MIT World Peace University

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

Implementation of Inheritance using C++ and Java

Practical Report

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 20

October 5, 2022

Contents

1 Aim and Objectives

Aim

To Implement inheritance using C++ and Java (with interfaces).

Objectives

1. To understand the inheritance or is-A relationship concept.
2. To understand code reusability.
3. To learn implementation of interfaces in java.

2 Problem Statements

2.1 Problem 1 in C++

Design and develop inheritance for a given case study, identify objects and relationships and implement inheritance wherever applicable using C++.
Employee class has Empname, Empid, Address, Mailid, and Mobileno as data members. Add the Following Classes

- Programmer
- Team Leader
- Assistant Project Manager
- Project Manager from Employee Class

Add Basic Pay as the member of all the inherited classes with 97 % of the Basic Pay as DA, 10 % of Basic Pay as HRA, 12 % of Basic Pay as PF, 0.1 % of Basic Pay for staff club fund.

Generate Pay slips for the Employees with their gross and net salaries.

2.2 Problem 2 in Java

Write a Java Program for demonstrating Inheritance in Java. Write a program in Java showing hierarchical inheritance with base class as Employee and derived classes as FullTimeEmployee and InternEmployee with methods DisplaySalary in base class and CalculateSalary in derived classes. Calculate salary method will calculate as per increment given to fulltime and intern Employees. Fulltime employee- 50% hike, Intern employee-25% hike. Display salary before and after hike.

2.3 Probelm 3 in Java

Write a java program to create two interfaces Motorbike and Cycle.

- Motorbike interface consists of the attribute speed.
- The method is totalDistance().
- Cycle interface consists of the attributes distance and the method speed().
- These interfaces are implemented by the class TwoWheeler.
- Calculate total distance travelled and Average Speed maintained by Two Wheeler.

3 Theory

3.1 Concept of Inheritance and its types

Inheritance is a process in which one object acquires all the properties and behaviors of its parent object automatically. In such way, you can *reuse, extend, or modify* the attributes and behaviors which are defined in other class.

In C++, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. The derived class is the specialized class for the base class. Most ideas of inheritance are directly applicable in Java as well.

3.1.1 Advantages

1. Minimizing duplicate code: Key benefits of Inheritance include minimizing the identical code as it allows sharing of the common code among other subclasses.
2. Flexibility: Inheritance makes the code flexible to change, as you will adjust only in one place, and the rest of the code will work smoothly.
3. Overriding: With the help of Inheritance, you can override the methods of the base class.
4. Data Hiding: The base class in Inheritance decides which data to be kept private, such that the derived class will not be able to alter it.

3.1.2 Types of Inheritance

1. Single inheritance : It is defined as the inheritance in which a derived class is inherited from the only one base class.
2. Multiple inheritance : It is the process of deriving a new class that inherits the attributes from two or more classes.
3. Hierarchical inheritance : When one class inherits another class which is further inherited by another class, it is known as multi level inheritance in C++. Inheritance is transitive so the last derived class acquires all the members of all its base classes.
4. Multilevel inheritance: It is a process of deriving a class from another derived class.
5. Hybrid inheritance : Any legal combination of any of the above inheritance techniques would be known as hybrid inheritance.

Let us Look an Examples of these.

```
class A
{
  // Base Class
};

class D
{
  // Base Class 2
}
class B : public A
{
  // Single Inheritance
```

```
  }
  class C : public B
  {
    // Multi Level Inheritance
  }
  class E : public A, public D
  {
    // Multiple Inheritance
  }
  class F : public E, private D
  {
    // hierarchical inheritance
  }
```

3.1.3 Base class and Derived Class Constructors

Whenever we create an object of a class, the default constructor of that class is invoked automatically to initialize the members of the class.

If we inherit a class from another class and create an object of the derived class, it is clear that the default constructor of the derived class will be invoked but before that the default constructor of all of the base classes will be invoke, i.e the order of invocation is that the base class's default constructor will be invoked first and then the derived class's default constructor will be invoked.

