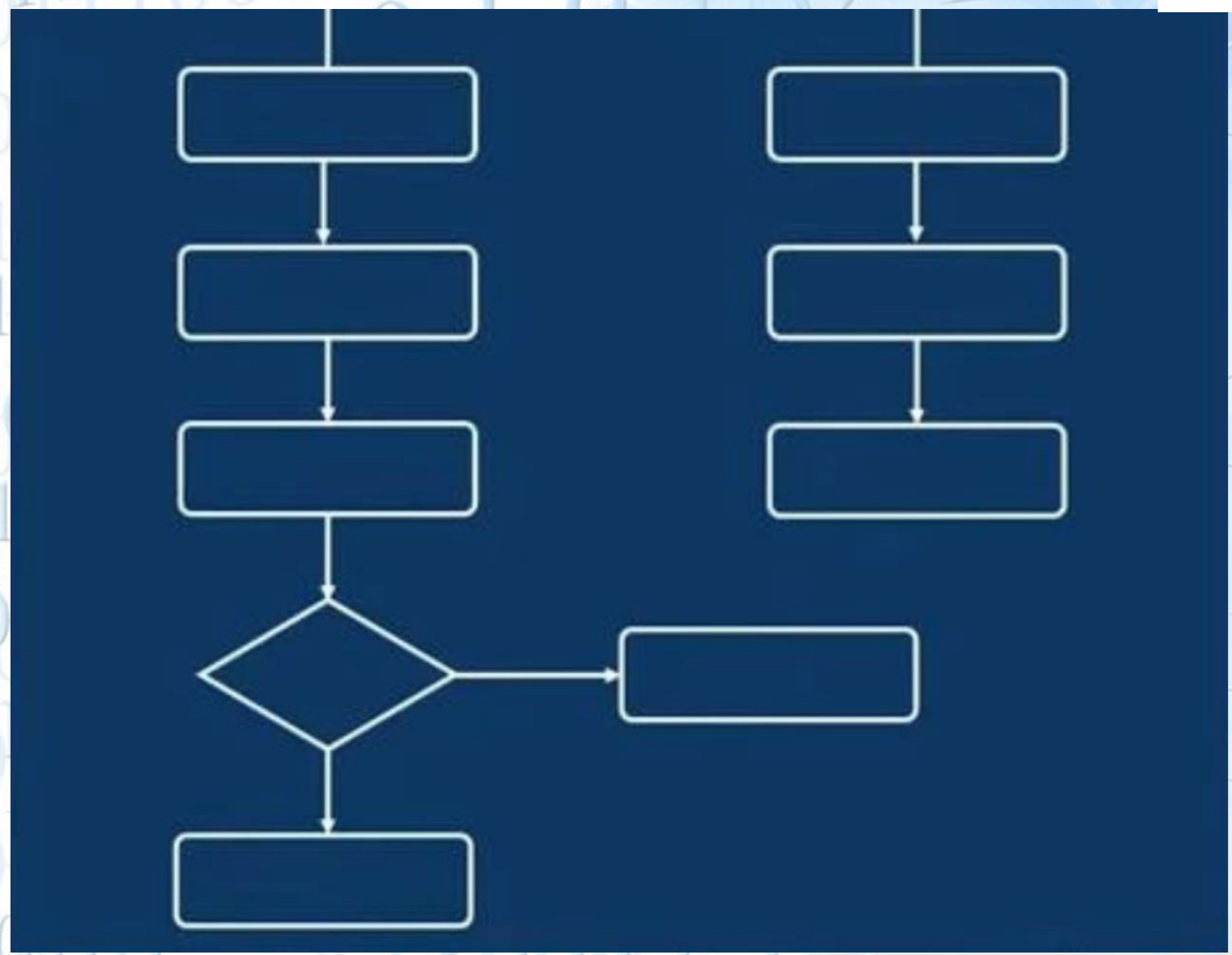

# ¿Qué es la Programación Orientada a Objetos (POO)?

¿Sabes en qué consiste exactamente la POO? Es uno de los conceptos más importantes que debes aprender como programador?

