

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 9, 2022

Contents

1 Aim and Objectives	1
2 Problem Statements	1
2.1 Problem 1 in C++	1
2.2 Problem 2 in Java	1
2.3 Probelm 3 in Java	1
3 Theory	2
3.1 Concept of Inheritance and its types	2
3.1.1 Advantages	2
3.1.2 Types of Inheritance	2
3.1.3 Base class and Derived Class Constructors	3
4 Platform	4
5 Input	4
6 Output	4
7 Code	5
7.1 C++ Implementation for Problem 1	5
7.1.1 C++ Input and Output	11
7.2 Java Implementation of Problem 2	14
7.2.1 Java Output for Problem 2	16
7.3 Java Implementation of Problem 3 using Interfaces	16
7.3.1 Java Output	17
8 Conclusion	19
9 FAQs	19

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 Mobilenos 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 Problem 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.

```
1  class A
2  {
3      // Base Class
4  };
5
6  class D
7  {
8      // Base Class 2
9  }
10 class B : public A
11 {
12     // Single Inheritance
```

```
13 }
14 class C : public B
15 {
16     // Multi Level Inheritance
17 }
18 class E : public A, public D
19 {
20     // Multiple Inheritance
21 }
22 class F : public E, private D
23 {
24     // hierarchical inheritance
25 }
```

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

```
1
2 // base class
3 class Parent
4 {
5     public:
6
7     // base class constructor
8     Parent()
9     {
10         cout << "Inside base class" << endl;
11     }
12 };
13
14 // sub class
15 class Child : public Parent
16 {
17     public:
18
19     //sub class constructor
20     Child()
```

```
21     {
22         cout << "Inside sub class" << endl;
23     }
24 };
25
26 // main function
27 int main() {
28
29     // creating object of sub class
30     Child obj;
31
32     return 0;
33 }
34
```

