEO = new Employee();
15         CEO.emp_id = 1000;
16         CEO.name = "Kom Pany Seeio";
17         CEO.address_city = "Seoul";
18         CEO.age = 45;
19         CEO.basic_sal = 1000000;
20         CEO.da = 1000;
21         CEO.ta = 2000;
22         CEO.position = "CEO";
23
24         // Defining an object using the copy constructor
25         Employee President = new Employee(CEO);
26         President.name = "Precy Dent";
27         President.age = 45;
28         President.address_city = "Delhi";
29         President.basic_sal *= 2;
30         President.position = "President";
31
32         Employee VP = new Employee(1003, 50, 200000, 3000, 1000,
33             "Mumbai", "Vice President", "Vishey Presed Ent");
34
35     }
```

```
35
36     System.out.println("Information about the CEO");
37     CEO.display();
38     System.out.println("Information about the President");
39     President.display();
40     System.out.println("Information about the President");
41     VP.display();
42
43     System.out.println("Enter the number of employees :");
44     count = input.nextInt();
45     Employee objs[] = new Employee[count];
46
47     for(int i = 0;i < count;i++)
48     {
49         objs[i] = new Employee();
50         objs[i].accept();
51         objs[i].display();
52     }
53     System.gc();
54 }
55 }
```

Listing 2: Source.java

9.1.1 Java Input

```
1 Enter the number of employees :
2 2
3 Default Constructor was called
4 Enter the age :
5 35
6 Employee ID is:
7 006
8 Employee Name:
9 Peter
10 Employee Age:
11 17
12 Employee Position:
13 Avenger
14 Employee basic Salary:
15 500000
16 Employee DA:
17 3440
18 Employee TA:
19 3550
20 Employee Address City:
21 Brooklyn
22
23
24 Default Constructor was called
25 Enter the age :
26 4
27 Employee ID is:
28 007
29 Employee Name:
30 Thor
31 Employee Age:
32 1500
33 Employee Position:
```

```
34 Avenger
35 Employee basic Salary:
36 6000000
37 Employee DA:
38 3000
39 Employee TA:
40 5000
41 Employee Address City:
42 Asgard
```

Listing 3: Python example

9.1.2 Java Output

```
1 This is the first Assignment
2 Default Constructor was called
3 Copy constructor was called
4 Parameterized constructor was called
5 Information about the CEO
6 Employee ssn is: 1
7 Employee ID is : 1000
8 Employee Name: Kom Pany Seeio
9 Employee Age: 45
10 Employee Position: CEO
11 Employee basic Salary: 1000000
12 Employee DA: 1000
13 Employee TA: 2000
14 Employee Gross Salary: 853000.0
15 Employee Address City: Seoul
16
17
18 Information about the President
19 Employee ssn is: 2
20 Employee ID is : 1000
21 Employee Name: Precy Dent
22 Employee Age: 45
23 Employee Position: President
24 Employee basic Salary: 2000000
25 Employee DA: 1000
26 Employee TA: 2000
27 Employee Gross Salary: 1703000.0
28 Employee Address City: Delhi
29
30
31 Information about the President
32 Employee ssn is: 3
33 Employee ID is : 1003
34 Employee Name: Visey Presed Ent
35 Employee Age: 50
36 Employee Position: Vice President
37 Employee basic Salary: 200000
38 Employee DA: 3000
39 Employee TA: 1000
40 Employee Gross Salary: 174000.0
41 Employee Address City: Mumbai
42
43
44 Employee ssn is: 4
```



```
45 Employee ID is : 6
46 Employee Name: Peter
47 Employee Age: 17
48 Employee Position: Avenger
49 Employee basic Salary: 500000
50 Employee DA: 3440
51 Employee TA: 3550
52 Employee Gross Salary: 431990.0
53 Employee Address City: Brooklyn
54
55
56 Employee ssn is: 5
57 Employee ID is : 7
58 Employee Name: Thor
59 Employee Age: 1500
60 Employee Position: Avenger
61 Employee basic Salary: 6000000
62 Employee DA: 3000
63 Employee TA: 5000
64 Employee Gross Salary: 5108000.0
65 Employee Address City: Asgard
```

Listing 4: Python example

9.2 C++ Implementation

```
1 #include <iostream>
2 using namespace std;
3 class Employee
4 {
5 private:
6     static int ssn;
7 protected:
8     int something = 5;
9
10 public:
11     int emp_id = 1000;
12     int age = 0, basic_sal = 0, da = 0, ta = 0;
13     string address_city, position, name;
14
15     // Default Constructor
16     Employee()
17     {
18         ssn += 1;
19         cout << "The Default Constructor was called" << endl;
20     }
21
22     // Parameterized Constructor
23     Employee(int e, int a, int b, int d, int t, string add, string pos, string nam
24 )
25     {
26         cout << "Parameterized constructor was called\n";
27         emp_id = e;
28         age = a;
29         basic_sal = b;
30         da = d;
31         ta = t;
32         address_city = add;
```