La programación es muy amplia y existen muchas formas de programar. Es tan amplia, que a muchas personas les estresa aprender programación. Sin embargo, cada programador tiene un estilo diferente de programar, y en términos generales, a eso se le llama paradigma.

El paradigma con el que nos enseñan a programar, a la mayoría, es secuencial o estructurado. Es decir, las instrucciones van de arriba hacia abajo, una después de la otra. Simplemente damos una orden, luego otra, leemos unos datos, lo manipulamos con alguna operación, ponemos una condicional para validar ese resultado y, según esos resultados, mándanos una cosa u otra. De arriba hacia abajo, así aprendimos todos a programar.



Pero, cuándo empiezas a trabajar con proyectos más grandes, te das cuenta que este paradigma no te ayuda mucho y allí se te complican las cosas. Vamos a poner un ejemplo muy simple: imagina que un cliente te pide una tienda en línea para vender zapatos. ¿Cómo vas a hacer ese programa de arriba hacia abajo? toca pensar de una manera diferente porque tienes ciertas cosas:

Los zapatos, que pueden tener un precio, color, una marca.

Los productos deben poder filtrarse.

Necesitas un carrito que se conecte con las pasarelas de pago para recibir el dinero.

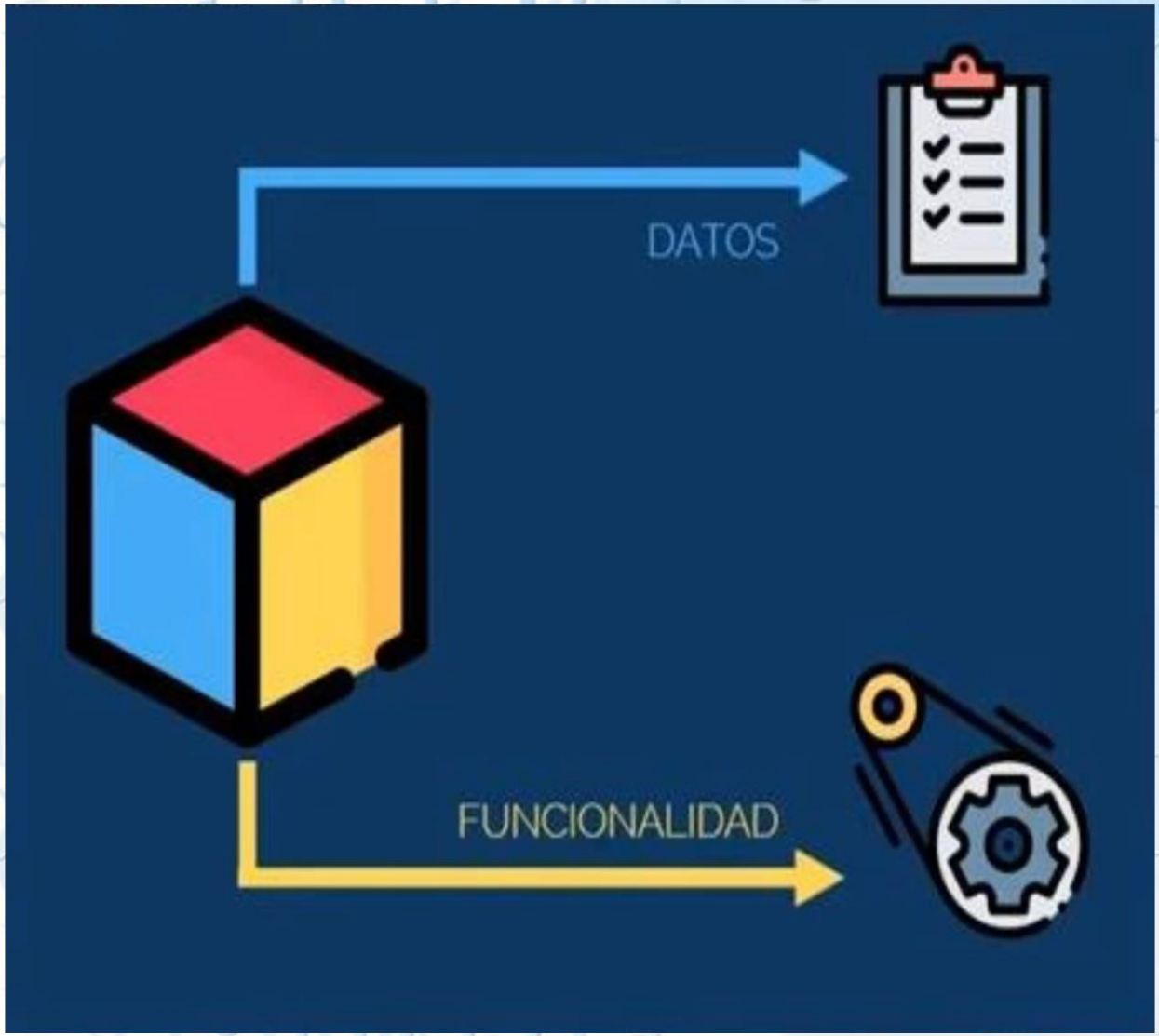
Tienes a los usuarios que van a hacer las compras.