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

```
1 // Design and develop inheritance for a given case study, identify objects and
  relationships
2 // and implement inheritance wherever applicable using C++.
3 // Employee class has Emp_name, Emp_id, Address, Mail_id, and Mobile_no as data
4 // members.
5
6 // Inherit the classes:
7 // Programmer
8 // Team Lead
9 // Assistant Project Manager and
10 // Project Manager from employee class.
11
12 // Add Basic Pay as the member of all the inherited classes with 97% of Basic Pay
  as DA, 10
13 // % of Basic Pay as HRA, 12% of Basic Pay as PF, 0.1% of Basic Pay for staff club
  fund.
14 // Generate pay slips for the employees with their gross and net salary.
15
16 #include <iostream>
17 using namespace std;
18
19 class Employee
20 {
21
22 protected:
23     static int ssn;
24     int emp_id = 1000;
25     int age = 0;
26     double basic_sal = 0, da = 0, ta = 0, gross_sal = 0, net_sal = 0;
27     string address_city, position, name;
28
29 public:
30     // Default Constructor
31     Employee()
32     {
33         cout << "The Default Constructor was called" << endl;
34     }
35
36     // Parameterized Constructor
37     Employee(int e, int a, string add, string nam)
38     {
39         cout << "Parameterized constructor was called\n";
40         emp_id = e;
41         age = a;
42         address_city = add;
```

```
43     name = nam;
44 }
45
46 // Copy Constructor
47 Employee(Employee &E)
48 {
49     cout << "Copy Constructor was called" << endl;
50     emp_id = E.emp_id;
51     age = E.age;
52     address_city = E.address_city;
53     name = E.name;
54 }
55
56 void display()
57 {
58     Employee::ssn++;
59     cout << "Employee ssn is:" << ssn << endl;
60     cout << "Employee ID is : " << emp_id << endl;
61     cout << "Employee Name: " << name << endl;
62     cout << "Employee Age: " << age << endl;
63     cout << "Employee Address City: " << address_city << endl;
64 }
65
66 void accept()
67 {
68     cout << "Enter the Employee ID: " << endl;
69     cin >> emp_id;
70     cout << "Enter the Employee Name: " << endl;
71     cin >> name;
72     cout << "Enter the Employee Age: " << endl;
73     cin >> age;
74     cout << "Enter the Employee Address City: " << endl;
75     cin >> address_city;
76 }
77
78 // Destructor
79 ~Employee()
80 {
81     cout << "The Destructor was called" << endl;
82 }
83 };
84
85 int Employee::ssn = 1000;
86
87 class Programmer : public Employee
88 {
89
90 protected:
91     double da = 0, hra = 0, pf = 0, scf = 0;
92
93 public:
94     void calc_gross_sal()
95     {
96         da = 0.97 * basic_sal;
97         hra = basic_sal;
98         pf = basic_sal;
99         scf = basic_sal;
100         gross_sal = da + hra + pf + scf + basic_sal;
101     }
```



```
102
103     void calc_net_sal()
104     {
105         // Reducing Income Taxes
106         net_sal = gross_sal - (0.15) * gross_sal;
107     }
108
109     void accept()
110     {
111         Employee::accept();
112         cout << "Enter the basic Salary of the Programmer : " << endl;
113         cin >> basic_sal;
114         calc_gross_sal();
115         calc_net_sal();
116     }
117
118     void display()
119     {
120         Employee::display();
121         cout << "The Gross Salary is: " << gross_sal << endl;
122         cout << "The Net Salary is: " << net_sal << endl;
123     }
124 };
125
126 class TeamLeader : public Employee
127 {
128
129     protected:
130         double da = 0, hra = 0, pf = 0, scf = 0;
131
132     public:
133         void calc_gross_sal()
134         {
135             da = 0.97 * basic_sal;
136             hra = basic_sal;
137             pf = basic_sal;
138             scf = basic_sal;
139             gross_sal = da + hra + pf + scf + basic_sal;
140         }
141
142         void calc_net_sal()
143         {
144             // Reducing Income Taxes
145             net_sal = gross_sal - (0.15) * gross_sal;
146         }
147
148         void accept()
149         {
150             Employee::accept();
151             cout << "Enter the basic Salary of the Team Leader : " << endl;
152             cin >> basic_sal;
153             calc_gross_sal();
154             calc_net_sal();
155         }
156
157         void display()
158         {
159             Employee::display();
160             cout << "The Gross Salary is: " << gross_sal << endl;
```

```
161         cout << "The Net Salary is: " << net_sal << endl;
162     }
163 };
164
165 class AssistantProjectManager : public Employee
166 {
167
168 protected:
169     double da = 0, hra = 0, pf = 0, scf = 0;
170
171 public:
172     void calc_gross_sal()
173     {
174         da = 0.97 * basic_sal;
175         hra = basic_sal;
176         pf = basic_sal;
177         scf = basic_sal;
178         gross_sal = da + hra + pf + scf + basic_sal;
179     }
