A Comparative study of inheritance in C++ and Java

Charanpreetkaur[1,*]

[1]Department of Computer Science and Engineering, Chandigarh University, Gharuan, India

*Corresponding author email: charanpreete6227@cumail.com

ABSTRACT

For reusability and achieving run-time polymorphism, Inheritance is one of the most important features of object-oriented programming. Java and C++ both are object-oriented programming languages and support inheritance. However, their way of inheriting classes is different from each other in some aspects. This paper articulates, the comparison of inheritance in java and C++ on the basis of eleven parameters.

Keywords - C++, Java, derived class, base class.

1. INTRODUCTION

Inheritance is one of the important features of Object-Oriented Programming (OOP). Inheritance allows one object to acquire the properties and behavior of another object, just like, in human beings, child acquires the properties and behavior of his parents and grandparents. C++ and Java are the two object-oriented programming languages. Obviously, both are having the feature of inheritance. However, their way of inheriting is different with some aspects.

1.1 Terminologies used in Inheritance

1.1.1 Class

This is a blueprint/framework which encapsulates the data members and their associate function together. Data members are the variables that declare inside the class. As you can see in Fig 1. id and name are the data members and the minus (-) sign signifies that these members are private, whereas getdata () and show_data () are the functions. The Plus (+) sign signifies that these are public members. Data members are used to define the properties of objects whereas member functions are used to define the behavior of objects. They declared inside the class to access its member variable.

1.1.2 Object

This is the instance of the class. One class can have many instances(objects).

1.1.3 Super Class

The class which is being inherited is called a super-class. Also known as parent class or base class.

1.1.4 Sub Class

The class which acquires the properties of another class is called a sub-class. It is also known as a derived class or child class.

1.1.5 Access Specifiers

Access specifiers are used to define the scope of the member of class. In, General there are three access specifiers, i.e., public, private and protected. However, java is having four access specifiers include default.

1.1.6 Object Class

This is the root class in java, inherited by all the classes. Therefore, there is a single tree of classes in java.

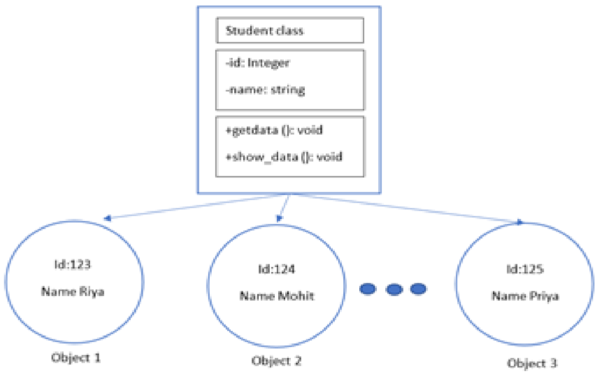


Figure 1. Class and Objects

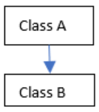
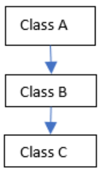
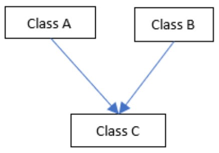
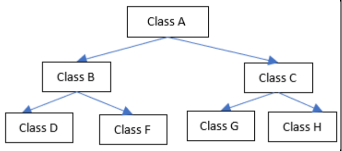
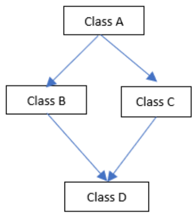
1.2 Objective of Inheritance

- To achieve Runtime polymorphism
- To Reuse the code

2. TYPES OF INHERITANCE

There are five types of inheritance supported by C++. These types have mentioned the Table 1. Java does not support multiple and hybrid inheritance. So, we can say that are only three types of inheritance in java. However, in java interfaces are used to achieve multiple and hybrid inheritance.

Table 1. Types of Inheritance.