```
32     position = pos;
33     name = nam;
34 }
35
36 // Copy Constructor
37 Employee(Employee &E)
38 {
39     cout << "Copy Constructor was called" << endl;
40     emp_id = E.emp_id;
41     age = E.age;
42     basic_sal = E.basic_sal;
43     da = E.da;
44     ta = E.ta;
45     address_city = E.address_city;
46     position = E.position;
47     name = E.name;
48 }
49
50 void display()
51 {
52     cout << "Employee ssn is:" << ssn << endl;
53     cout << "Employee ID is : " << emp_id << endl;
54     cout << "Employee Name: " << name << endl;
55     cout << "Employee Age: " << age << endl;
56     cout << "Employee Position: " << position << endl;
57     cout << "Employee basic Salary: " << basic_sal << endl;
58     cout << "Employee DA: " << da << endl;
59     cout << "Employee TA: " << ta << endl;
60     cout << "Employee Gross Salary: " << calc_gross_sal() << endl;
61     cout << "Employee Address City: " << address_city << endl;
62 }
63
64 void accept()
65 {
66     cout << "Enter the Employee ID: " << endl;
67     cin >> emp_id;
68     cout << "Enter the Employee Name: " << endl;
69     cin >> name;
70     cout << "Enter the Employee Age: " << endl;
71     cin >> age;
72     cout << "Enter the Employee Position: " << endl;
73     cin >> position;
74     cout << "Enter the Employee basic Salary: " << endl;
75     cin >> basic_sal;
76     cout << "Enter the Employee DA: " << endl;
77     cin >> da;
78     cout << "Enter the Employee TA: " << endl;
79     cin >> ta;
80     cout << "Enter the Employee Address City: " << endl;
81     cin >> address_city;
82 }
83
84 float calc_gross_sal()
85 {
86     int gross_sal = da + ta + basic_sal - (.15 * basic_sal);
87     return gross_sal;
88 }
89
90 // Destructor
```

```
91     ~Employee()
92     {
93         cout << "The Destructor was called" << endl;
94     }
95 };
96
97 int Employee::ssn = 1000;
98
99
100 int main()
101 {
102     // Defining an Object using the default Constructor
103     Employee CEO;
104     CEO.emp_id = 1000;
105     CEO.name = "Kom Pany Seeio";
106     CEO.address_city = "Seoul";
107     CEO.age = 45;
108     CEO.basic_sal = 1000000;
109     CEO.da = 1000;
110     CEO.ta = 2000;
111     CEO.position = "CEO";
112
113
114     // Defining an object using the copy constructor
115     Employee President(CEO);
116     President.name = "Precy Dent";
117     President.age = 45;
118     President.address_city = "Delhi";
119     President.basic_sal *= 2;
120     President.position = "President";
121
122     // Defining anothe object using the parameterized constructor
123     Employee VP(1003, 50, 200000, 3000, 1000, "Mumbai", "Vice President", "Visay
Presed Ent");
124
125     // Taking user input for the number of employees
126     int count = 0;
127     cout << "How many values do you wanna input ? ";
128     cin >> count;
129
130     // Dynamically Creating new Objects
131     Employee *Obj = new Employee[count];
132     cout << "Enter the Details" << endl;
133
134     // Accepting Objects from the User.
135     for (int i = 0; i < count; i++)
136     {
137         cout << "Enter information about the Employee Number: " << i + 1 << endl;
138         Obj[i].accept();
139     }
140
141     // Displaying Information about the CEO and the President
142     cout << "Information about the CEO" << endl;
143     CEO.display();
144     cout << "Information about the President" << endl;
145     President.display();
146     cout << "Information about the Vice President" << endl;
147     VP.display();
148
```

```
149 // Displaying Information about the other Employees
150 for (int i = 0; i < count; i++)
151 {
152     cout << "Information about the Employee Number: " << i + 1 << endl;
153     Obj[i].display();
154 }
155
156 delete[] Obj;
157 return 0;
158 }
```

Listing 5: Main.Cpp

9.2.1 C++ Input

```
1 The Default Constructor was called
2 Copy Constructor was called
3 Parameterized constructor was called
4 How many values do you wanna input ? 2
5 The Default Constructor was called
6 The Default Constructor was called
7
8 Enter the Details
9
10 Enter information about the Employee Number: 1
11 Enter the Employee ID:
12 005
13 Enter the Employee Name:
14 Tony
15 Enter the Employee Age:
16 40
17 Enter the Employee Position:
18 Philanthropist
19 Enter the Employee basic Salary:
20 5000000
21 Enter the Employee DA:
22 3000
23 Enter the Employee TA:
24 3000
25 Enter the Employee Address City:
26 NewYork
27
28 Enter information about the Employee Number: 2
29 Enter the Employee ID:
30 006
31 Enter the Employee Name:
32 Steve
33 Enter the Employee Age:
34 105
35 Enter the Employee Position:
36 Captain
37 Enter the Employee basic Salary:
38 600000
39 Enter the Employee DA:
40 2999
41 Enter the Employee TA:
42 2000
43 Enter the Employee Address City:
```

44 Brooklyn

Listing 6: C++ Input

9.2.2 C++ Output