180
181     void calc_net_sal()
182     {
183         // Reducing Income Taxes
184         net_sal = gross_sal - (0.15) * gross_sal;
185     }
186
187     void accept()
188     {
189         Employee::accept();
190         cout << "Enter the basic Salary of the Assistant Project Manager : " <<
endl;
191         cin >> basic_sal;
192         calc_gross_sal();
193         calc_net_sal();
194     }
195
196     void display()
197     {
198         Employee::display();
199         cout << "The Gross Salary is: " << gross_sal << endl;
200         cout << "The Net Salary is: " << net_sal << endl;
201     }
202 };
203
204 class ProjectManager : public Employee
205 {
206
207 protected:
208     double da = 0, hra = 0, pf = 0, scf = 0;
209
210 public:
211     void calc_gross_sal()
212     {
213         da = 0.97 * basic_sal;
214         hra = basic_sal;
215         pf = basic_sal;
216         scf = basic_sal;
217         gross_sal = da + hra + pf + scf + basic_sal;
218     }
```

```
219
220 void calc_net_sal()
221 {
222     // Reducing Income Taxes
223     net_sal = gross_sal - (0.15) * gross_sal;
224 }
225
226 void accept()
227 {
228     Employee::accept();
229     cout << "Enter the basic Salary of the Project Manager :" << endl;
230     cin >> basic_sal;
231     calc_gross_sal();
232     calc_net_sal();
233 }
234
235 void display()
236 {
237     Employee::display();
238     cout << "The Gross Salary is: " << gross_sal << endl;
239     cout << "The Net Salary is: " << net_sal << endl;
240 }
241 };
242
243 int main()
244 {
245     cout << "Welcome to Employee Payroll Management System" << endl
246         << endl;
247
248     int choice = 1, number = 1;
249
250     do
251     {
252         cout << "\n\nWhose Details do you wanna enter? " << endl;
253         cout << "1. Programmer\n2. Team Leader\n3. Assistant Project Manager\n4.
Project Manager\n5. Quit\n";
254         cin >> choice;
255
256         if (choice == 1)
257         {
258             cout << "How many Programmers are we talking? ";
259             cin >> number;
260             Programmer pr[number];
261             for (int i = 0; i < number; i++)
262             {
263                 cout << "Enter the Information about the Programmer" << endl;
264                 pr[i].accept();
265             }
266             cout << "\nHere is their Information and their Pay Slips" << endl;
267             cout << endl
268                 << endl;
269
270             cout << "Programmer" << endl;
271
272             for (int i = 0; i < number; i++)
273             {
274                 cout << "Info and Pay Slip of Programmer " << i + 1 << endl;
275                 pr[i].display();
276                 cout << endl;
```

```
277     }
278 }
279 else if (choice == 2)
280 {
281     cout << "How many Team Leaders are we talking? ";
282     cin >> number;
283     TeamLeader tl[number];
284     for (int i = 0; i < number; i++)
285     {
286         cout << "Enter the Information about the Team Leader " << i + 1 <<
endl;
287         tl[i].accept();
288     }
289     cout << "Here is their Information and their Pay Slips" << endl;
290     cout << endl
291         << endl;
292     for (int i = 0; i < number; i++)
293     {
294         cout << "Info and Pay Slip of Team Leader " << i + 1 << endl;
295         tl[i].display();
296         cout << endl;
297     }
298     cout << endl
299         << endl;
300 }
301 else if (choice == 3)
302 {
303     cout << "How many Assistant Project Managers are we talking? ";
304     cin >> number;
305     AssistantProjectManager ap[number];
306     for (int i = 0; i < number; i++)
307     {
308         cout << "Enter the Information about the Assitant Project Manager
" << i + 1 << endl;
309         ap[i].accept();
310     }
311     cout << "Here is their Information and their Pay Slips" << endl;
312     cout << endl
313         << endl;
314     for (int i = 0; i < number; i++)
315     {
316         cout << "Info and Pay Slip of Assitant Project Manager " << i + 1
<< endl;
317         ap[i].display();
318         cout << endl;
319     }
320     cout << endl
321         << endl;
322 }
323
324 else if (choice == 4)
325 {
326     cout << "How many Project Managers are we talking? ";
327     cin >> number;
328     ProjectManager pm[number];
329     for (int i = 0; i < number; i++)
330     {
331         cout << "Enter the Information about the Project Manager " << i +
1 << endl;
```

```
332         pm[i].accept();
333     }
334     cout << "Here is their Information and their Pay Slips" << endl;
335     cout << endl
336         << endl;
337     for (int i = 0; i < number; i++)
338     {
339         cout << "Info and Pay Slip of Project Manager " << i + 1 << endl;
340         pm[i].display();
341         cout << endl;
342     }
343 }
344 } while (choice != 5);
345
346 return 0;
347 }
```