When a class is inherited from other, The data members and member functions of base class comes automatically in derived class based on the access specifier but the definition of these members exists in base class only. So when we create an object of derived class, all of the members of derived class must be initialized but the inherited members in derived class can only be initialized by the base class's constructor as the definition of these members exists in base class only.

This is why the constructor of base class is called first to initialize all the inherited members.

Let us see an Example

```

// base class
class Parent
{
    public:

    // base class constructor
    Parent()
    {
        cout << "Inside base class" << endl;
    }
};

// sub class
class Child : public Parent
{
    public:

    //sub class constructor
    Child()
```

```
    {
        cout << "Inside sub class" << endl;
    }
};

// main function
int main() {

    // creating object of sub class
    Child obj;

    return 0;
}

```

Output would be

```
Inside base class
Inside sub class
```

4 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

5 Input

For C++

1. Number of Each Type of Employee
2. Name, Age, Address City, and Salary of Each Employee

For Java

1. The Information and Salary about the Full Time Employee
2. The Information and Salary of the Intern Employee
3. Speed, Time and Distance

6 Output

For C++

1. General Information about Each Employee
2. The Gross Salary of Each Employee
3. The Net Salary of Each Employee

For Java

1. The General information about the Full time and the Intern Employee
2. The Hiked salaries of both the Intern Employee, and the Full Time Employee.
3. Speed and Distance.