Types of Inheritance	Description	Diagram
Single Level Inheritance	An object acquires the properties of another single object, as you can see in Fig 2.	 Figure 2. Single Level Inheritance.
Multilevel Inheritance	As shown in Fig 3., in multilevel Inheritance, an object acquires the properties of another object which itself acquires the properties of another object.	 Figure 3. Multilevel Inheritance
Multiple Inheritance	In multiple inheritance, an object acquires the properties of more than one object, as mentioned in Fig 4.	 Figure 2. Multiple Inheritance
Hierarchical Inheritance	As shown in Fig 5., Hierarchical inheritance is achieved when an object is being inherited by a number of objects, and further, that inherited object is being inherited by another number of objects, and so on.	 Class A Class B Class C Class D Class F Class G Class H Figure. 3. Hierarchical Inheritance.
Hybrid Inheritance	A combination of more than one inheritance is known as Hybrid Inheritance. In Fig 6. Hierarchical and Hybrid inheritance is combined to achieve Hybrid Inheritance.	 Figure 6. Hybrid Inheritance.

3. ACCESSIBILITY OF BASE CLASS MEMBERS IN CHILD CLASS

Members of base class can be access with the object of base class as well as with the object of child class. However, java and C++ provide the restrictions in this concept with the help of access specifiers (public, private, and protected) mentioned in Table 2(a)and Table 2(b).

Note: child class members cannot be accessed via the base class object.

Table 2(a). Accessibility of base class members with object

Base Class Members' Access Specifier	**Accessible with base class object**	**Accessibility with child class object**
Private	Not Accessible	Not Accessible
Public	Accessible	Accessible
Protected	Not Accessible	Not Accessible

Table 3(b). Accessibility of base class members inside other classes

Base Class Members' Access Specifier	**Accessibility inside the base class**	**Accessibility inside the derived class**	**Accessibility inside another class**
Private	Accessible	Not Accessible	Not Accessible
Public	Accessible	Accessible	Accessible
Protected	Accessible	Accessible	Not Accessible

4. MODES OF INHERITANCE IN C++

Public, private, and protected are the three modes of inheritance in C++. By default, the mode of inheritance is private. When the mode of inheritance is private or protected, the access specifiers of all the public and protected members changes in the derived class.

However, sometimes, the access specifier of a public or protected member of the base class may need to restore in the derived class. This can be done by declaring such members explicitly with its original access specifier.

Table 4. Effect of Inheritance on the accessibility of members

Mode of Inheritance	**Private members**	**Public members**	**Protected member**
Child class publicly inherits the base class	Private	Public	Protected
Child class privately Inherits the base class	Private	Private	Private
Child class protectively inherits base class	Private	Protected	Protected

From *table 3*, this is concluded that the private members cannot be inherited as their accessibility does not affect with the mode of inheritance.

5. AMBIGUITY IN INHERITANCE

Ambiguity is the problem of duplicity due to which compiler is unable to call the appropriate member of the class. In C++, Ambiguity can be resolved with the help of a virtual base class and scope resolution operator. However, in java, due to ambiguity problems, multiple inheritance is not included. However, interfaces are used to achieve multiple inheritance.

5.1 Ambiguity in Multiple Inheritance

Let us suppose that there is the same function available in two different parent classes of a single derived class. As per the rule of inheritance, the two same functions of the base classes are being inherited by the child class. In this case, the compiler cannot find out which function has to call.

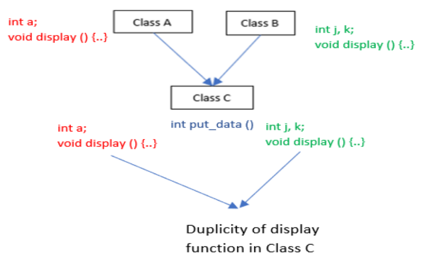


Figure 4. Ambiguity in Multiple Inheritance

5.2 Ambiguity Hybrid Inheritance

In hybrid inheritance, ambiguity arises when a base class inherited twice by a child class. For example, in Fig 8. Class A is inherited by Class B and Class C. Both Class B and Class C has the properties of Class A, which are further inherited by class D. members of Class A are indirectly inherited two times by Class D through Class B and Class C.

