**UNIVERSIDAD METROPOLITANA**

**DE HONDURAS**

**VICERRECTORIA DE PRE-GRADO**

**INGENIERIA EN INFORMATICA**



ASIGNATURA

Programación orientada a objetos

NOMBRE DEL DOCENTE

Ing. Irvin Vásquez

TRABAJO

Investigación Herencia

PRESENTADO POR:

Manuel Alberto Gutiérrez Molina 201901210

Tocoa Colon 08 de Febrero del 2022

**Contenido**

[I. HERENCIA 1](#_Toc95175181)

[1.1 ¿Qué es herencia en programación? 1](#_Toc95175182)

[1.2 Términos importantes 3](#_Toc95175183)

[1.3 Tipos de herencia en Java 3](#_Toc95175184)

[1.4 Ventajas de la Herencia 4](#_Toc95175185)

[1.5 Ejemplos de herencia en programación 5](#_Toc95175186)

[1.5.1 Primer ejemplo 5](#_Toc95175187)

[1.5.2 Segundo ejemplo 6](#_Toc95175188)

[1.5.3 Ejemplo Herencia única 7](#_Toc95175189)

[1.5.4 Ejemplo Herencia multinivel 8](#_Toc95175190)

[1.5.5 Ejemplo Herencia jerárquica 9](#_Toc95175191)

[1.5.6 Ejemplo Herencia múltiple 10](#_Toc95175192)

[1.5.7 Ejemplo Herencia Hibrida 11](#_Toc95175193)

[1.6 Jerarquía de clases 12](#_Toc95175194)

[1.7 ¿Qué se puede hacer en una Subclase? 13](#_Toc95175195)

[1.8 Conclusiones sobre la herencia en Java 14](#_Toc95175196)

[II. ANEXOS 15](#_Toc95175197)

[*2.1 Figura 1* 15](#_Toc95175198)

[*2.2 Figura 2* 15](#_Toc95175199)

[*2.3 Figura 3* 16](#_Toc95175200)

[*2.4 Figura 4* 16](#_Toc95175201)

[*2.5 Figura 5* 17](#_Toc95175202)

# HERENCIA

## 1.1 ¿Qué es herencia en programación?

Cuando hablamos de herencia en programación no nos referimos precisamente a que algún familiar lejano nos ha dejado una fortuna. En realidad se trata de uno de los pilares fundamentales de la programación orientada a objetos. Es el mecanismo por el cual una clase permite heredar las características (atributos y métodos) de otra clase. La herencia permite que se puedan definir nuevas clases basadas de unas ya existentes a fin de reutilizar el código, generando así una jerarquía de clases dentro de una aplicación. Si una clase deriva de otra, esta hereda sus atributos y métodos y puede añadir nuevos atributos, métodos o redefinir los heredados.

El concepto de clases divididas en subclases se utiliza en la vida diaria y conduce al de herencia; la clase animal se divide en mamíferos, anfibios, insectos, aves, etcétera; la clase vehículo se divide en autos, camiones, autobuses, motocicletas, etcétera. La clase electrodoméstico se divide en lavadora, frigorífico, tostadora, microondas, y así sucesivamente. La idea principal de estas divisiones reside en el hecho de que cada subclase comparte características con la clase de la cual se deriva. Los autos, camiones, autobuses, y motocicletas, tienen motor, ruedas y frenos; pero, además de estas características compartidas, cada subclase tiene las propias; los autos, por ejemplo, pueden tener maletero, cinco asientos; los camiones, cabina y caja para transportar carga, entre otras características.

La clase principal de la que derivan las restantes se denomina base, padre o superclase; las subclases también se denominan derivadas o hijas. Las clases reflejan que el mundo real contiene objetos con propiedades o atributos y comportamiento; la herencia manifiesta que dichos objetos tienden a organizarse en jerarquías; esta jerarquía, desde el punto de vista del modelado, se denomina relación de generalización o es-un (del inglés is-a). En programación orientada a objetos, la relación de generalización se denomina herencia; cada clase derivada hereda las características de la cual es base y además añade sus propias características, atributos y operaciones. Las clases bases también pueden ser subclases o derivarse de otras superclases. Así, el programador puede definir una clase Animal que encapsule todas las propiedades o atributos (altura, peso, número de patas, etc.) y el comportamiento u operaciones (comer, dormir, andar) que pertenecen a cada uno. Los animales específicos como mono, jirafa, canguro o pingüino tienen a su vez características propias. Como las técnicas de herencia se representan con la citada relación es-un, se puede decir que mono es-un Animal con características propias: subir a los árboles, saltar entre ellos, entre otras; además, comparte con jirafa, canguro y pingüino las características propias de cualquier animal, como comer, beber, correr, dormir, etcétera. Otro ejemplo basado en la industria del automóvil es el siguiente: la clase caja de cambios hace unos años tenía cuatro marchas adelante y una atrás; posteriormente se incorporó una delantera y en los últimos años ya se comercializan automóviles con seis; también se pueden considerar las cajas automáticas como otra extensión de la clase base.

