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

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

3.1 Algorithm

An *Algorithm* is a set of statements for solving a problem. They're the building blocks for programming, and they allow things like computers, smartphones, and websites to function and make decisions.

It looks like this, and can be generally implemented in any suitable programming language.

```
step 1: Start
step 2: Do something
step 3: Do something else
step 4: Return result
step 5: Stop
```

3.2 Class

1. 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).
2. It is a basic concept of Object-Oriented Programming which revolve around the real-life entities.
3. It is like a template for creating objects. You can initialize various variables of varying data types in it.
4. The Syntax for C++ and Java is given below.

```cpp
// Syntax in C++
class Name_of_Class
{
  private:
    int a;
  public:
    int b;
  protected:
    int c;
    data_type member_function()
    {
      return data_type;
    }
}objects;
```

```cpp
// Syntax in Java
class Name_of_Class
{
  private:
    int a;
  public:
    int b;
  protected:
    int c;
    data_type member_function()
    {
      return data_type;
    }
}
```

3.3 Objects

1. *Object* is an instance of a *class*. An object in *OOP* is nothing but a self-contained component which consists of methods and properties to make a particular type of data useful.

2. For example color name, table, bag, barking. When you send a message to an object, you are asking the object to invoke or execute one of its methods as defined in the class.

3. From a programming point of view, an object in OOPS can include a data structure, a variable, or a function. It has a memory location allocated. Java Objects are designed as class hierarchies.

4. The syntax for Java and C++ are given below.

```cpp
// C++ Syntax
class_name obj;
Employee CEO;
Employee President(Name, Age);
```

```cpp
// Java Syntax
class_name obj = new class_name;
Employee CEO = new Employee();
Employee President = new Employee(Name, Age);
```

3.4 Default, Parameterized, and Copy Construtor

A constructor is a special member function with exact same name as the class name.
There are **3** Types of Constructors:

1. **Default Constructor**: A constructor with no arguments (or parameters) in the definition is a default constructor. Usually, these constructors use to initialize data members (variables) with real values. If no constructor is explicitly declared, the compiler automatically creates a default constructor with no data member (variables) or initialization.
2. **Parameterized Constructor**: Unlike Default constructor, It contains parameters (or arguments) in the constructor definition and declaration. More than one argument can also pass through a parameterized constructor. Moreover, these types of constructors use for overloading to differentiate between multiple constructors.
3. **Copy Constructor**: A copy constructor is a member function that initializes an object using another object of the same class. It helps to copy data from one object to another.
4. The Syntax for Each of them is same for C++ and Java and is given below.

```
class Avenger
{
  static int hero = 0;
  int age;
  bool from_earth, is_a_God;
  string Superpower;

  // default Constructor
  Avenger()
  {
    hero++;
  }

  // Parameterized Constructor
  Avenger(bool from_earth, bool is_a_God, int age, String name, string
  Superpower)
  {
    if(!from_earth)
    {
      alien_avengers++;
    }
  }

  // Copy Constructor

  Avenger(Avenger &new_Avenger)
  {
    this.superpower = new_Avenger.superpower;
  }
}
```

3.5 Destuctors

Destructor is an instance member function which is invoked automatically whenever an object is going to be destroyed. Meaning, a destructor is the last function that is going to be called before an object is destroyed.

1. Destructor is also a *special member* function like constructor. Destructor destroys the class objects created by constructor.
2. Destructor has the same name as their class name preceded by a tiled ( ) symbol.
3. It is not possible to define more than one destructor.
4. The destructor is only one way to destroy the object create by constructor. Hence destructor can-not be overloaded.
5. Destructor neither requires any argument nor returns any value.
6. It is automatically called when object goes out of scope.
7. Destructor release memory space occupied by the objects created by constructor.
8. In destructor, objects are destroyed in the reverse of an object creation.