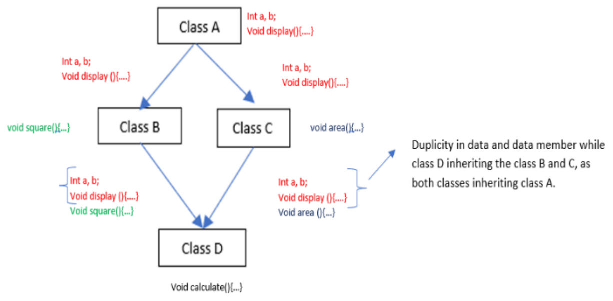


Figure 5. Ambiguity in Hybrid Inheritance

6. INHERITANCE OF OBJECT CLASS

In java every class is a child class of object class which is declared under "java. Lang" package. When we create a single class in java it directly inherits the object class. Whereas, if we "extends" one class from another then that class indirectly inherits the object class. So, we can say that their a root class in java, i.e., Object class all the other classes are a child of this class. Therefore, there is a single tree of classes in java. Also, the Object class has many inbuilt methods that we can use to fulfill the requirement.

However, if we talk about C++, there is no such object class. Every class is individual. So, when we use the concept of inheritance in C++, there is a forest of classes.

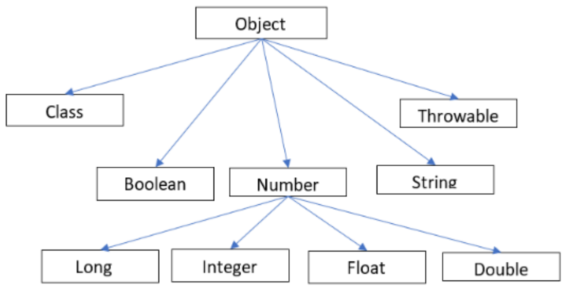


Figure 6. Object Class in java.

7. ORDER OF CONSTRUCTOR AND DESTRUCTOR CALL DURING INHERITANCE

Constructor is used to initialize the data members object. When a class inherits another class, the data members of base class will become the data members of the child class. Child class data members are obviously initialized by the base class constructor, however, to initialize the data member of the base class, the base class constructor should call first.

A destructor is a special function that is used to destroy an object. When the object scope ends, the destructor will automatically be called. If a class inherits another class the destructor of the child class will call first and then the base class constructor will call. The order is the reverse of the order of the constructor call during inheritance.

Table 5. Comparison of Java and C++ based on Inheritance.

Parameter	C++	Java	Explanation
Keyword/operator Used	colon (:)	extends keyword	Java use extends/implements keyword for inheritance whereas C++ uses colon operator to inherit the classes
Single Level Inheritance supported	Yes	Yes	Both java and C++ support Single Level Inheritance
Multilevel Inheritance supported	Yes	Yes	Both java and C++ support multilevel Inheritance
Multiple Inheritance supported	Yes	No	C++ supports multiple inheritance because ambiguity can be resolved by the scope resolution operator. However, Java does not, because there is no scope resolution operator in java to resolve the ambiguous issue. However, it can be achieved by using the interface.
Hierarchical Inheritance supported	Yes	Yes	Both java and C++ support Hierarchical Inheritance.
Hybrid Inheritance supported	Yes	may or may not	If hybrid inheritance is the combination of multiple inheritance with any other inheritance, then, java will not support otherwise, it can be achieved.
Protected Access Specifier	Yes	Yes	The meaning of protected access specifier in java is different from C++. in Java, protected members of a class “A” are accessible in other class “B” of the same package, even if B doesn’t inherit from A (they both have to be in the same package). Whereas, in C++ protected members of the class are only visible to the child class.
Accessibility of ancestor’s member	Directly	Indirectly	In C++, Grandparent members can be directly accessed by the scope resolution operator. However, in java grandparent members can be indirectly accessed through the parent class and super keyword.
Default constructor	Automatically	Automatically	Like C++, in java parent call default constructor is automatically called by the child class.
Parameterized constructor	Initializer list	Super keyword	In java, the super keyword is used to represent the parent class.

