¿Qué es la programación orientada a objetos (POO)?

* Se trata de un paradigma de programación que usa **objetos** y sus interacciones para diseñar aplicaciones y programa informáticos.
* Permite describir el problema en términos del mismo problema y no en términos de la computadora en donde va a ser ejecutada la solución.
* Proporciona al programador conceptos y herramientas que permiten modelar y representar los elementos del espacio del problema en donde se quiere resolver la situación.
* A estos elementos de este “espacio de problema” y sus representaciones en el espacio de solución se los denomina **objetos**.

¿Qué es un objeto?

* Es un componente de software que **encapsula** un estado y comportamiento. Cuando hablamos de **encapsulamiento** estamos diciendo que solo se puede acceder a su estado y a su comportamiento desde el mismo objeto.

¿Qué es una clase?

* Una **clase** define los atributos y comportamientos comunes de un conjunto de objetos.
* Las clases actúan de una forma muy parecida a una plantilla o a un molde, en el sentido de que nos permite **crear** o **instanciar** objetos.

Clase vs Objetos

|  |  |
| --- | --- |
| Clase | Objeto (instancia) |
| Todos los objetos de una clase son idénticos en estructura y comportamiento. | Pertenece a una clase en particular. |
| Cada objeto tiene identidad única. | Residen en el espacio de memoria dinámica. |
| Un patrón para la definición del estado y el comportamiento de un tipo particular de objetos. | Los objetos son creados y destruidos en tiempo de ejecución. |

¿Qué es una referencia?

* Es una variable que almacena la dirección de memoria donde reside un objeto.
* Si no referencia a ningún objeto, almacena el valor “null”.
* Estas son almacenadas en la memoria estática. Cuando se carga el programa, se dispone un espacio de almacenamiento fijo para estas.

Creación de un objeto:

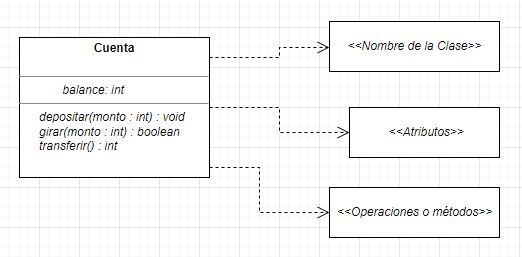
* Un objeto se crea en memoria dinámica.
* Para crear un objeto se utiliza el operador “new”.

Variable (cualquier tipo) vs Referencia

|  |  |
| --- | --- |
| Variable | Referencia |
| Almacena el valor de forma directa. | Está asociada a un tipo de una clase, y almacena una dirección de almacenamiento en donde se encuentra el objeto con el que está relacionado. |

Representación de clases

* Para representar a las clases usamos la **diagramación UML**.



Podemos identificar dos tipos de relaciones entre clases:

* Relación de uso: Dadas dos clases, una ClaseA y otra ClaseB, decimos que existe una relación de uso entre estas sí, una ClaseA hace referencia en algunos de sus métodos a una instancia de la ClaseB o cualquier otra. En la diagramación UML se utiliza flechas punteadas entre las clases.
* Relación de composición: Dadas dos clases, una ClaseA y otra ClaseB, decimos que existe una relación de composición entre estas sí, la ClaseA esta compuesta por el tipo o instancia de la ClaseB o cualquier otra. En la diagramación UML se utiliza flechas rellenas entre las clases.