In Java, The job of the constructor is done by the compiler, as garbage collection in general is done better by the compiler in java than the programmer. It is for this reason that the finalize() function exists in java but is depricated and its strongly recommended not to use it.

```
// C++ Implementation
class Avenger()
{
  ~Avenger()
  {
    pay_respects();
  }

  // in java
  finalize()
  {
    System.gc() // call the garbage collector
  }
}
```

3.6 Dynamic Allocation and DeAllocation

1. Many times, you are not aware in advance how much memory you will need to store particular information in a defined variable and the size of required memory can be determined at run time.
2. You can allocate memory at run time within the heap for the variable of a given type using a special operator in C++ which returns the address of the space allocated. This operator is called new operator.
3. If you are not in need of dynamically allocated memory anymore, you can use delete operator, which de-allocates memory that was previously allocated by new operator.
4. The Syntax for C++ and Java is given below.

```
1
2  // C++
3  int a = new int;
4  delete a;
5
6  // Java
7  Employee Obj = new Employee()
```

4 Algorithm

1. **Start**
2. Create Employee Class
3. Delcare appropirate Data Memebers and define the member funtions
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. Distory 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

```java
package assignment_1;
import java.util.Scanner;

public class Employee {

    // create an object of Scanner
    Scanner input = new Scanner(System.in);

    int emp_id;
    int age, basic_sal, da, ta;
    String address_city, position, name;
    static int ssn;

    Employee()
    {
        System.out.println("Default Constructor was called");
    }

    // Parameterized Constructor
    Employee(int e, int a, int b, int d,
    int t, String add, String pos, String nam)
    {
        System.out.println("Parameterized constructor was called");
        emp_id = e;
        age = a;
        basic_sal = b;
        da = d;
        ta = t;
        address_city = add;
        position = pos;
        name = nam;
    }

    // Copy Constructor
    Employee(Employee E)
    {
        System.out.println("Copy constructor was called");
        emp_id = E.emp_id;
        age = E.age;
        basic_sal = E.basic_sal;
        da = E.da;
        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     //     }
```

```
    //      finally
    //      {
    //          super.finalize();
    //      }
    // }

}
```

Listing 1: Employee.java

```
package assignment_1;
import java.util.Scanner;


public class Source {

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

    public static void main(String[] args) {
        System.out.println("This is the first Assignment");
        int count = 0;

        // Default Constructor
        Employee CEO = new Employee();
        CEO.emp_id = 1000;
        CEO.name = "Kom Pany Seeio";
        CEO.address_city = "Seoul";
        CEO.age = 45;
        CEO.basic_sal = 1000000;
        CEO.da = 1000;
        CEO.ta = 2000;
        CEO.position = "CEO";

        // Defining an object using the copy constructor
        Employee President = new Employee(CEO);
        President.name = "Precy Dent";
        President.age = 45;
        President.address_city = "Delhi";
        President.basic_sal *= 2;
        President.position = "President";

        Employee VP = new Employee(1003, 50, 200000, 3000, 1000,
         "Mumbai", "Vice President", "Visey Presed Ent");


        System.out.println("Information about the CEO");
        CEO.display();
        System.out.println("Information about the President");
        President.display();
        System.out.println("Information about the President");
        VP.display();

        System.out.println("Enter the number of employees :");
        count = input.nextInt();
        Employee objs[] = new Employee[count];

        for(int i = 0;i < count;i++)
        {
            objs[i] = new Employee();
```

```
            objs[i].accept();
            objs[i].display();
        }
        System.gc();
    }
}
```

Listing 2: Source.java

9.1.1 Java Input

```
Enter the number of employees :
2
Default Constructor was called
Enter the age :
35
Employee ID is:
006
Employee Name:
Peter
Employee Age:
17
Employee Position:
Avenger
Employee basic Salary:
500000
Employee DA:
3440
Employee TA:
3550
Employee Address City:
Brooklyn


Default Constructor was called
Enter the age :
4
Employee ID is:
007
Employee Name:
Thor
Employee Age:
1500
Employee Position:
Avenger
Employee basic Salary:
6000000
Employee DA:
3000
Employee TA:
5000
Employee Address City:
Asgard
```

