

Clases abstractas

Habrán ocasiones en las cuales necesitemos crear una clase padre donde únicamente coloquemos la estructura de una abstracción, una estructura muy general, dejando que sean las clases hijas quienes definan los detalles. En estos casos haremos uso de las clases abstractas.

Una clase abstracta es prácticamente idéntica a una clase convencional; las clases abstractas pueden poseer atributos, métodos, constructores, etc. La principal diferencia entre una clase convencional y una clase abstracta es que la clase abstracta debe poseer por lo menos **un** método abstracto. Ok, pero ahora, ¿Qué es un método abstracto? Verás, un método abstracto no es más que un método vacío, un método el cual no posee cuerpo, por ende no puede realizar ninguna acción. La utilidad de un método abstracto es definir qué se debe hacer pero no el cómo se debe hacer.

Veamos un ejemplo para que nos quede más en claro.

```
public class Figura {  
  
    private int numeroLados;  
  
    public Figura() {  
        this.numeroLados = 0;  
    }  
  
    public float area() {  
        return 0f;  
    }  
  
}
```

En este caso la clase posee un atributo, un constructor y un método, a partir de esta clase podré generar la n cantidad de figuras que necesite, ya sean cuadrados, rectángulos, triángulos, círculos etc...

Dentro de la clase encontramos el método área, método que se encuentra pensado para obtener el área de cualquier figura, sin embargo cómo sabemos todas las figuras poseen su propia fórmula matemática para calcular su área. Si yo comienzo a heredar de la clase Figura todas las clases hijas tendrían que sobre escribir el método área e implementar su propia fórmula para así poder calcular su área. En estos casos, en los casos la clase hija siempre deba que sobrescribir el método lo que podemos hacer es convertir al método convencional en un método abstracto, un método que defina qué hacer, pero no cómo se deba hacer.

```
public abstract float area();
```

Ahora que el método área es un método abstracto la clase se convierte en una clase abstracta.

```
public abstract float Figura;
```

Es importante mencionar que las clases abstractas pueden ser heredadas por la n cantidad de clases que necesitemos, pero **no** pueden ser instanciadas. Para heredar de una clase abstracta basta con

utilizar la palabra reservada *extends*.

```
public class Triangulo extends Figura{
```

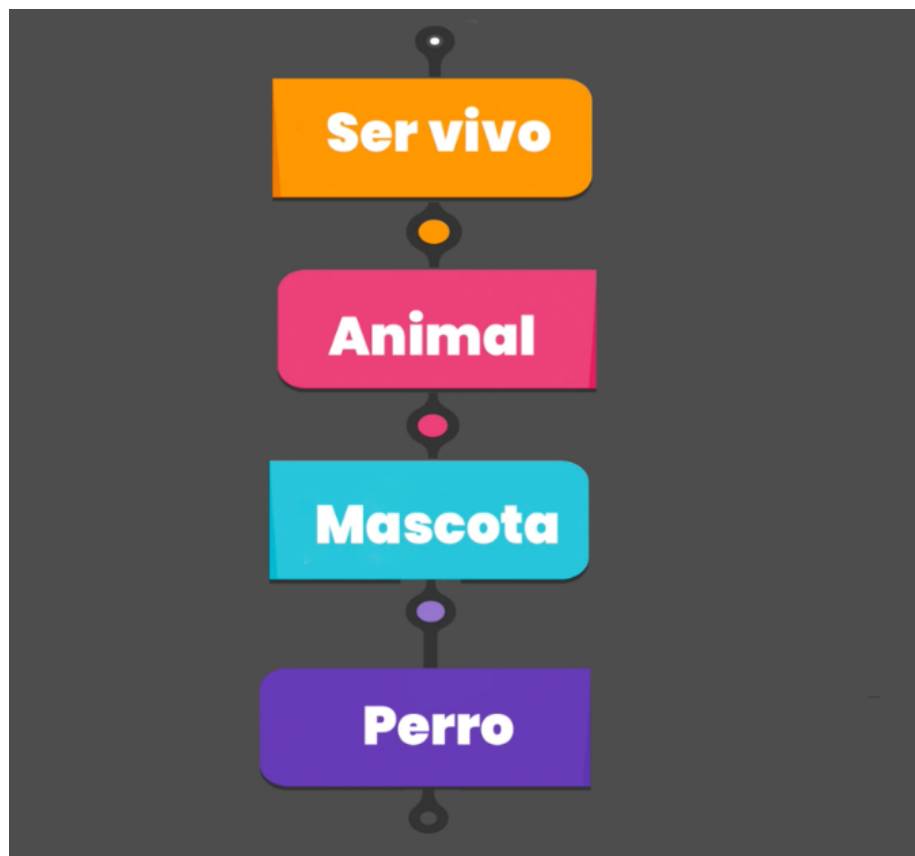
Al nosotros heredar de una clase abstracta es obligatorio implementar todos sus métodos abstractos, es decir debemos definir comportamiento, definir cómo se va a realizar la tarea.

Interfaces

Ahora hablaremos de interfaces. A pesar que es un tema un poco complejo si nosotros hemos comprendido el tema de clases abstractas y el por que de ellas, el tema de interfaces será un tema muy sencillo.

Veamos. A diferencia de otros lenguajes de programación, en Java no es posible la herencia múltiple, nuestras clases únicamente podrán heredar de una y solo una clase.

Si conceptualizamos esto una representación pudiese ser la siguiente.



El nivel de jerarquía es descendente.

Esto sin duda funciona, sin embargo, si queremos representar conceptos de la vida real necesitaremos una jerarquía mucho más compleja, algo como esto.



Para que podamos diagramar nuestro proyecto de esta forma, teniendo en cuenta que únicamente es posible heredar de una clase, entonces haremos uso de interfaces.

Podemos definir a una interfaz como una colección de métodos abstractos y propiedades constantes en las que se especifica que se debe de hacer pero no como, serán las clases hijas quienes definan el comportamiento.

A diferencia de una clase abstracta, una interface no puede hacer nada por sí sola, es prácticamente un contrato, en donde las clases que la implementen deben, obligatoriamente, definir el comportamiento de todos los métodos abstractos, contestando a la pregunta ¿Cómo se debe hacer?

Veamos un ejemplo de una interfaz.

```
public interface Canino{  
    public abstract void aullar();  
    public abstract void ladrar();  
}
```

Cómo podemos observar en la interfaz solo encontraremos métodos abstractos, método vacíos. Para poder implementar la interfaz basta con utilizar la palabra reservada *implements*.

```
public class Perro implements Canino{
```

Si bien es cierto que en versiones actuales de Java podemos encontrar los métodos default en las interfaces, métodos que nos permite definir comportamientos, en esencia las interfaces serán

contratos que indicarán que es lo que se debe hacer sin proveer ninguna funcionalidad.

Otra diferencia entre una clase abstracta y una interface recae en su implementación ya que una clase hija solo podrá heredar de una clase abstracta, por otro lado podrá hacer uso de la n cantidad de interfaces que necesite.

```
public class Perro extends mascota implements Canino{
```

Bibliografía:

- Extraído de [<<Diferencias entre clases abstractas e interfaces en Java>>](#). 26 de septiembre de 2022. Consultado el 28 de noviembre del 2022