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:
58 Queens
59 Enter the basic Salary of the Team Leader :
60 70000
61 Here is their Information and their Pay Slips
62
63
64 Info and Pay Slip of Team Leader 1
65 Employee ssn is:1002
66 Employee ID is : 1002
67 Employee Name: Steve
68 Employee Age: 80
69 Employee Address City: Queens
70 The Gross Salary is: 347900
71 The Net Salary is: 295715
72
73
74
75 The Destructor was called
76
77
78 Whose Details do you wanna enter?
79 1. Programmer
80 2. Team Leader
81 3. Assistant Project Manager
82 4. Project Manager
83 5. Quit
84 3
85 How many Assistant Project Managers are we talking? 1
86 The Default Constructor was called
87 Enter the Information about the Assitant Project Manager 1
88 Enter the Employee ID:
89 1003
90 Enter the Employee Name:
91 Caulson
92 Enter the Employee Age:
93 60
94 Enter the Employee Address City:
95 Delhi
96 Enter the basic Salary of the Assistant Project Manager :
97 60000
```

OOPJC Assignment 2

```
98 Here is their Information and their Pay Slips
99
100
101 Info and Pay Slip of Assitant Project Manager 1
102 Employee ssn is:1003
103 Employee ID is : 1003
104 Employee Name: Caulson
105 Employee Age: 60
106 Employee Address City: Delhi
107 The Gross Salary is: 298200
108 The Net Salary is: 253470
109
110
111
112 The Destructor was called
113
114
115 Whose Details do you wanna enter?
116 1. Programmer
117 2. Team Leader
118 3. Assistant Project Manager
119 4. Project Manager
120 5. Quit
121 4
122 How many Project Managers are we talking? 1
123 The Default Constructor was called
124 Enter the Information about the Project Manager 1
125 Enter the Employee ID:
126 1005
127 Enter the Employee Name:
128 Fury
129 Enter the Employee Age:
130 56
131 Enter the Employee Address City:
132 Pune
133 Enter the basic Salary of the Project Manager :
134 600000
135 Here is their Information and their Pay Slips
136
137
138 Info and Pay Slip of Project Manager 1
139 Employee ssn is:1004
140 Employee ID is : 1005
141 Employee Name: Fury
142 Employee Age: 56
143 Employee Address City: Pune
144 The Gross Salary is: 2.982e+06
145 The Net Salary is: 2.5347e+06
146
147 The Destructor was called
148
149
150 Whose Details do you wanna enter?
151 1. Programmer
152 2. Team Leader
153 3. Assistant Project Manager
154 4. Project Manager
155 5. Quit
```

156 5

Listing 2: C++ Output

7.2 Java Implementation of Problem 2

```
1 package assignment_2;
2 import java.util.Scanner;
3
4 class Employee
5 {
6     Scanner input = new Scanner(System.in);
7     String name;
8     int emp_id;
9     double salary;
10    int hike = 0;
11
12    final static int full_time_hike_perc = 100;
13    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
14        is: " + salary);
15        calculate_salary();
16        System.out.println("The Salary after the Hike for this Full time Employee
17        is : " + hiked_salary);
18    }
19 }
```

Listing 5: Full Time Employee.java

```
1
2 // Write a Java Program for demonstrating Inheritance in Java.
3 // Write a program in Java showing hierarchical inheritance with base class as
4 // Employee and
5 // derived classes as FullTimeEmployee and InternEmployee with methods
6 // DisplaySalary in
7 // base class and CalculateSalary in derived classes.
8 // Calculate salary method will calculate as per increment given to fulltime and
9 // intern
10 // Employees. Fulltime employee- 50% hike, Intern employee-25% hike. Display
11 // salary
12 // before and after hike.
13
14 package assignment_2;
15
16 import java.util.Scanner;
17
18 class Main {
19     public static void main(String args[]) {
20         System.out.println("Welcome to Salary Hiking Program");
21         Employee emp = new Employee();
22         FullTimeEmployee full_time_emp = new FullTimeEmployee();
23         System.out.println("Enter the Details about the Full Time Employee: \n");
24         full_time_emp.accept();
25         full_time_emp.display_salary();
26         InternEmployee intern_emp = new InternEmployee();
27         System.out.println("Enter the Details about the Intern Employee: \n");
28         intern_emp.accept();
29         intern_emp.display_salary();
30     }
31 }
```

Listing 6: Main.java

7.2.1 Java Output for Problem 2