Listing 3: Python example

9.1.2 Java Output

```
This is the first Assignment
Default Constructor was called
Copy constructor was called
Parameterized constructor was called
Information about the CEO
Employee ssn is: 1
Employee ID is : 1000
Employee Name: Kom Pany Seeio
Employee Age: 45
Employee Position: CEO
Employee basic Salary: 1000000
Employee DA: 1000
Employee TA: 2000
Employee Gross Salary: 853000.0
Employee Address City: Seoul


Information about the President
Employee ssn is: 2
Employee ID is : 1000
Employee Name: Precy Dent
Employee Age: 45
Employee Position: President
Employee basic Salary: 2000000
Employee DA: 1000
Employee TA: 2000
Employee Gross Salary: 1703000.0
Employee Address City: Delhi


Information about the President
Employee ssn is: 3
Employee ID is : 1003
Employee Name: Visey Presed Ent
Employee Age: 50
Employee Position: Vice President
Employee basic Salary: 200000
Employee DA: 3000
Employee TA: 1000
Employee Gross Salary: 174000.0
Employee Address City: Mumbai


Employee ssn is: 4
Employee ID is : 6
Employee Name: Peter
Employee Age: 17
Employee Position: Avenger
Employee basic Salary: 500000
Employee DA: 3440
Employee TA: 3550
Employee Gross Salary: 431990.0
Employee Address City: Brooklyn


Employee ssn is: 5
Employee ID is : 7
Employee Name: Thor
Employee Age: 1500
```

```
Employee Position: Avenger
Employee basic Salary: 6000000
Employee DA: 3000
Employee TA: 5000
Employee Gross Salary: 5108000.0
Employee Address City: Asgard
```

Listing 4: Python example

9.2 C++ Implementation

```cpp
#include <iostream>
using namespace std;
class Employee
{
private:
    static int ssn;
protected:
    int something = 5;

public:
    int emp_id = 1000;
    int age = 0, basic_sal = 0, da = 0, ta = 0;
    string address_city, position, name;

    // Default Constructor
    Employee()
    {
        ssn += 1;
        cout << "The Default Constructor was called" << endl;
    }

    // Parameterized Constructor
    Employee(int e, int a, int b, int d, int t, string add, string pos, string nam
    )
    {
        cout << "Parameterized constructor was called\n";
        emp_id = e;
        age = a;
        basic_sal = b;
        da = d;
        ta = t;
        address_city = add;
        position = pos;
        name = nam;
    }

    // Copy Constructor
    Employee(Employee &E)
    {
        cout << "Copy Constructor was called" << endl;
        emp_id = E.emp_id;
        age = E.age;
        basic_sal = E.basic_sal;
        da = E.da;
        ta = E.ta;
        address_city = E.address_city;
        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";
```

```
    CEO.address_city = "Seoul";
    CEO.age = 45;
    CEO.basic_sal = 1000000;
    CEO.da = 1000;
    CEO.ta = 2000;
    CEO.position = "CEO";


    // Defining an object using the copy constructor
    Employee President(CEO);
    President.name = "Precy Dent";
    President.age = 45;
    President.address_city = "Delhi";
    President.basic_sal *= 2;
    President.position = "President";

    // Defining anothe object using the parameterized constructor
    Employee VP(1003, 50, 200000, 3000, 1000, "Mumbai", "Vice President", "Visey
    Presed Ent");

    // Taking user input for the number of employees
    int count = 0;
    cout << "How many values do you wanna input ? ";
    cin >> count;

    // Dynamically Creating new Objects
    Employee *Obj = new Employee[count];
    cout << "Enter the Details" << endl;

    // Accepting Objects from the User.
    for (int i = 0; i < count; i++)
    {
        cout << "Enter information about the Employee Number: " << i + 1 << endl;
        Obj[i].accept();
    }

    // Displaying Information about the CEO and the President
    cout << "Information about the CEO" << endl;
    CEO.display();
    cout << "Information about the President" << endl;
    President.display();
    cout << "Information about the Vice President" << endl;
    VP.display();

    // Displaying Information about the other Employees
    for (int i = 0; i < count; i++)
    {
        cout << "Information about the Employee Number: " << i + 1 << endl;
        Obj[i].display();
    }

    delete[] Obj;
    return 0;
}
```

Listing 5: Main.Cpp

9.2.1 C++ Input

```
The Default Constructor was called
Copy Constructor was called
Parameterized constructor was called
How many values do you wanna input ? 2
The Default Constructor was called
The Default Constructor was called

Enter the Details

Enter information about the Employee Number: 1
Enter the Employee ID:
005
Enter the Employee Name:
Tony
Enter the Employee Age:
40
Enter the Employee Position:
Philanthropist
Enter the Employee basic Salary:
5000000
Enter the Employee DA:
3000
Enter the Employee TA:
3000
Enter the Employee Address City:
NewYork

Enter information about the Employee Number: 2
Enter the Employee ID:
006
Enter the Employee Name:
Steve
Enter the Employee Age:
105
Enter the Employee Position:
Captain
Enter the Employee basic Salary:
600000
Enter the Employee DA:
2999
Enter the Employee TA:
2000
Enter the Employee Address City:
Brooklyn
```