7 Code

7.1 C++ Implementation for Problem 1

```cpp
#include <iostream>
using namespace std;

class Employee
{

protected:
    static int ssn;
    int emp_id = 1000;
    int age = 0;
    double basic_sal = 0, da = 0, ta = 0, gross_sal = 0, net_sal = 0;
    string address_city, position, name;

public:
    // Default Constructor
    Employee()
    {
        cout << "The Default Constructor was called" << endl;
    }

    // Parameterized Constructor
    Employee(int e, int a, string add, string nam)
    {
        cout << "Parameterized constructor was called\n";
        emp_id = e;
        age = a;
        address_city = add;
        name = nam;
    }

    // Copy Constructor
    Employee(Employee &E)
    {
        cout << "Copy Constructor was called" << endl;
        emp_id = E.emp_id;
        age = E.age;
        address_city = E.address_city;
        name = E.name;
    }

    void display()
    {
        Employee::ssn++;
        cout << "Employee ssn is:" << ssn << endl;
        cout << "Employee ID is : " << emp_id << endl;
```

```
46          cout << "Employee Name: " << name << endl;
47          cout << "Employee Age: " << age << endl;
48          cout << "Employee Address City: " << address_city << endl;
49      }
50
51      void accept()
52      {
53          cout << "Enter the Employee ID: " << endl;
54          cin >> emp_id;
55          cout << "Enter the Employee Name: " << endl;
56          cin >> name;
57          cout << "Enter the Employee Age: " << endl;
58          cin >> age;
59          cout << "Enter the Employee Address City: " << endl;
60          cin >> address_city;
61      }
62
63      // Destructor
64      ~Employee()
65      {
66          cout << "The Destructor was called" << endl;
67      }
68  };
69
70  int Employee::ssn = 1000;
71
72  class Programmer : public Employee
73  {
74
75  protected:
76      double da = 0, hra = 0, pf = 0, scf = 0;
77
78  public:
79      void calc_gross_sal()
80      {
81          da = 0.97 * basic_sal;
82          hra = basic_sal;
83          pf = basic_sal;
84          scf = basic_sal;
85          gross_sal = da + hra + pf + scf + basic_sal;
86      }
87
88      void calc_net_sal()
89      {
90          // Reducing Income Taxes
91          net_sal = gross_sal - (0.15) * gross_sal;
92      }
93
94      void accept()
95      {
96          Employee::accept();
97          cout << "Enter the basic Salary of the Programmer : " << endl;
98          cin >> basic_sal;
99          calc_gross_sal();
100         calc_net_sal();
101     }
102
103     void display()
104     {
```

```
105         Employee::display();
106         cout << "The Gross Salary is: " << gross_sal << endl;
107         cout << "The Net Salary is: " << net_sal << endl;
108     }
109 };
110 
111 class TeamLeader : public Employee
112 {
113 
114 protected:
115     double da = 0, hra = 0, pf = 0, scf = 0;
116 
117 public:
118     void calc_gross_sal()
119     {
120         da = 0.97 * basic_sal;
121         hra = basic_sal;
122         pf = basic_sal;
123         scf = basic_sal;
124         gross_sal = da + hra + pf + scf + basic_sal;
125     }
126 
127     void calc_net_sal()
128     {
129         // Reducing Income Taxes
130         net_sal = gross_sal - (0.15) * gross_sal;
131     }
132 
133     void accept()
134     {
135         Employee::accept();
136         cout << "Enter the basic Salary of the Team Leader : " << endl;
137         cin >> basic_sal;
138         calc_gross_sal();
139         calc_net_sal();
140     }
141 
142     void display()
143     {
144         Employee::display();
145         cout << "The Gross Salary is: " << gross_sal << endl;
146         cout << "The Net Salary is: " << net_sal << endl;
147     }
148 };
149 
150 class AssistantProjectManager : public Employee
151 {
152 
153 protected:
154     double da = 0, hra = 0, pf = 0, scf = 0;
155 
156 public:
157     void calc_gross_sal()
158     {
159         da = 0.97 * basic_sal;
160         hra = basic_sal;
161         pf = basic_sal;
162         scf = basic_sal;
163         gross_sal = da + hra + pf + scf + basic_sal;
```

```
164     }
165
166     void calc_net_sal()
167     {
168         // Reducing Income Taxes
169         net_sal = gross_sal - (0.15) * gross_sal;
170     }
171
172     void accept()
173     {
174         Employee::accept();
175         cout << "Enter the basic Salary of the Assistant Project Manager :" <<
    endl;
176         cin >> basic_sal;
177         calc_gross_sal();
178         calc_net_sal();
179     }
180
181     void display()
182     {
183         Employee::display();
184         cout << "The Gross Salary is: " << gross_sal << endl;
185         cout << "The Net Salary is: " << net_sal << endl;
186     }
187 };
188
189 class ProjectManager : public Employee
190 {
191
192 protected:
193     double da = 0, hra = 0, pf = 0, scf = 0;
194
195 public:
196     void calc_gross_sal()
197     {
198         da = 0.97 * basic_sal;
199         hra = basic_sal;
200         pf = basic_sal;
201         scf = basic_sal;
202         gross_sal = da + hra + pf + scf + basic_sal;
203     }
204
205     void calc_net_sal()
206     {
207         // Reducing Income Taxes
208         net_sal = gross_sal - (0.15) * gross_sal;
209     }
210
211     void accept()
212     {
213         Employee::accept();
214         cout << "Enter the basic Salary of the Project Manager :" << endl;
215         cin >> basic_sal;
216         calc_gross_sal();
217         calc_net_sal();
218     }
219
220     void display()
221     {
```

```
222         Employee::display();
223         cout << "The Gross Salary is: " << gross_sal << endl;
224         cout << "The Net Salary is: " << net_sal << endl;
225     }
226 };
227 
228 int main()
229 {
230     cout << "Welcome to Employee Payroll Management System" << endl
231          << endl;
232 
233     int choice = 1, number = 1;
234 
235     do
236     {
237         cout << "\n\nWhose Details do you wanna enter? " << endl;
238         cout << "1. Programmer\n2. Team Leader\n3. Assistant Project Manager\n4.
    Project Manager\n5. Quit\n";
239         cin >> choice;
240 
241         if (choice == 1)
242         {
243             cout << "How many Programmers are we talking? ";
244             cin >> number;
245             Programmer pr[number];
246             for (int i = 0; i < number; i++)
247             {
248                 cout << "Enter the Information about the Programmer" << endl;
249                 pr[i].accept();
250             }
251             cout << "\nHere is their Information and their Pay Slips" << endl;
252             cout << endl
253                  << endl;
254 
255             cout << "Programmer" << endl;
256 
257             for (int i = 0; i < number; i++)
258             {
259                 cout << "Info and Pay Slip of Programmer " << i + 1 << endl;
260                 pr[i].display();
261                 cout << endl;
262             }
263         }
264         else if (choice == 2)
265         {
266             cout << "How many Team Leaders are we talking? ";
267             cin >> number;
268             TeamLeader tl[number];
269             for (int i = 0; i < number; i++)
270             {
271                 cout << "Enter the Information about the Team Leader " << i + 1 <<
    endl;
272                 tl[i].accept();
273             }
274             cout << "Here is their Information and their Pay Slips" << endl;
275             cout << endl
276                  << endl;
277             for (int i = 0; i < number; i++)
278             {
```

```
                cout << "Info and Pay Slip of Team Leader " << i + 1 << endl;
                tl[i].display();
                cout << endl;
            }
            cout << endl
                 << endl;
        }
        else if (choice == 3)
        {
            cout << "How many Assistant Project Managers are we talking? ";
            cin >> number;
            AssistantProjectManager ap[number];
            for (int i = 0; i < number; i++)
            {
                cout << "Enter the Information about the Assitant Project Manager " << i + 1 << endl;
                ap[i].accept();
            }
            cout << "Here is their Information and their Pay Slips" << endl;
            cout << endl
                 << endl;
            for (int i = 0; i < number; i++)
            {
                cout << "Info and Pay Slip of Assitant Project Manager " << i + 1 << endl;
                ap[i].display();
                cout << endl;
            }
            cout << endl
                 << endl;
        }

        else if (choice == 4)
        {
            cout << "How many Project Managers are we talking? ";
            cin >> number;
            ProjectManager pm[number];
            for (int i = 0; i < number; i++)
            {
                cout << "Enter the Information about the Project Manager " << i + 1 << endl;
                pm[i].accept();
            }
            cout << "Here is their Information and their Pay Slips" << endl;
            cout << endl
                 << endl;
            for (int i = 0; i < number; i++)
            {
                cout << "Info and Pay Slip of Project Manager " << i + 1 << endl;
                pm[i].display();
                cout << endl;
            }
        }
    } while (choice != 5);

    return 0;
}
```

