Capacitarte
Material Complementario

Programación Orientada a Objetos vs Programación Estructurada

Todo lenguaje de programación tiene un paradigma o paradigmas múltiples sobre los que opera. Estos proporcionan diversos conceptos a través del cual los elementos de un programa pueden ser representados y manipulados.

Algunos lenguajes de programación emplean varios paradigmas, que, en este caso, se llama multi-paradigma. Dos de los paradigmas de programación más populares incluyen Programación Procedimental o estructurada y Programación Orientada a Objetos. Estos dos ejecutan los lenguajes más potentes y populares que conocemos, incluidos, entre otros, Java, C, Python y C++.

Es esencial conocer las diferencias entre programación orientada a objetos y programación procedimental. Conocer los conceptos detrás de ellos, las características y los lenguajes que respaldan te guiarán a elegir el lenguaje adecuado para usar en un proyecto en particular.
La programación orientada a objetos (POO) y Programación Procedimental difieren, por lo que no deben confundirse entre sí. A continuación, hay algunas diferencias desatcables:

Definición

No existe un estándar internacionalmente aceptado cuando se trata de definir los términos. En pocas palabras, la programación orientada a objetos es un estilo que trata los datos como objetos con atributos y métodos que pueden aplicarse a estos objetos y también ser heredados por otros objetos. Java es un gran ejemplo de un lenguaje que emplea este concepto. Pero Java es un lenguaje multi-paradigma y también utiliza algunos conceptos familiares para la Programación Procedimental.

La programación estructurada, por otro lado, es un tipo de programación imperativa, donde las declaraciones se ponen en procedimientos, que se pueden volver a llamar cuando sea necesario. C usa programación procedimental.

La POO se centra en clases y objetos. Al representar variables como objetos, se le puede pasar una función (método). Un objeto que pertenece a una clase en particular se puede tratar de forma independiente. La POO puede estar basada en clases, que, en este caso,

los objetos se basan en clases predefinidas. La POO basada en prototipos también existe, por lo que no hay necesidad de clases y solo se utilizan objetos.

La programación procedimental no necesita objetos. Como su nombre lo indica, tiene procedimientos que podrían ser estructuras de datos, rutinas y subrutinas.

Terminología
La terminología utilizada en cada paradigma varía, aunque pueden significar lo mismo. En la programación de procedimental, las funciones se denominan “procedimientos”, mientras que en POO; preferirán ser nombrados como “métodos”.

La nomenclatura de las estructuras de datos también difiere. La programación procedimental los etiqueta como “registros” mientras que POO usa “objetos”. La Programación procedimental usa una llamada de procedimiento para llamar a una función, mientras tanto, POO utiliza una llamada de mensaje para solicitar acciones de objetos.

Herencia
La característica más distinguible del paradigma de POO es la herencia. La herencia da un impulso a POO, ya que permite una facilidad general a través de la cual el código se puede reutilizar y extender sin cambiar el código existente. Los objetos nuevos son capaces de “heredar” las propiedades de los objetos más antiguos. Una subclase puede, por lo tanto, anular un método definido en una superclase. En los casos en que un objeto hereda características de más de un objeto principal, se genera el concepto de Herencia Múltiple.

La programación procedimental no admite la herencia. La herencia solo se puede aplicar a los objetos. Debido a que la programación procedimental carece de objetos, carece de esta característica, por lo que la distingue de la POO.

Subtipificación
En POO, se puede lograr un polimorfismo de subtipo, mediante el cual una función escrita para los elementos de un tipo de datos (supertipo) se puede hacer funcionar en los elementos de otro tipo de datos relacionado (subtipo). El principio de sustituibilidad entra en juego; los objetos de un tipo pueden ser reemplazados por objetos de otro tipo si existe una relación “is-a-subtty-of” entre los tipos. POO es versátil y, como tal, la sustituibilidad se puede implementar sin cambiar otras propiedades.

La programación estructurada no tiene esta habilidad. Como tal, subtipos y supertipos no

pueden ser declarados. Tampoco se puede lograr la subtipificación del comportamiento.

Encapsulación
A diferencia de su contraparte, la POO es capaz de datos vinculantes, así como los métodos que manejan los datos. Forma una cápsula imaginaria que envuelve los datos y métodos, protegiéndolos así de la interferencia externa. La encapsulación es un tipo de abstracción que la POO hace bastante bien. El código se puede escribir para restringir el uso de datos fuera de la cápsula en la que se emplea.

Este ha sido un breve resumen de las diferencias entre la programación orientada a objetos y la programación estructurada.

Fuente: https://blog.educacionit.com/2018/05/21/programacion-orientada-a-objetos-vs-programacion-estructurada/