1. Creamos la clase abstracta **ReinoAnimal**. Esta clase hereda de **Object** implícitamente, es decir, no tiene **extends** pero por defecto extiende de la clase Object. Dentro de la clase declaramos el método abstracto **habla()**.

|  |
| --- |
|  |

1. Creamos la subclase **Aves** que herede de **ReinoAnimal**. **Aves** es una subclase abstracta que hereda de la superclase abstracta **ReinoAnimal**. En **Aves** declaramos el método abstracto **vuela()**.

|  |
| --- |
|  |

En este caso, no estamos obligados a redefinir los métodos de la superclase **ReinoAnimal** ya que la clase **Aves** es abstracta.

1. Creamos la clase **Gallina**, que hereda de **Aves**.

En este caso, estamos obligados a redefinir los métodos de la superclase **Aves**, el cual es solo uno: **vuela()**.

La clase Gallina tiene dos niveles de superclases, la primera esa **Aves**, y esta hereda de **ReinoAnimal**, entonces también debemos redefinir el método **habla()**:

|  |
| --- |
|  |

1. Ahora utilizamos la clase Gallina en una clase main:

|  |
| --- |
|  |

**Polimorfismo**

* Podemos crear instancias (objetos) de una subclase asignándola a su superclase directa. En este ejemplo creamos una instancia de **Gallina** y la asignamos a su superclase **Aves**.

|  |
| --- |
|  |

* Podemos acceder al método **vuela()** de la superclase **Aves**.
* Como la superclase de Gallina es **Aves** que extiende de **ReinoAnimal** podemos acceder al método **habla()**, creado en la clase **ReinoAnimal**.
* Podemos crear instancias (objetos) de una subclase asignándola a una clase superior a su superclase directa, si es que existe. En este caso, creamos una instancia de Gallina y la asignamos a la superclase ReinoAnimal, que está un nivel más arriba de la clase Aves, de la cual extiende la clase Gallina.

|  |
| --- |
|  |

* Sin embargo ahora solo podemos acceder a la método **habla()** que está definido en **ReinoAnimal**. No podemos acceder al método **vuela()** porque está definido en la clase **Aves**.

**Continuando con Clases Abstractas**

«Sigamos, digamos que ahora queremos crear una subclase de **Gallina** al que llamaremos **GallinaEncubadora**, (momento crítico, acá respondo a una pregunta realizada en el grupo que fue motivación a esta publicación). Qué tal si se nos ocurre hacer a la clase **GallinaEncubadora** una clase **abstracta** (Que no tiene sentido, pero responde a la pregunta: ¿Una subclase puede ser abstracta incluso si su superclase es concreta?), lo dejo de tarea, igual si lo prueban hacer “public abstract class GallinaEncubadora extends Gallina” se darán cuenta que si se puede»

|  |
| --- |
|  |

«Bien voy a mencionar un aspecto importante y es que para GallinaEncubadora ya no es necesario redefinir los métodos de la superclase, si acaso llegara a implementarlos estaría sobrescribiendo los métodos de la superclase y serían los valores obtenidos en el momento de la prueba desde la clase Main»

«Conclusión: Si es posible que una subclase sea abstracta independientemente de la clase que se derive, la única restricción para ello es que la clase de la que se desea heredar (superclase) no sea de tipo “**final**” ojo con eso, porque es algo muy importante y no se habló, tampoco he mencionado las funciones o variables con modificador de acceso **protected**»

En java una clase que está definida como **final** no puede ser madre de otra clase. Los atributos o métodos con modificador **protected** de una superclase solo pueden ser accedidos por sus subclases.

¿Qué pasa cuando hacemos esto: ReinoAnimal animal = new Aves()?

**Respuesta**: error: Aves is abstract; cannot be instantiated.

|  |
| --- |
|  |