Listing 1: Main.Cpp

7.1.1 C++ Input and Output

```
1
2  Welcome to Employee Payroll Management System
3
4  Whose Details do you wanna enter?
5  1. Programmer
6  2. Team Leader
7  3. Assistant Project Manager
8  4. Project Manager
9  5. Quit
10 1
11 How many Programmers are we talking? 1
12 The Default Constructor was called
13 Enter the Information about the Programmer
14 Enter the Employee ID:
15 1001
16 Enter the Employee Name:
17 Tony
18 Enter the Employee Age:
19 45
20 Enter the Employee Address City:
21 Berlin
22 Enter the basic Salary of the Programmer :
23 450000
24
25 Here is their Information and their Pay Slips
26
27
28 Programmer
29 Info and Pay Slip of Programmer 1
30 Employee ssn is:1001
31 Employee ID is : 1001
32 Employee Name: Tony
33 Employee Age: 45
34 Employee Address City: Berlin
35 The Gross Salary is: 2.2365e+06
36 The Net Salary is: 1.90102e+06
37
38 The Destructor was called
39
40
41 Whose Details do you wanna enter?
42 1. Programmer
43 2. Team Leader
44 3. Assistant Project Manager
45 4. Project Manager
46 5. Quit
47 2
48 How many Team Leaders are we talking? 1
49 The Default Constructor was called
50 Enter the Information about the Team Leader 1
51 Enter the Employee ID:
52 1002
53 Enter the Employee Name:
54 Steve
55 Enter the Employee Age:
56 80
57 Enter the Employee Address City:
```

```
Queens
Enter the basic Salary of the Team Leader :
70000
Here is their Information and their Pay Slips


Info and Pay Slip of Team Leader 1
Employee ssn is:1002
Employee ID is : 1002
Employee Name: Steve
Employee Age: 80
Employee Address City: Queens
The Gross Salary is: 347900
The Net Salary is: 295715



The Destructor was called


Whose Details do you wanna enter?
1. Programmer
2. Team Leader
3. Assistant Project Manager
4. Project Manager
5. Quit
3
How many Assistant Project Managers are we talking? 1
The Default Constructor was called
Enter the Information about the Assitant Project Manager 1
Enter the Employee ID:
1003
Enter the Employee Name:
Caulson
Enter the Employee Age:
60
Enter the Employee Address City:
Delhi
Enter the basic Salary of the Assistant Project Manager :
60000
Here is their Information and their Pay Slips


Info and Pay Slip of Assitant Project Manager 1
Employee ssn is:1003
Employee ID is : 1003
Employee Name: Caulson
Employee Age: 60
Employee Address City: Delhi
The Gross Salary is: 298200
The Net Salary is: 253470



The Destructor was called


Whose Details do you wanna enter?
1. Programmer
```

```
2. Team Leader
3. Assistant Project Manager
4. Project Manager
5. Quit
4
How many Project Managers are we talking? 1
The Default Constructor was called
Enter the Information about the Project Manager 1
Enter the Employee ID:
1005
Enter the Employee Name:
Fury
Enter the Employee Age:
56
Enter the Employee Address City:
Pune
Enter the basic Salary of the Project Manager :
600000
Here is their Information and their Pay Slips


Info and Pay Slip of Project Manager 1
Employee ssn is:1004
Employee ID is : 1005
Employee Name: Fury
Employee Age: 56
Employee Address City: Pune
The Gross Salary is: 2.982e+06
The Net Salary is: 2.5347e+06

The Destructor was called


Whose Details do you wanna enter?
1. Programmer
2. Team Leader
3. Assistant Project Manager
4. Project Manager
5. Quit
5
```

