Inheritance in Object-Oriented Programming

What is Inheritance?

Inheritance is a fundamental concept in Object-Oriented Programming (OOP) that allows a class (Derived class) to inherit properties and behaviors from another class (Base class) This mechanism is supported by OOP in which a 'Class' is used as a template to create an 'Object'. For example, 'Student' is a class in which 'Lara' (with its own unique identity) is an object. These objects can communicate with each other, providing a modular and organized approach to software development. Inheritance allows code extension and encourages code reuse. It is especially valuable for customizing a new class based on an existing one.

Real-time Inheritance

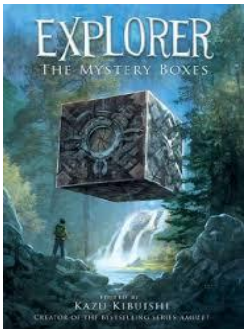
Inheritance establishes an 'is a' relationship. For example, Car and Truck are derived classes, each of which inherits from a base class 'Vehicle'. This can be expressed as follows: 'Car **is a** vehicle' and 'A truck **is a** vehicle'.

The concept of inheritance can be further explained by using The Book Analogy.

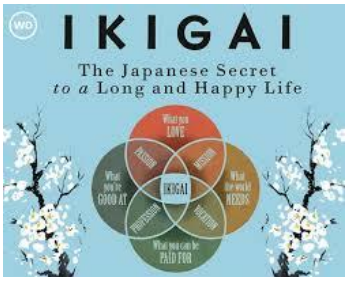
Imagine a library of books, where each book represents a class. The base class is a general 'Book' class and derived classes are more specific genres like 'Mystery', 'Non-Fiction', 'Educational', etc.

Each genre inherits the existing properties of a 'Book' but adds unique characteristics specific to its genre.

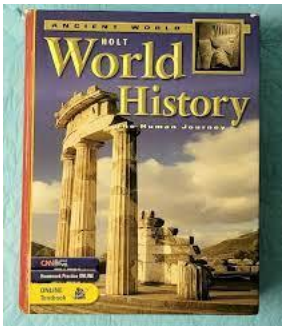
In this example: Base class/ Parent class - Book, Parent class properties: Name, Author. Derived class/Child classes - Mystery, Non-Fiction, Educational.



Class: Mystery
Name: Explorer
Author: Kazu Kibuishi
Region: Atlantic



Class: Non-Fiction
Name: Ikigai
Author: Hector Garcia, Francesc Miralles
Origin: Japan



Class: Educational
Name: World History
Author: CNN Group
Period: 1970s

All subclasses inherit the attributes 'Name' and 'Author' from the Book superclass. These features need not be repeated when new classes are created. Thus, inheritance curbs code complexity and fosters a more intuitive understanding of code.

'Region', 'Origin' and 'Period' are class-specific attributes for classes Mystery, Non-Fiction and Educational respectively.

When it is necessary to apply new properties to all classes, they can be easily added to the Base Class. In the book analogy, if a new feature such as 'Number of pages' is needed, it can be added to the 'Book' class, and all its subclasses will inherit it. This way, inheritance makes it easier to extend and maintain code, and it also improves its robustness and adaptability.

Types of Inheritance

Depending on the features that need to be inherited and the base class from which they are derived, Inheritance can be classified into five different types:

I. Single Inheritance

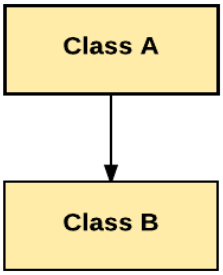


Figure 1

This is the simplest type of inheritance, wherein there is one base class and one derived class. The child class inherits attributes and behaviors from a single parent class. This is depicted in *Figure 1*.

Here, Class A is the base class and Class B is the derived class.

Class B inherits Class A.

II. Multiple Inheritance

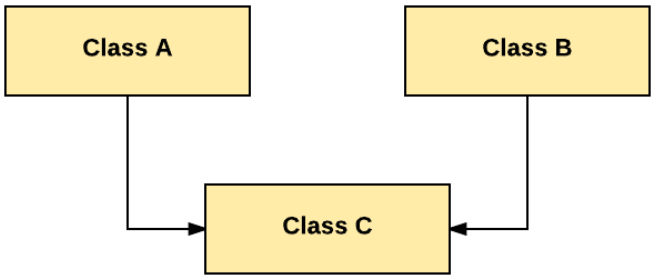


Figure 2

Multiple Inheritance involves a derived class inheriting from two or more base classes as seen in *Figure 2*.

This allows the derived class to acquire attributes and behaviors from multiple parent classes.

Here, classes A and B are base classes. Class C is a derived class. **Class C inherits both Class A and Class B.**

III. Multi-level Inheritance

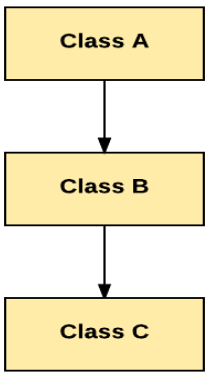


Figure 3

This type involves a chain of inheritance, where child classes become parent classes at subsequent levels.