En Java tenemos que tener claro cómo llamar a la clase principal de la que heredamos y aquella que hereda de ella, así, clase principal se le denomina superclase. La clase que hereda se llama subclase. Por lo tanto, una subclase es una versión especializada de una superclase. Hereda todas las variables y métodos definidos por la superclase y agrega sus propios elementos únicos.

## 1.2 Términos importantes

* Superclase: la clase cuyas características se heredan se conoce como superclase (o una clase base o una clase principal).
* Subclase: la clase que hereda la otra clase se conoce como subclase (o una clase derivada, clase extendida o clase hija). La subclase puede agregar sus propios campos y métodos, además de los campos y métodos de la superclase.
* Reutilización: la herencia respalda el concepto de “reutilización”, es decir, cuando queremos crear una clase nueva y ya hay una clase que incluye parte del código que queremos, podemos derivar nuestra nueva clase de la clase existente. Al hacer esto, estamos reutilizando los campos/atributos y métodos de la clase existente.

## 1.3 Tipos de herencia en Java

Java, como la mayoría de lenguajes de programación modernos, dispone de diferentes tipos de herencia que podemos usar para hacer todavía más eficiente nuestro programa al añadir características o atributos procedentes de diferentes clases. Lo que sí debemos tener en cuenta es que solo puede existir una superclase, como veremos en los siguientes tipos de herencia:

* Herencia única: donde las subclases heredan las características de solo una superclase.
* Herencia Multinivel: una clase derivada heredará una clase base y, además, la clase derivada también actuará como la clase base de otra clase.
* Herencia Jerárquica: una clase sirve como una superclase (clase base) para más de una subclase.
* Herencia Múltiple (a través de interfaces): una clase puede tener más de una superclase y heredar características de todas las clases principales. Pero Java no admite herencia múltiple con clases, así que para lograrlo tenemos que usar Interfaces. *public interface Actor {int activo = 1;int inactivo = 0;}*
* Herencia Híbrida (a través de Interfaces): Es una mezcla de dos o más tipos de herencia anteriores. Como Java no admite herencia múltiple con clases, la herencia híbrida tampoco es posible con clases, pero como en el ejemplo anterior, podemos lograr el mismo resultado a través de Interfaces.

## 1.4 Ventajas de la Herencia

Entre las principales ventajas que ofrece la herencia en el desarrollo de aplicaciones, están:

* Reutilización del código: En aquellos casos donde se necesita crear una clase que, además de otros propios, deba incluir los métodos definidos en otra, la herencia evita tener que reescribir todos esos métodos en la nueva clase.
* Mantenimiento de aplicaciones existentes: Utilizando la herencia, si tenemos una clase con una determinada funcionalidad y tenemos la necesidad de ampliar dicha funcionalidad, no necesitamos modificar la clase existente (la cual se puede seguir utilizando para el tipo de programa para la que fue diseñada) sino que podemos crear una clase que herede a la primera, adquiriendo toda su funcionalidad y añadiendo la suya propia.

## 1.5 Ejemplos de herencia en programación

### 1.5.1 Primer ejemplo

class Vehiculo

{

public decimal VelocidadMaxima { get; set; }

public int NumeroRuedas { get; set; }

public string Marca { get; set; }

public string Modelo { get; set; }

}

class Moto : Vehiculo

{

public int Cilindrada { get; set; }

}

class Coche : Vehiculo

{

public string Traccion { get; set; }

}

|  |
| --- |
| **Vehiculo** |
| -VelocidadMaxima  -NumeroRuedas  -marca  modelo |

|  |
| --- |
| **Coche** |
| -Traccion |

|  |
| --- |
| **Moto** |
| -Cilindrada |

### 1.5.2 Segundo ejemplo

//Clase Persona. La clase Persona es la Clase Base

public class Persona {

private String nif;

private String nombre;