Listing 2: C++ Output

7.2 Java Implementation of Problem 2

```java
package assignment_2;
import java.util.Scanner;

class Employee
{
    Scanner input = new Scanner(System.in);
    String name;
    int emp_id;
    double salary;
    int hike = 0;

    final static int full_time_hike_perc = 100;
    final static int intern_hike_perc = 50;
```

```
14
15      void accept()
16      {
17          System.out.println("Enter the Employee Name");
18          name = input.next();
19          System.out.println("Enter The Employee ID");
20          emp_id = input.nextInt();
21          System.out.println("Enter The Employee Salary");
22          salary = input.nextInt();
23      }
24
25      void display_salary()
26      {
27          System.out.println("The Employee Name is: " + name);
28          System.out.println("The Employee ID is: " + emp_id);
29      }
30  }
```

Listing 3: Employee.java

```
1  package assignment_2;
2
3  class InternEmployee extends Employee {
4      double hiked_salary = 0;
5
6      void calculate_salary() {
7          hiked_salary = salary + salary * (full_time_hike_perc / 100);
8      }
9
10     @Override
11     void display_salary() {
12         super.display_salary();
13         System.out.println("The Salary Before the Hike for this Intern Employee is
       : " + salary);
14         calculate_salary();
15         System.out.println("The Salary after the Hike for this Intern Employee is
       : " + hiked_salary);
16     }
17 }
```

Listing 4: Intern Employee.java

```
1  package assignment_2;
2
3  class FullTimeEmployee extends Employee {
4      double hiked_salary = 0;
5
6      void calculate_salary() {
7          hiked_salary = salary + salary * (full_time_hike_perc / 100);
8      }
9
10     @Override
11     void display_salary() {
12         super.display_salary();
13         System.out.println("The Salary Before the Hike for this Full time Employee
       is: " + salary);
14         calculate_salary();
15         System.out.println("The Salary after the Hike for this Full time Employee
       is : " + hiked_salary);
```

```
    }
}
```

Listing 5: Full Time Employee.java

```
package assignment_2;

import java.util.Scanner;

class Main {
    public static void main(String args[]) {
        System.out.println("Welcome to Salary Hiking Program");
        Employee emp = new Employee();
        FullTimeEmployee full_time_emp = new FullTimeEmployee();
        System.out.println("Enter the Details about the Full Time Employee: \n");
        full_time_emp.accept();
        full_time_emp.display_salary();
        InternEmployee intern_emp = new InternEmployee();
        System.out.println("Enter the Details about the Intern Employee: \n");
        intern_emp.accept();
        intern_emp.display_salary();
    }
}
```