8. CONCLUSION

From the review, it is concluded that there is lots of difference in inheritance in java and C++. In java, multiple inheritance cannot be performed. However, it can perform with the help of interfaces. Interfaces are used to achieve 100% abstraction. Also, there is no root class, i.e., object class in C++, due to which C++ has a forest of inherited classes. Whereas in java every single class is a child class of the root class, i.e., object class. There is one more important point to note, java has a super keyword to represent the base class, whereas in java there is no such keyword. The super keyword is used to call the parent class constructor and is automatically included by the compiler as a first line in the child class's constructor. So, it is concluded that java inheritance is well defined and organized as compared to C++.

REFERENCE

[1] M. R. S. RAUT, "Research Paper on Object-Oriented Programming (OOP)," *International Research Journal of Engineering and Technology (IRJET),* 2020 pp. 1452-1456 .

[2] T. A. Ines Ayadi and Noe"mie Simoni, "SLA approach for "cloud as a Service", *IEEE Sixth International Conference on Cloud Computing,* Santa Clara, CA, USA, 2013.

[3] R. Sharma, "A Review on Cloud Computing- An Emerging Technology," *International Journal of Scientific & Engineering Research*, 2013, pp. 2120-2124.

[4] K. Rohit, *Data Structures and Object Oriented Programming with C++ (For Anna University)*, Vikas Publishing House Pvt Limited, 2010.

[5] M. M. J. Mrs. Kanchanmala D Talekar, "Inheritance in java," in *International Research Journal of Engineering and Technology, IRJET*, 2018.

[6] A. M. Fawzi Albalooshi, "A Comparative Study on the Effect of Multiple Inheritance Mechanism in Java, C++, and Python on Complexity and Reusability of Code," *International Journal of Advanced Computer Science and Applications,* 2017, vol. 8, no. 6, pp. 109-116.

[7] A. G. Sujeet Kumar, *A Brief Study On Inheritance*, 2014, 10.13140/2.1.1659.8729.

[8] W. L. H. William R. Cook, "Inheritance Is Not Subtyping.," *Theoretical aspects of object-oriented programming Journal*, 1990.

[9] Shivam, "A Study on Inheritance Using Object Oriented Programming with C++," I*nternational Journal of Advance Research in Computer Science and Management Studies*, 2013, vol. 1(2), pp 10-21.

[10] E. Balagurusamy, *Object Oriented programming with C++* 6th edition, Tata McGraw-Hill 2013

[11] Kaur L, Kaur. N., Ummat, a., Kaur, J., & Kaur, N. Research paper on object oriented software engineering., *International Journal Of Computer Science And Technology*, 2016, 36-38.

[12] Kak, Avinash C. *Programming with Objects, A Comparative, Presentation of Object-Oriented Programming with C++ and Java*, John Wiley, 2003. ISBN 0-471-26852-6.

[13] Svenk, Goran, *Object-Oriented Programming: Using C++ for Engineering and Technology* Delmar, 2003. ISBN 0-7668-3894-3.

[14] Seed, Graham M., *An Introduction to Object-Oriented Programming in C++ with Applications in Computer Graphics*, Second Ed., Springer-Verlag, 2001.ISBN 1-85233-450-9.

[15] H. Schildt, "*Java The Complete Reference*" 11th edition, The future, 2020.

[16] E balaguruswamy, *Programming with java A primer 3e*, Tata Mcgraw Hill Education Private Limited, 2013