public String getNif() {

return nif; }

public void setNif(String nif) {

this.nif = nif;}

public String getNombre() {

return nombre; }

public void setNombre(String nombre) {

this.nombre = nombre; }

}

public class Alumno extends Persona{

private String curso;

public String getCurso() {

return curso; }

public void setCurso(String curso) {

this.curso = curso}

|  |
| --- |
| **Persona** |
| -nif  -nombre |

}

|  |
| --- |
| **Alumno** |
| -curso |

### 1.5.3 Ejemplo Herencia única

Nombre del archivo SingleA.java

*public class SingleA {*

*public String baseclass;*

*public void singlebaseclass() {*

*System.out.println("Base class method");*

*}*

*}*

Nombre del archivo SingleB.java

*public class SingleB extends SingleA {*

*public void singlechildclass() {*

*System.out.println("Child class method");*

*}*

*public static void main(String[] args) {*

*SingleB b = new SingleB();*

*b.baseclass = "Base class name";*

*System.out.println(b.baseclass);*

*b.singlebaseclass(); //calls parent method*

*b.singlechildclass(); //calls child method*

*}*

*}*

### 1.5.4 Ejemplo Herencia multinivel

*public class MultiA {*

*public void methodA() {*

*System.out.println("Method A");*

*}*

*}*

*public class MultiB extends MultiA {*

*public void methodB() {*

*System.out.println("Method B");*

*}*

*}*

*public class MultiC extends MultiB{*

*public void methodC() {*

*System.out.println("Method C");*

*}*

*public static void main(String[] args) {*

*MultiC c = new MultiC();*

*c.methodA();*

*c.methodB();*

*c.methodC();*

*}*

*}*

### 1.5.5 Ejemplo Herencia jerárquica

*public class HierarchyA {*

*public void methodA() {*

*System.out.println("Method A");*

*}*

*}*

*public class HierarchyB extends HierarchyA {*

*public void methodB() {*

*System.out.println("Method B");*

*}*

*public static void main(String[] args) {*

*HierarchyB b = new HierarchyB();*

*b.methodA();*

*b.methodB();*

*}*

*}*

*public class HierarchyC extends HierarchyA{*

*public void methodC() {*

*System.out.println("Method C");*

*}*

*public static void main(String[] args) {*

*HierarchyC c = new HierarchyC();*

*c.methodA();*

*c.methodC();*

*}*

*}*

### 1.5.6 Ejemplo Herencia múltiple

La herencia múltiple se admite en C ++. En Java, podemos implementar este concepto utilizando interfaces que veremos hacia el final de este tutorial.

*class classA {*

*public void show() {*

*System.out.println("Class A");*

*}*

*}*

*class classB {*

*public void show() {*

*System.out.println("Class B");*

*}*

*}*

*public class MultipleInheritance extends classA,classB { //Suppose if allowed*

*public static void main(String[] args) {*

*MultipleInheritance obj = new MultipleInheritance();*

*obj.show();*

*}*

*}*

### 1.5.7 Ejemplo Herencia Hibrida

*class classA {*

*public void methodA() {*

*System.out.println("Method A");*

*}*

*}*

*class classB extends classA {*

*public void methodB() {*

*System.out.println("Method B");*

*}*

*}*

*class classC extends classA {*

*public void methodC() {*

*System.out.println("Method C");*

*}*

*}*

*class classD extends classB {*

*public void methodD() {*

*System.out.println("Method D");*

*}*

*}*

*public class HybridExample {*

*public static void main(String[] args) {*

*classB b = new classB();*

*classC c = new classC();*

*classD d = new classD();*

*System.out.println("Calling from class B");*

*b.methodA();*

*b.methodB();*

*System.out.println("Calling from class C");*

*c.methodA();*

*c.methodC();*

*System.out.println("Calling from class D");*

*d.methodA();*

*d.methodB();*

*d.methodD();*

*}*

*}*

## 1.6 Jerarquía de clases

La herencia permite la transmisión de propiedades y métodos de un objeto a otro a través de la organización jerárquica de clases a la que pertenece. Al clasificarse, se obtiene una jerarquía de clases y surge la generalización y especialización. La herencia establece las relaciones entre clases (una o varias).La herencia permite crear otro objeto a partir de uno ya existente. Al tipo de objeto original se lo conoce como base o ascendente, mientras que el nuevo que hereda las características se lo llama derivada o descendente. La herencia es transitiva, esto significa que las características de un objeto se pueden transferir a otro, a través de una clase intermedia, es decir, si C es descendente de B y B es descendente de A entonces A le transmite características a C.