As shown in Figure 3, Class A is the base class from which Class B inherits. At the next level, Class B, which was a derived class before, now becomes the base class from which Class C inherits.

In this way, child classes become parent classes at the succeeding levels and the state of each class keeps changing.

Class A is inherited by Class B which in turn is inherited by Class C.

IV. Hierarchical Inheritance

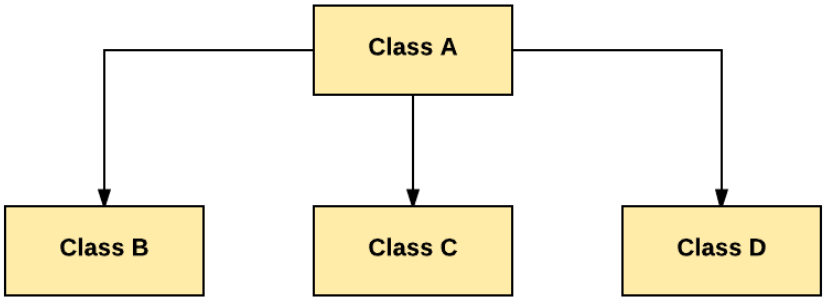


Figure 4

Hierarchical Inheritance, as shown in *Figure 4* consists of one base class and several derived classes. Each derived class shares attributes and behaviors from the common parent class**.**

In the illustrated diagram, Class A is the base class. Classes B, C and D are all derived classes that inherit from the common base or parent class A.

Class A is inherited by Classes B, C and D.

V. Hybrid Inheritance

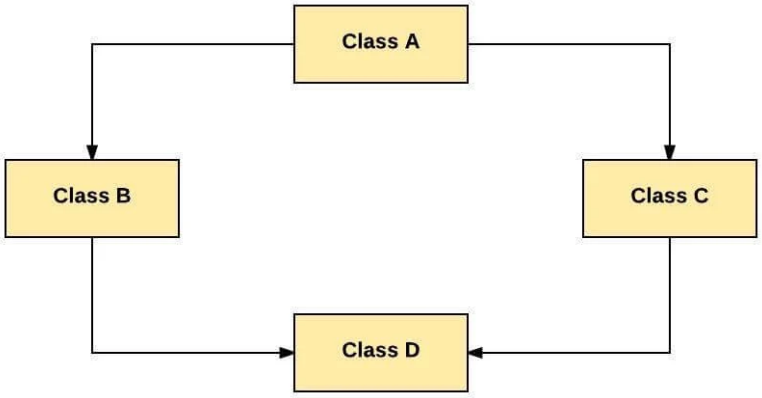


Figure 5

This is a combination of two or more types of inheritance. It leverages the advantages of different inheritance types.

In *Figure 5,*

i. Consider Classes A, B and C. They follow **Hierarchical Inheritance** wherein Class A is the base class and Classes B and C are derived classes. *Class A is inherited by Classes B and C.*

ii. Consider Classes B, C, and D. Class D is the derived class and Classes B and C are base classes exhibiting **Multiple Inheritance**. *Class D inherits Classes B and C.*

iii. Classes B and C which were child classes in (i) have become parent classes for Class D in (ii) which is an example of **Multi-level Inheritance**. *Class A is inherited by Class B and Class C, while Class D inherits Class B and Class C.*

Benefits of Inheritance

1. **Code Reuse and Extensibility:** Inheritance promotes the reuse of code, reducing redundancy and enabling the extension of existing functionality.

2. **Code Organization and Maintenance:** Class hierarchies organize code in a structured manner, making it easier to manage and maintain.

3. **Increased Readability and Modularity:** Inheritance enhances code readability by establishing class relationships and promotes modularity, simplifying updates and modifications.

Note: While inheritance offers numerous benefits, judicious usage is essential to prevent overcomplications and to maintain a clear and understandable codebase.

References

[1] Graphic/Image.Inheritance types. https://logicmojo.com/inheritance-in-oops(Accessed January 21, 2024)

[2]Graphic.Image.Explorer.https://www.google.com/imgres?imgurl=https%3A%2F%2Fm.media-amazon.com%2Fimages%2FI%2F71do84N1W2L._AC_UF1000%2C1000_QL80_.jpg&tbnid=BShtzSUoOyx34M&vet=12ahUKEwj5j6G94e-DAxVhAWIAHaoeD_0QMygAegQIARBE..i&imgrefurl=https%3A%2F%2Fwww.amazon.ca%2FExplorer-Mystery-Boxes-Kazu-Kibuishi%2Fdp%2F141970009X&docid=FAwltBl3Kc4AAM&w=667&h=1000&q=explorer%20kazu%20kibuishi&ved=2ahUKEwj5j6G94e-DAxVhAWIAHaoeD_0QMygAegQIARBE (Accessed January 21, 2024)

[3] Graphic.Image.Ikigai.https://55nda.com/blogs/anil-khosla/2022/10/26/ikigai-the-mysterious-word/ (Accessed January 21, 2024)

[4] Inheritance. https://www.geeksforgeeks.org/classes-objects-java/ (Accessed January 21, 2024)