```
1 Information about the CEO
2 Employee ssn is:1003
3 Employee ID is : 1000
4 Employee Name: Kom Pany Seeio
5 Employee Age: 45
6 Employee Position: CEO
7 Employee basic Salary: 1000000
8 Employee DA: 1000
9 Employee TA: 2000
10 Employee Gross Salary: 853000
11 Employee Address City: Seoul
12
13 Information about the President
14 Employee ssn is:1003
15 Employee ID is : 1000
16 Employee Name: Precy Dent
17 Employee Age: 45
18 Employee Position: President
19 Employee basic Salary: 2000000
20 Employee DA: 1000
21 Employee TA: 2000
22 Employee Gross Salary: 1.703e+06
23 Employee Address City: Delhi
24
25 Information about the Vice President
26 Employee ssn is:1003
27 Employee ID is : 1003
28 Employee Name: Visey Presed Ent
29 Employee Age: 50
30 Employee Position: Vice President
31 Employee basic Salary: 200000
32 Employee DA: 3000
33 Employee TA: 1000
34 Employee Gross Salary: 174000
35 Employee Address City: Mumbai
36
37 Information about the Employee Number: 1
38 Employee ssn is:1003
39 Employee ID is : 5
40 Employee Name: Tony
41 Employee Age: 40
42 Employee Position: Philanthropist
43 Employee basic Salary: 5000000
44 Employee DA: 3000
45 Employee TA: 3000
46 Employee Gross Salary: 4.256e+06
47 Employee Address City: NewYork
48
49 Information about the Employee Number: 2
50 Employee ssn is:1003
51 Employee ID is : 6
52 Employee Name: Steve
```

```
53 Employee Age: 105
54 Employee Position: Captain
55 Employee basic Salary: 600000
56 Employee DA: 2999
57 Employee TA: 2000
58 Employee Gross Salary: 514999
59 Employee Address City: Brooklyn
60
61 The Destructor was called
62 The Destructor was called
63 The Destructor was called
64 The Destructor was called
65 The Destructor was called
```

Listing 7: C++ Output

10 FAQs

1. What are classes?

In object-oriented programming, a class is a blueprint for creating objects (a particular data structure), providing initial values for state (member variables or attributes), and implementations of behavior (member functions or methods).

It is a basic concept of Object-Oriented Programming which revolve around the real-life entities.

```
class <class_name>{  
    field;  
    method;  
}
```

2. Explain : Array of Objects:

An array of objects is like any other array in C++ and Java. An Array usually is just a collection of variables that have the same data type, and are placed in contiguous memory locations. An Array of Objects is similar in that instead of variables there are objects which are placed contiguously in memory.

Syntax:

```
Employee obj[5];
```

3. Explain when to use different types of constructors? There are 3 Types of constructors:

- (a) **Default Constructor:** This type is called when the Object is just created in its most simple declaration. It does not take any parameters, or arguments. So if you have a simple class, that does not have many user dependent variables and fields, that does a rather general task, then it is better to use default constructors, where you do not have to assign any user variables to class variables, and have to just call some basic instantiating functions depending on class requirements and functions.
- (b) **Parameterized Constructor:** Say there are variables that the user has entered that need to be assigned to the class object, or there are certain properties of each object different from other objects of the same class, like in enemies in a game, or employees in a Company, each object can be initialized with a set of variables. In this situation it is better to just use a parameterized constructor.
- (c) **Copy Constructor:** If you have many constructors that are often similar in definition and declaration, but have very few dissimilar properties, it is better to use copy constructors. For example Trees in a RPG game, where each tree has the same basic structure, but you might have small variation in just the height or the position of the tree.

4. Explain use of static member functions.

Static member functions in java are those that can be accessed by other classes without declaring an Object of that class. This is often why the main function needs to be public and static.

Every other member function needs to be accessed by an object of that class, as opposed to static ones.

In terms of memory, the static keyword in C++ is used when you have a variable that needs to be accessed by several objects of the same class, and this variable doesn't need to be different for each object. An example would be the Security number of an Employee, which just needs to be incremented as an object is created. It is accessed by each object, and therefore it makes sense for it to be declared in a way where it does not get copied for each object, thereby saving space.

5. How java program is executed?
6. What is the use of JVM?
7. What are the different control statements used in C++ and Java?
8. Write couple of examples/applications suitable to use OOP concepts specially use of classes, objects and constructors.
 - (a) Game Development: Enemies, walls, obstacles, trees, NPCs, are often structured as classes. This is because they have a set template that each member follows, and there are often many of them in a game. This makes OOP the perfect choice.
 - (b) Machine Learning: Machine learning often requires extensive and complex algorithms that need to be written and applied on a set of data. If such algorithms are put together as classes, then their objects can be fed that data and that algorithm can be run on it efficiently and easily, as opposed to writing it every time for each data set.
 - (c) Software Development: GUI components like buttons, sliders, panels, frames, bars, etc often have a singular functionality associated with them. As there are many such components in a GUI, it makes sense to make them into classes, and spawn their objects in various meaningful positions in the UI.