Son muchas cosas que no pueden escribirse de arriba hacia abajo. Y aún así, como es la única manera en la qué sabemos programar, lo hacemos de esa forma. Y cómo también he pasado por allí, sé que es muy difícil cambiar la mentalidad.

# ¿Cuál es el paradigma más usado?

- El paradigma más usado en el mundo, (no el mejor, porque eso es relativo), **es la programación orientada a objetos (POO).** Es el más usado porque cada uno de estos elementos que necesita el sistema o la aplicación, como el carrito, el producto, el usuario, entre otros, es un objeto en este paradigma. Estos objetos tienen sus propios datos y su propio comportamiento. Por ejemplo:


- Los zapatos son un objeto, y por lo tanto, tienen estas dos propiedades:
- **Datos**: marca, precio, nombre.
- **Funcionalidad**: pueden ser comprados o agregados al carrito.
- Los usuarios también se dividen igual:
- **Datos**: su nombre, número de tarjeta de crédito.
- **Funcionalidad**: comprar los productos.
- Lo mismo pasa con el carrito:
- **Datos**: productos agregados al carrito o que usuarios están en el carrito.
- **Funcionalidad**: mandar esa orden de compra a la pasarela de pago, para que se procese y recibir el mensaje si el pago fue exitoso.
- Cada uno de estos elementos en los que vamos dividiendo el sistema, es un objeto, y los objetos, tienen datos y funcionalidades.


DATOS
FUNCIONALIDAD


- Con la Programación Orientada a Objetos pasamos de tener un código de arriba hacia abajo, en el que las funcionalidades están mezcladas y son difíciles de separar o escalar, a un sistema en el que tenemos los elementos (Objetos) separados y se comunican entre ellos:
- El usuario se comunica con el producto para comprarlo.
- El producto se comunica con el carrito para ser agregado.
- El carrito se comunica con la pasarela de pago y con el usuario.
- De esta manera es más fácil manejar y mantener un sistema y hacerlo crecer. Si luego necesitáramos otra funcionalidad, podemos agregar otro objeto, o incluso, agregarle atributos o funcionalidad a los objetos que ya existen.