```
1 Welcome to Salary Hiking Program
2 Enter the Details about the Full Time Employee:
3
4 Enter the Employee Name
5 Tony
6 Enter The Employee ID
7 1001
8 Enter The Employee Salary
9 100000
10 The Employee Name is: Tony
11 The Employee ID is: 1001
12 The Salary Before the Hike for this Full time Employee is: 100000.0
13 The Salary after the Hike for this Full time Employee is : 200000.0
14 Enter the Details about the Intern Employee:
15
16 Enter the Employee Name
17 Steve
18 Enter The Employee ID
19 1002
20 Enter The Employee Salary
21 50000
22 The Employee Name is: Steve
23 The Employee ID is: 1002
24 The Salary Before the Hike for this Intern Employee is: 50000.0
25 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

```
1 package assignment_2b;
2
3 import java.util.Scanner;
4
5 public class TwoWheeler implements Motorcycle, Cycle {
6     public int speed = 0;
7     public int time = 0;
8     public int total_distance = 0;
9
10    @Override
11    public void speed() {
12        System.out.println(total_distance / time);
13    }
14
15    @Override
16    public void totalDistance() {
17        System.out.println(speed * time);
18    }
19 }
```

Listing 8: Two Wheeler.java

```
1 package assignment_2b;
2
3 public interface Motorcycle {
4     public abstract void totalDistance();
5 }
```

5 }

Listing 9: MotorCycle.java

```
1 package assignment_2b;
2
3 public interface Cycle {
4     public abstract void speed();
5 }
```

Listing 10: Cycle.java

```
1 // Write a java program to create two interfaces Motorbike and Cycle.
2 // Motorbike interface consists of the attribute speed.
3 // The method is totalDistance().
4 // Cycle interface consists of the attributes distance and the method speed().
5 // These interfaces are implemented by the class TwoWheeler.
6 // Calculate total distance travelled and Average Speed maintained by Two Wheeler.
7
8 package assignment_2b;
9
10 import java.util.Scanner;
11
12 public class Main {
13
14     public static void main(String args[]) {
15         Scanner input = new Scanner(System.in);
16         TwoWheeler obj = new TwoWheeler();
17
18         System.out.println("Calculating Speed: ");
19         System.out.println("Enter the Distance Travelled by Your 2 Wheeler: ");
20         obj.total_distance = input.nextInt();
21
22         System.out.println("Enter the Time you Travelled on Your 2 Wheeler: ");
23         obj.time = input.nextInt();
24
25         System.out.println("The Speed is: ");
26         obj.speed();
27
28         System.out.println("Calculating Total Distance: ");
29         System.out.println("Enter the Speed of Your 2 Wheeler: ");
30         obj.speed = input.nextInt();
31
32         System.out.println("Enter the Time you Travelled on Your 2 Wheeler: ");
33         obj.time = input.nextInt();
34
35         System.out.println("The Total Distance is: ");
36         obj.totalDistance();
37     }
38 }
```

Listing 11: Main.java

7.3.1 Java Output

```
1 Calculating Speed:
2 Enter the Distance Travelled by Your 2 Wheeler:
3 50
4 Enter the Time you Travelled on Your 2 Wheeler:
5 2
```

```
6 The Speed is:
7 25
8 Calculating Total Distance:
9 Enter the Speed of Your 2 Wheeler:
10 25
11 Enter the Time you Travelled on Your 2 Wheeler:
12 2
13 The Total Distance is:
14 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:

```
1      ObjectName.ClassName::FunctionName();
2
```

An Example:

```
1  // C++ program to resolve inheritance
2  // ambiguity
3
4  #include<iostream>
5  using namespace std;
6
7  // Base class A
8
9  class A {
10     public:
11
12     void func() {
13         cout << " I am in class A" << endl;
14     }
15 };
16
17 // Base class B
18
19 class B {
20     public:
21
22     void func() {
23         cout << " I am in class B" << endl;
24     }
25 };
26
27 // Derived class C
28 class C: public A, public B {
29
30
31 };
```

```
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 that's 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 it's the same thing, except you still can't 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.

```
1 // Here is an example of Creating a GUI in Java using Swing.
2
3 // This is the definition of the JButton class, which is used for placing
4 // buttons on the screen.
5 public class JButton extends AbstractButton implements Accessible
6
7 import javax.swing.*;
8 public class ButtonExample {
9     public static void main(String[] args) {
10
11         JFrame f=new JFrame("Button Example");
12         JButton b=new JButton("Click Here");
13
14         b.setBounds(50,100,95,30);
15         f.add(b);
16         f.setSize(400,400);
17         f.setLayout(null);
18         f.setVisible(true);
19     }
20 }
21
22
```