Listing 6: Main.java

7.2.1 Java Output for Problem 2

```
Welcome to Salary Hiking Program
Enter the Details about the Full Time Employee:

Enter the Employee Name
Tony
Enter The Employee ID
1001
Enter The Employee Salary
100000
The Employee Name is: Tony
The Employee ID is: 1001
The Salary Before the Hike for this Full time Employee is: 100000.0
The Salary after the Hike for this Full time Employee is : 200000.0
Enter the Details about the Intern Employee:

Enter the Employee Name
Steve
Enter The Employee ID
1002
Enter The Employee Salary
50000
The Employee Name is: Steve
The Employee ID is: 1002
The Salary Before the Hike for this Intern Employee is: 50000.0
The Salary after the Hike for this Intern Employee is : 100000.0
```

Listing 7: Java Output for Problem 2

7.3 Java Implementation of Problem 3 using Interfaces

```java
package assignment_2b;

import java.util.Scanner;

public class TwoWheeler implements MotorCycle, Cycle {
    public int speed = 0;
    public int time = 0;
    public int total_distance = 0;

    @Override
    public void speed() {
        System.out.println(total_distance / time);
    }

    @Override
    public void totalDistance() {
        System.out.println(speed * time);
    }
}
```

Listing 8: Two Wheeler.java

```java
package assignment_2b;

public interface MotorCycle {
    public abstract void totalDistance();
}
```

Listing 9: MotorCycle.java

```java
package assignment_2b;

public interface Cycle {
    public abstract void speed();
}
```

Listing 10: Cycle.java

```java
package assignment_2b;

import java.util.Scanner;

public class Main {

    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        TwoWheeler obj = new TwoWheeler();

        System.out.println("Calculating Speed: ");
        System.out.println("Enter the Distance Travelled by Your 2 Wheeler: ");
        obj.total_distance = input.nextInt();

        System.out.println("Enter the Time you Travelled on Your 2 Wheeler: ");
        obj.time = input.nextInt();

        System.out.println("The Speed is: ");
        obj.speed();

        System.out.println("Calculating Total Distance: ");
        System.out.println("Enter the Speed of Your 2 Wheeler: ");
```

```
        obj.speed = input.nextInt();

        System.out.println("Enter the Time you Travelled on Your 2 Wheeler: ");
        obj.time = input.nextInt();

        System.out.println("The Total Distance is: ");
        obj.totalDistance();
    }
}
```

Listing 11: Main.java

7.3.1 Java Output

```
Calculating Speed:
Enter the Distance Travelled by Your 2 Wheeler:
50
Enter the Time you Travelled on Your 2 Wheeler:
2
The Speed is:
25
Calculating Total Distance:
Enter the Speed of Your 2 Wheeler:
25
Enter the Time you Travelled on Your 2 Wheeler:
2
The Total Distance is:
50
```

Listing 12: Java Output for Program 3

8 Conclusion

Thus, learned to use reusability by applying concept of inheritance, interfaces and implemented solution of the given problem statement using C++ and Java.

9 FAQs

1. **Discuss ambiguity arises in multipath inheritance and how it is to be avoided in C++?**
   In *multiple* inheritances, when one class is derived from two or more base classes then there may be a possibility that the base classes have functions with the same name, and the derived class may not have functions with that name as those of its base classes.

   If the derived class object needs to access one of the similarly named member functions of the base classes then it results in ambiguity because the compiler gets confused about which base's class member function should be called.