# Atributos y métodos

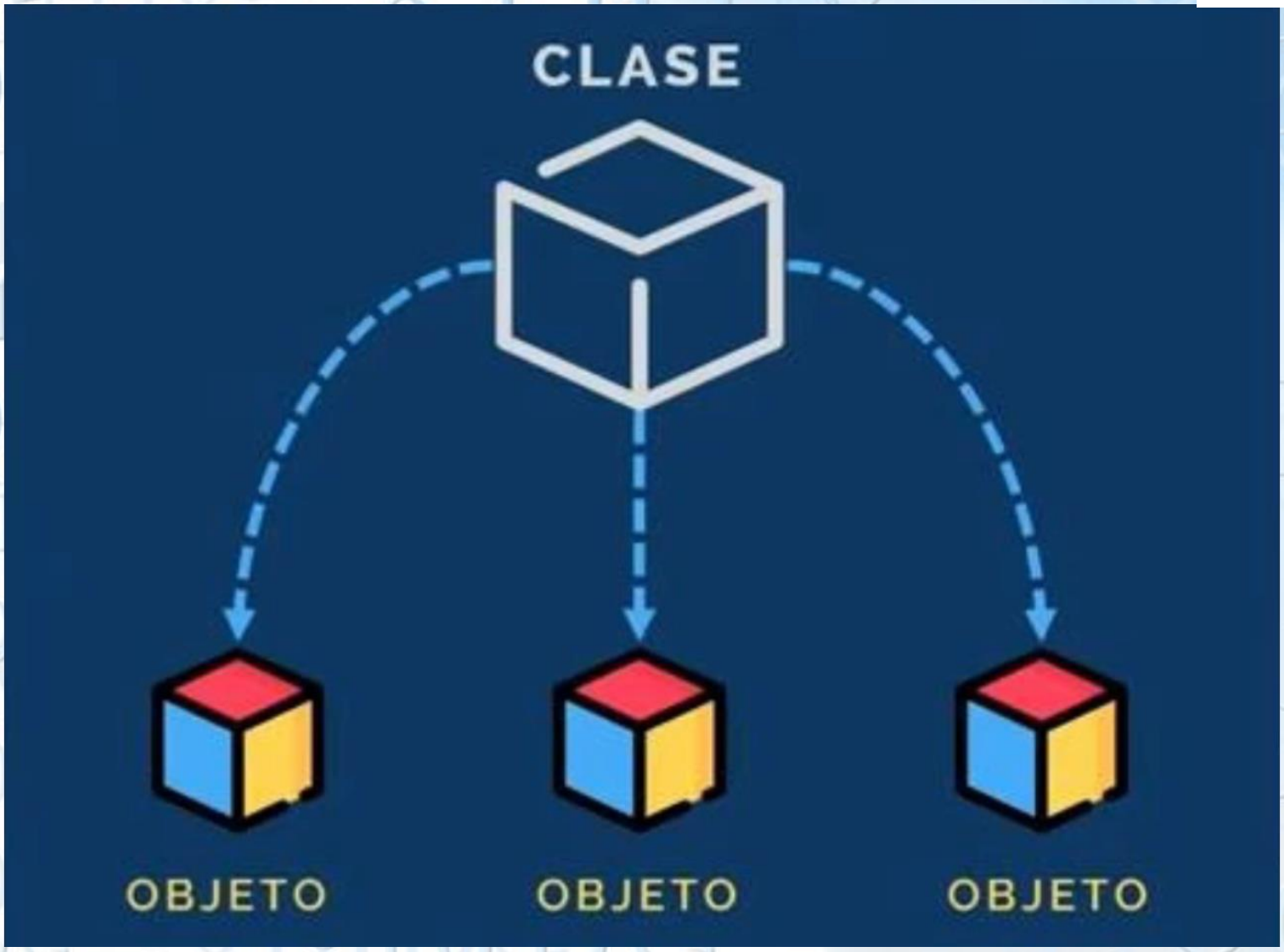
- Como dije antes, los objetos tienen datos y funcionalidad y en la POO se les llama de esta manera:
- Datos → **Atributos**
- Funcionalidad → **Métodos**
- Cada objeto tiene sus atributos y sus métodos. Te pondré un caso de vida real para que quede mucho más claro. Estamos programando la aplicación de cursos de una empresa, y queremos crear usuarios, entonces hacemos un proceso llamado abstracción. Significa pensar los atributos y métodos que debería de tener este usuario para la aplicación.
- Luego de la reflexión, llegamos a la conclusión de que nuestro usuario debe tener: nombres, apellidos, contraseñas y premium (sería un valor que puede ser verdadero o falso).
- A través de este mismo proceso de abstracción, definimos sus métodos: iniciar sesión, cerrar sesión, editar su perfil y contraseña, pasar a premium o publicar un artículo en la comunidad.