Una clase derivada hereda todos los métodos de la clase base, exceptuando los siguientes:

* Constructores, destructores y constructores de copia de la clase base.
* Operadores sobrecargados de la clase base.
* Funciones amigas de la clase base.

## 1.7 ¿Qué se puede hacer en una Subclase?

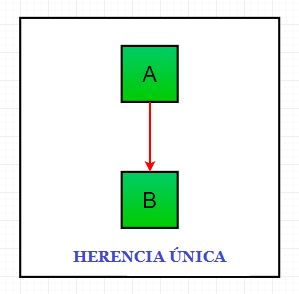
* En las subclases podemos heredar los miembros tal como están, reemplazarlos, ocultarlos o complementarlos con nuevos miembros:
* Los campos heredados se pueden usar directamente, al igual que cualquier otro campo.
* Podemos declarar nuevos campos en la subclase que no están en la superclase.
* Los métodos heredados se pueden usar directamente tal como son.
* Podemos escribir un nuevo método de instancia en la subclase que tenga la misma firma que el de la superclase, anulándolo (como en el ejemplo anterior, el método toString () se reemplaza).
* Se puede escribir un nuevo método estático en la subclase que tiene la misma firma que el de la superclase, escondiéndolo así.
* Podemos declarar nuevos métodos en la subclase que no están en la superclase.
* Podemos escribir un constructor de subclase que invoca el constructor de la superclase, ya sea implícitamente o mediante la palabra clave súper.

## 1.8 Conclusiones sobre la herencia en Java

* Una subclase hereda  los miembros públicos y protegidos de la Superclase si  está en el mismo paquete.
* Los miembros privados solamente se heredan si están en mismo paquete.
* Pueden utilizarse mismo nombre para designar miembros tanto en la superclase como la subclase, esto responde al concepto de ocultación de miembros de la superclase.
* Se pueden crear tantos atributos o métodos en la subclase como consideremos oportuno y a su vez poder utilizar todos los miembros de la Superclase.

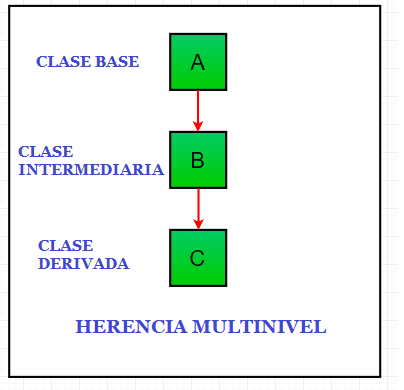
# ANEXOS

## *2.1 Figura 1*

*Herencia única*

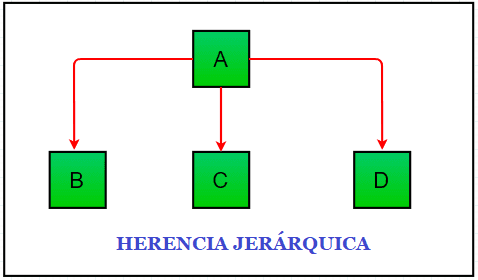
*Nota: obtenida de la web.*

## *2.2 Figura 2*

*Herencia Multinivel*

*Nota: obtenida de la web.*

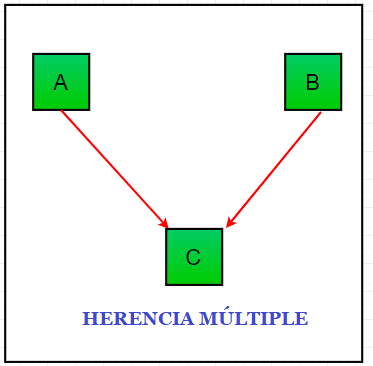
## *2.3 Figura 3*

*Herencia Jerárquica*

*Nota: obtenida de la web.*

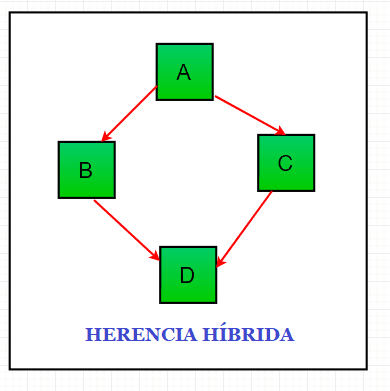
## *2.4 Figura 4*

*Herencia Múltiple*



*Nota: obtenida de la web.*

## *2.5 Figura 5*

*Herencia Hibrida*

*Nota: obtenida de la web.*