   To solve this ambiguity scope resolution operator is used denoted by ' :: '

   The Syntax to use it is:

```
    ObjectName.ClassName::FunctionName();

```

   An Example:

```
// C++ program to resolve inheritance
// ambiguity

#include<iostream>
using namespace std;

// Base class A

class A {
    public:

    void func() {
        cout << " I am in class A" << endl;
    }
};

// Base class B

class B {
    public:

    void func() {
        cout << " I am in class B" << endl;
    }
};

// Derived class C
class C: public A, public B {


};
```

```
32
33 // Driver Code
34
35 int main() {
36
37     // Created an object of class C
38     C obj;
39
40     // Calling function func() in class A
41     obj.A::func();
42
43     // Calling function func() in class B
44     obj.B::func();
45
46     return 0;
47 }
48
49
```

2. **What's the difference between public, private, and protected?**
   public, private, and protected are known as access modifiers. Like their name suggests, they modify the access given to objects and subclasses.

   (a) If you do class child : **private** parent; then every private data member becomes inaccessible, coz anyway thats what should happen, then the protected data members become private, and public data members also become private.

   (b) If you do class child : **protected** parent; then its the same thing, except you still cant access private variables, but protected and public data members become protected

   (c) Same with class child : **public** parent; everything remains unchanged. The objects will behave in accordance with the usual laws of objects.

3. **Why can't derived class access private things from base class?**:
   The **private** access modifier, when used in any class, by its definition restricts sub classes and its objects to access variables declared in its scope. As a result, derived classes can not access the private variables defined in the base class.

```
1     class A
2     {
3       private:
4         int a;
5     };
6     class B: public A
7     {
8
9     }obj;
10    int main()
11    {
12      A.a // not accessible
13    }
14
```

4. **Explain use of 'super' keyword with suitable example**
   The *super* keyword in Java refers to the parent class's variables and functions. Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

   - super can be used to refer immediate parent class instance variable.
   - super can be used to invoke immediate parent class method.
   - super() can be used to invoke immediate parent class constructor.

5. **Why to use concept of interface in Java**

   An **Interface** in Java programming language is defined as an abstract type used to specify the behavior of a class. An interface in Java is a blueprint of a class. A Java interface contains static constants and abstract methods.
   The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not the method body. It is used to achieve abstraction and multiple inheritance in Java. In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body. Java Interface also represents the IS-A relationship.

   Like a class, an interface can have methods and variables, but the methods declared in an interface are by default abstract (only method signature, no body).

   - **Provides Communication** - One of the uses of the interface is to provide communication. Through interface you can specify how you want the methods and fields of a particular type.
   - **Multiple Inheritance** - Java doesn't support multiple inheritance, using interfaces you can achieve multiple inheritance.
   - **Abstraction**- Abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does it.
     Since all the methods of the interface are abstract and user doesn't know how a method is written except the method signature/prototype. Using interfaces, you can achieve (complete) abstraction.
   - **Loose Coupling** : Coupling refers to the dependency of one object type on another, if two objects are completely independent of each other and the changes done in one doesn't affect the other both are said to be loosely coupled. You can achieve loose coupling in Java using interfaces

6. **Write Couple of Examples or Applications suitable to Demonstrate use of Inheritances.**

   Inheritance is a core part of Object Oriented Programming. In every Object oriented language, it is heavily used, and these languages are heavily used in almost all fields involving programming, which makes the uses of inheritance vast and versatile. So some of them are listed below.

(a) In Application Development: Every GUI that we see on screens, is often built on simple base classes like buttons, sliders, labels, titlebars etc. The Programmer often just inherits it, and overrides certain methods to his or her liking.

(b) Games are built in many different ways, but as the scale of games increase it becomes pertinent to use simple base classes like trees, enemies, players, friends, coins, As there might be several of the same kind of enemies, and several sorts of trees in the game environment, which will all inherit the base class and override or create certain new methods.

(c) In Data Science, Many algorithms are their own predefined classes, The programmer then tweaks certain parameters, and this helps them streamline and organize the process of applying this algorithm on hundreds of different datasets without hassle.

```
// Here is an example of Creating a GUI in Java using Swing.

// This is the definition of the JButton class, which is used for placing buttons on the screen.
public class JButton extends AbstractButton implements Accessible


import javax.swing.*;
public class ButtonExample {
  public static void main(String[] args) {

    JFrame f=new JFrame("Button Example");
    JButton b=new JButton("Click Here");

    b.setBounds(50,100,95,30);
    f.add(b);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
  }
}


```