ATRIBUTOS
Nombres
Apellidos
Correo
Contraseña
Premium
MÉTODOS
Iniciar sesion
Cerrar sesion
Editar perfil
Cambiar contraseña
Pasar a premium
Publicar en comunidad


- Obviamente en el proceso podemos darnos cuenta que necesitamos también del atributo país, género, fecha de nacimiento y los agregamos. Ya tenemos la estructura y los agregamos sin problemas. Estos objetos se crean en código, obviamente, pero te imaginas que, para cada vez que un usuario quiera registrarse a la empresa, ¿llamemos al programador y lo pongamos a escribir código para el usuario? no sería eficiente. Tendríamos una cola de cientos de usuarios esperando que el programador termine con el actual, para pasar al siguiente.
- ¿Cómo resolvemos este problema? **usamos algo llamado clase.**

# ¿Qué es una clase?

- La clase es una plantilla, un molde, que tiene esa estructura básica del objeto (atributos: datos y métodos: funcionalidad). En el ejemplo anterior, no creamos realmente el objeto-usuario, creamos la plantilla (la clase). Entonces, cada vez que una persona se registra en la empresa, realmente está usando la clase que ya creamos y que ya está en el código, para crear nuevos objetos (usuarios).


CLASE
OBJETO
OBJETO
OBJETO


- Ese proceso de crear objetos a partir de una plantilla, llamada clase, se llama instanciar. Por eso es que cada uno de esos objetos, también se les llama instancia, y de esa manera, es que, con una sola clase, podemos crear decenas o cientos de usuarios sin tener que escribir código nuevamente. Solamente escribimos una vez la plantilla.
- En este punto, debes tener claro, que un objeto tiene datos (atributos) y funcionalidad (métodos ) y que a través de una clase, podemos crear varios objetos. ¿Cómo es el proceso de usar estos objetos en una aplicación real? Te lo mostraré con un ejemplo, para que nunca más tengas dudas.

# Programación Orientada Objetos en una aplicación real

- El siguiente esquema representa el proceso que tiene que seguir un usuario para convertirse en un estudiante premium de esta empresa.
- Tenemos al tío Alex, que para este ejemplo, no conoce la empresa. Un amigo le recomienda la aplicación, porque tiene cursos excelentes y puedes comenzar gratis. Él entra en la web de la empresa y decide registrarse. En el proceso de registro, invoca a la clase usuario y la instancia, y entonces crea su usuario. Ese nuevo usuario es el 528, que tiene como nombre Alex, y de apellido, Lomas.

python™

- **Registro** (Class Usuario)
- Recogemos las **propiedades** del nuevo usuario (nombre, apellidos, contraseña, etc) y creamos el objeto → **usuario_528.**