Listing 6: C++ Input

9.2.2 C++ Output

```
Information about the CEO
Employee ssn is:1003
Employee ID is : 1000
Employee Name: Kom Pany Seeio
Employee Age: 45
Employee Position: CEO
Employee basic Salary: 1000000
Employee DA: 1000
Employee TA: 2000
```

```
Employee Gross Salary: 853000
Employee Address City: Seoul

Information about the President
Employee ssn is:1003
Employee ID is : 1000
Employee Name: Precy Dent
Employee Age: 45
Employee Position: President
Employee basic Salary: 2000000
Employee DA: 1000
Employee TA: 2000
Employee Gross Salary: 1.703e+06
Employee Address City: Delhi

Information about the Vice President
Employee ssn is:1003
Employee ID is : 1003
Employee Name: Visey Presed Ent
Employee Age: 50
Employee Position: Vice President
Employee basic Salary: 200000
Employee DA: 3000
Employee TA: 1000
Employee Gross Salary: 174000
Employee Address City: Mumbai

Information about the Employee Number: 1
Employee ssn is:1003
Employee ID is : 5
Employee Name: Tony
Employee Age: 40
Employee Position: Philanthropist
Employee basic Salary: 5000000
Employee DA: 3000
Employee TA: 3000
Employee Gross Salary: 4.256e+06
Employee Address City: NewYork

Information about the Employee Number: 2
Employee ssn is:1003
Employee ID is : 6
Employee Name: Steve
Employee Age: 105
Employee Position: Captain
Employee basic Salary: 600000
Employee DA: 2999
Employee TA: 2000
Employee Gross Salary: 514999
Employee Address City: Brooklyn

The Destructor was called
The Destructor was called
The Destructor was called
The Destructor was called
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 contigiously 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 arguements. 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 intantiating 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 defintion 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 obejct 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 doesnt 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?**
   Java, being a *platform-independent programming language*, doesn't work on the one-step compilation. Instead, it involves a two-step execution, first through an OS-independent compiler; and second, in a virtual machine (JVM) which is custom-built for every operating system. First, the source '.java' file is passed through the compiler, which then encodes the source code into a machine-independent encoding, known as Bytecode. The content of each class contained in the source file is stored in a separate '.class' file.

   The class files generated by the compiler are independent of the machine or the OS, which allows them to be run on any system. To run, the main class file (the class that contains the method main) is passed to the JVM and then goes through three main stages before the final machine code is executed.

6. **What is the use of JVM?**
   Java Virtual Machine, or JVM, loads, verifies and executes Java bytecode. It is known as the interpreter or the core of Java programming language because it executes Java programming.

   JVM is specifically responsible for converting bytecode to machine-specific code and is necessary in both JDK and JRE. It is also platform-dependent and performs many functions, including memory management and security. In addition, JVM can run programs written in other programming languages that have been translated to Java bytecode.

7. **What are the different control statements used in C++ and Java?**
   There are several Control statements in Java and C++

   (a) Loops like for, while and do while

   (b) Logical Control statements like if, else if, else, switch statements

   (c) Ternary Operator as another form of logical operation

   (d) Branching Statements include Keywords like break and continue.

   Examples:

```
1    // Loops
2    for(int i = 0; i < 5; i++)
3    {
4      cout<<"This is a basic for loop, and exists in both cpp and java.";
5    }
6    do
7    {
8      cout<<"This is a do while loop";
```

```
    }while(condition);
    while(true)
    {
      cout<<"This is a while loop";
    }

    // Logical Statements
    if(condition)
    {
      cout<<"Condition true";
    }
    else cout << "Condition false";

    condition = condition ? true : false;

    // Branching statements:
    switch(condition)
    {
      case 1: cout<<"Case 1 is being executed";
        break;
      case 2: cout<<"Case 2 is being executed";
        break;
      default: cout<<"Default case";
        break;
    }

```

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.