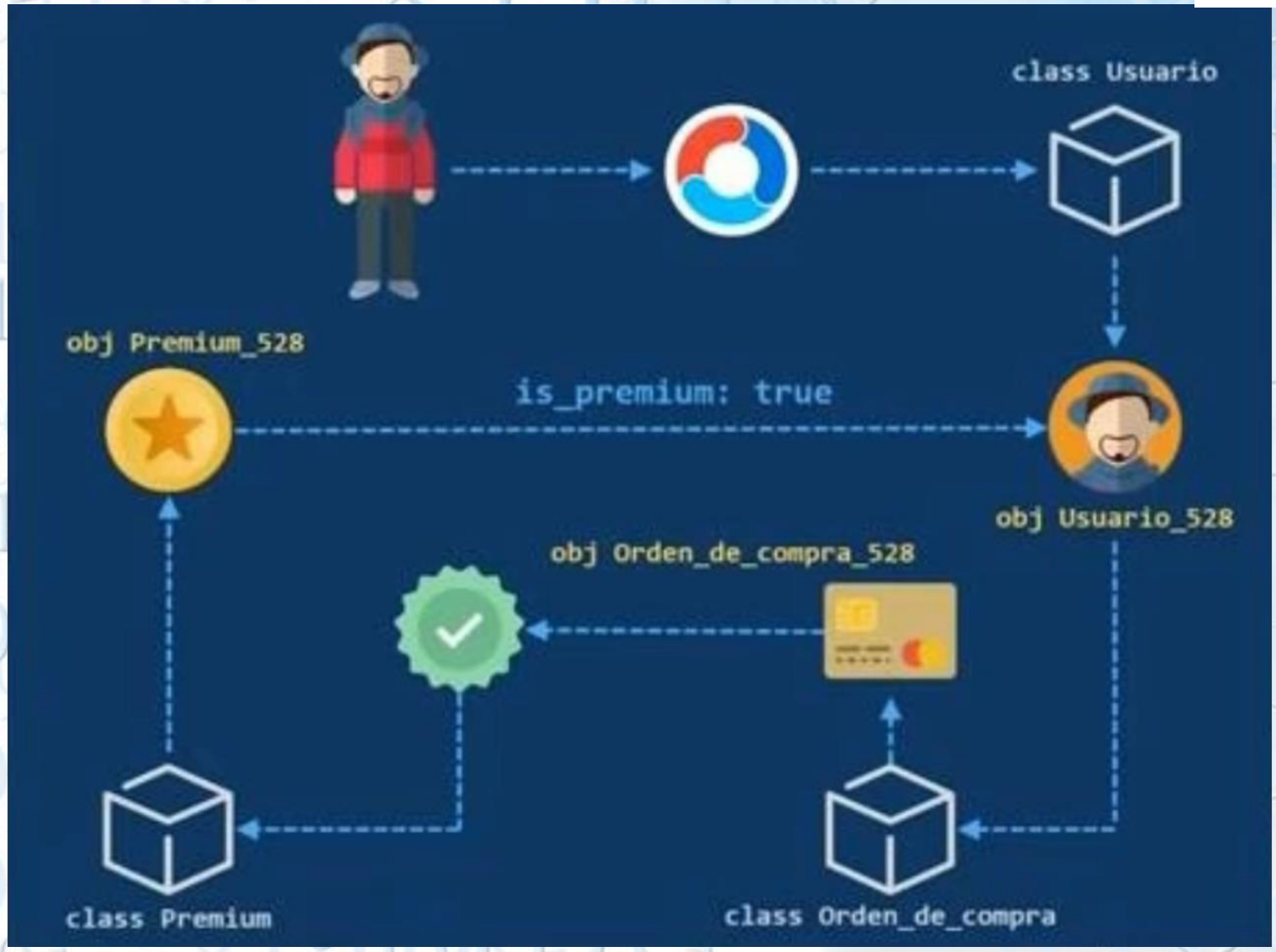

- Este usuario comienza a consumir los cursos gratis de la empresa y le gustan mucho, pero se da cuenta que tendrá acceso a más de 200 cursos si pasa a premium, y no solo a las gratuitos. El usuario va a subir a premium y elige el plan que mejor se adapte a él e introduce sus datos de compra. En ese momento, está llamando a la clase orden de compra, que instancia a un objeto, asociado a este usuario.

python™

- **Pagar** (Class Orden_de_compra)

- Esta clase obtendrá las siguientes **propiedades:** Método de pago, id de usuario, etc y creará una instancia asociada a nuestro usuario_528 → **Orden_de_compra_528.**


- Esta orden de compra se debe procesar, y si es correcto, se valida el pago. Entonces llama a otra clase, la clase premium, y se instancia en un objeto premium, también con el mismo ID del usuario.
- **Pasar a premium** (Class Premium)
- La clase premium tiene como **propiedades**: cursos, especialidades, clases en vivo, etc.
- Se crea el objeto → **Premium_528** y se le asigna los beneficios al usuario_528


class Usuario
obj Premium_528
is_premium: true
obj Usuario_528
obj Orden_de_compra_528
class Premium
class Orden_de_compra


- Entonces, es importante que recuerdes que las clases ya están en el sistema. Los cubos grises de la imagen son las clases, que en el proceso, se instancian para crear objetos reales, como el usuario y orden de compra. Así es como funciona la programación orientada a objetos.
- Recuerda que las clases son las plantillas, y los objetos se crean a partir de esas plantillas. Por ejemplo, el plano de una casa sería el equivalente a una clase, porque a partir de ese plano puedes crear varios objetos. Y el proceso de crear varios objetos a partir de una clase, se llama instanciar.


Clase
Instanciar
Objeto


# Pilares de la POO

- La programación orientada a objetos como paradigma, se basa en cuatro pilares fundamentales: abstracción, encapsulamiento, polimorfismo y herencia. Estos términos son la base de la POO y al comienzo, puede ser un poco confuso entenderlo.

python™

ABSTRACCIÓN

ENCAPSULAMIENTO

POLIMORFISMO

HERENCIA


# 1. Abstracción

- De este término ya unos párrafos más arriba. ¿Te acuerdas? cuando queríamos crear un usuario. El proceso de abstracción es pensar que atributos y qué métodos iba a tener.
- Cuando creamos un sistema, tenemos que hacer una abstracción para todas las clases. Por ejemplo, en la empresa creada, serían: los cursos, las clases, las suscripciones, las publicaciones en la comunidad. Cada uno de estos elementos sería una clase, y tenemos que hacer ese proceso mental: qué atributos y qué métodos van a tener cada uno. Esos conceptos hay que llevarlos y convertirlos en clases.

# 2. Encapsulamiento

- Como ya hemos visto, los objetos se comunican entre ellos. Esto podría traer problemas de seguridad si un objeto puede modificar los datos de cualquier otro. Por eso, se necesita proteger la información de manipulaciones no autorizadas. De esta manera, cuando se comunican los objetos, hay caminos que se pueden seguir y hay caminos que no, datos protegidos, datos privados o públicos, métodos para acceder a cierta información, entre otros. Así se mantiene organizado el sistema.
- Imagínate que cualquier objeto pueda acceder a cualquier información sin autorización, puede ocasionar cambios indebidos, que puede hacer que todo el sistema colapse y se caiga.


# 3. Polimorfismo

- Volvamos a recordar el esquema del tío Alex convirtiéndose en premium.
- Cuando pasas a premium, se notifica vía tres canales en la empresa: slack, web y correo. La primera es privada para nuestro equipo, mientras que las últimas dos, es visible para el usuario, confirmándole que ya es premium.
- Si te das cuenta, es la misma funcionalidad (notificación), aunque internamente cada uno de esos métodos funciona diferente. Es decir, la notificación de Slack tiene su propia API, métodos y lógica, igual para la web y el correo. Pero en esencia, es la misma acción. Eso nos simplifica mucho el trabajo porque podemos dar ordenes coherentes a varios objetos sin preocuparnos cómo se ejecutarán.
- Por ejemplo, si le decimos al tío Alex y a mí que bailemos, cada uno lo hará de forma diferente, cada uno a su modo. Sin embargo, nos dieron una orden y la cumplimos, aunque a nuestra manera particular. Eso es el polimorfismo, poder darle la misma orden a diferentes objetos y que cada uno de ellos respondan a su propia manera.

# 4. Herencia

- Este pilar de la POO es, quizás, el más fácil de entender porque tiene relación con el mundo real. Por ejemplo, mis hijos han heredado muchas cosas de mí, tanto atributos como funcionalidades. En atributos, podemos decir que el color de ojos, el cabello o el color de piel. En la funcionalidad, el carácter o personalidad. Sin embargo, ninguno es una copia exacta de mí. Sobre esas cosas que han heredado, tienen sus propios atributos y funcionalidad.
- Es lo mismo que en la programación orientad a objetos. Tenemos una clase padre, y las clases hijas heredan funcionalidades y atributos, pero no son idénticas. Solamente aprovechan eso que ya existen y luego se le añaden nuevas cosas.


- Por ejemplo, si tenemos una clase para crear usuarios genéricos, pero luego necesitamos un usuario diferente, de staff, solamente para el equipo de la empresa y que tenga diferentes funcionalidades y atributos que un usuario normal ¿Qué podemos hacer? crear una nueva clase que herede de la clase padre, y esa sería el staff.
- Si luego necesitamos otro usuario que sea profesor, entonces heredamos de la clase usuario y creamos la clase profesor, y de allí, creamos todos los usuarios de profesores del equipo. Ahora bien, si queremos meter invitados a la página, personas que sin pagar una suscripción puedan tener acceso a los sorteos o campañas, podemos crear un rol de invitados. Y así funciona la herencia en la POO.