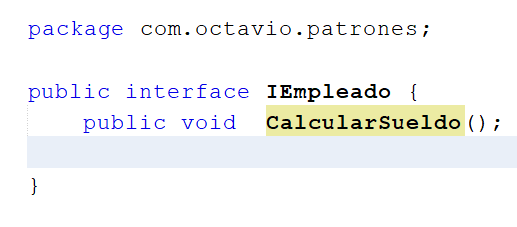
EXAMEN: OCTAVIO RUEDA HERNANDEZ

PATRONES DE DISEÑO DE SOFTWARE.

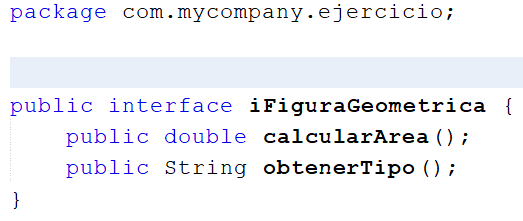
* Describa la diferencia e importancia de cada uno de los pilares de la programación orientada a objetos demostrando cada caso con un ejemplo práctico (descrito y en código).

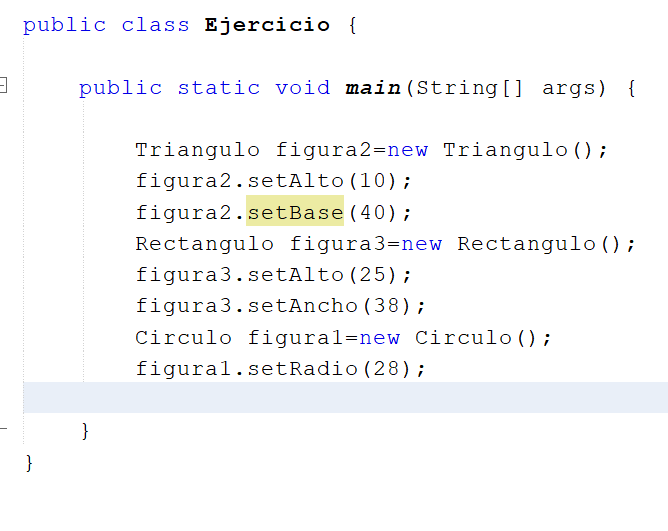
|  |  |  |
| --- | --- | --- |
| CUADRO COMPARATIVO | | |
| PILAR | IMPORTANCIA | EJEMLO PRACTICO (DESCRIPCION) |
| ABSTRACCION | 1. Debido a que la POO trata de modela la realidad en el código, la abstracción además de un pilar es una buena herramienta debido a que solo recopilamos las características principales de las entidades del mundo real y las plasmamos en el código fuente. | Un ejemplo serio los empleados los cuales para una empresa todos son iguales lo que cambia es su sueldo y sus funciones |
| POLIMORFISMO | 1. Este pilar está relacionado con la manera en la cual las clases y los objetos se adaptan al cambio, por tanto, como su nombre lo indica “poli de varios” y “morfismo que tiene la cualidad de cambiar de forma” | Ejemplo claro será la manera de calcular el área de una figura geométrica o hallar el volumen de recipiente, el cual a pesar de que sean figuras diferentes o recipientes diferentes la clase estará en la capacidad de calcular el área o el volumen de un recipiente de manera diferente. |
| ENCAPSULAMIENTO | 1. Podemos controlar cierto tipo de cosas como el cambio de valores de un objeto de manera no controlada. 2. Este pilar también permite separar las interfaces de su implementación esto nos muestra que podemos saber con detalle que hacen los atributos de dicha clase, solo saber que existen. | Cuando tenemos los datos de un objeto y no queremos que sea modificado los datos de dicho objeto de manera directa por el usuario, así mantenemos la integridad de dichos datos, para el caso de ejemplo usare una clase llamada AbogadoPenal la cual tiene atributos encapsulados para calcular el sueldo |
| HERENCIA | 1. La herencia, a mi parecer es de los pilares más fuertes de la programación orientada a objetos, debido a que este nos permite eliminar gran parte de la duplicidad de código y la implementación de clases. 2. Este pilar pude facilitar la extensión de aplicaciones a gran escala, además podemos facilitar el mantenimiento de clases y subsistemas muy grandes. | Haciendo uso de pilar de la abstracción podemos intuir que si abastaremos los empleados de un bufete de abogados y creamos una interfase de empleados, con un método calcular sueldo, todos los abogados puedes usar este método e implementarlo de manera diferente a las clases hijas |

* Código de ejemplos mencionados en la tabla.
  + Abstracción

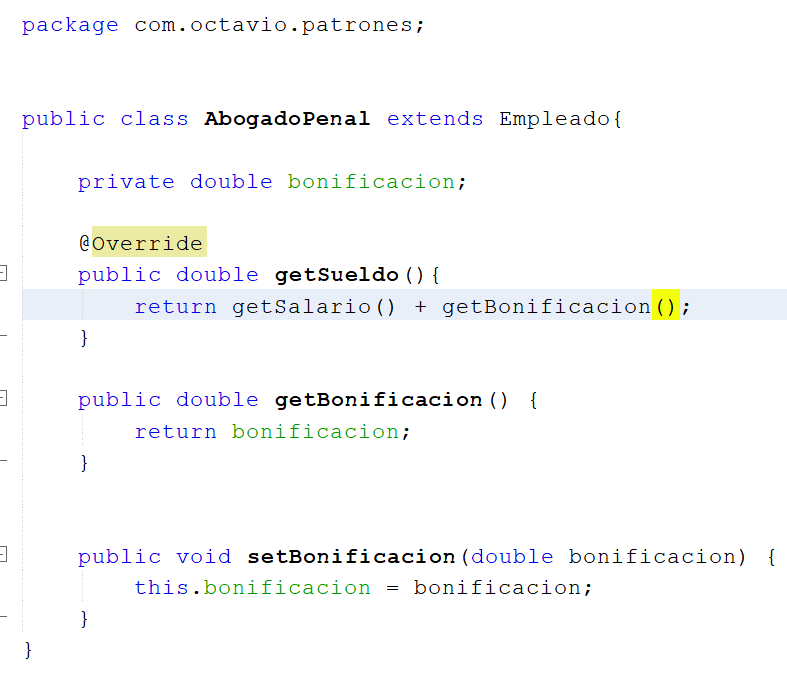


* + Polimorfismo

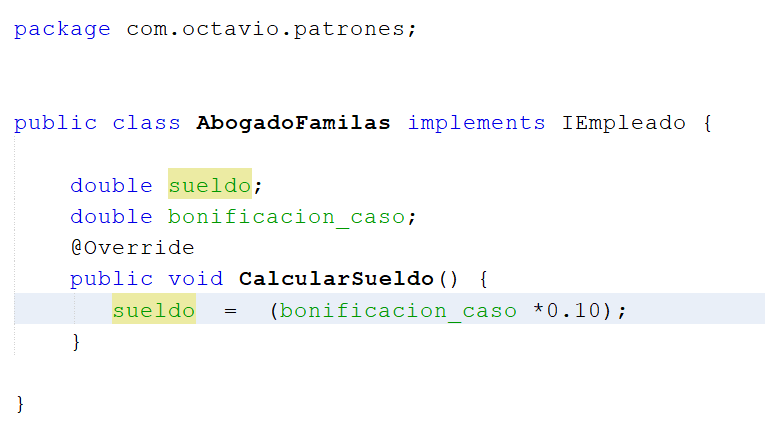




* + Encapsulamiento



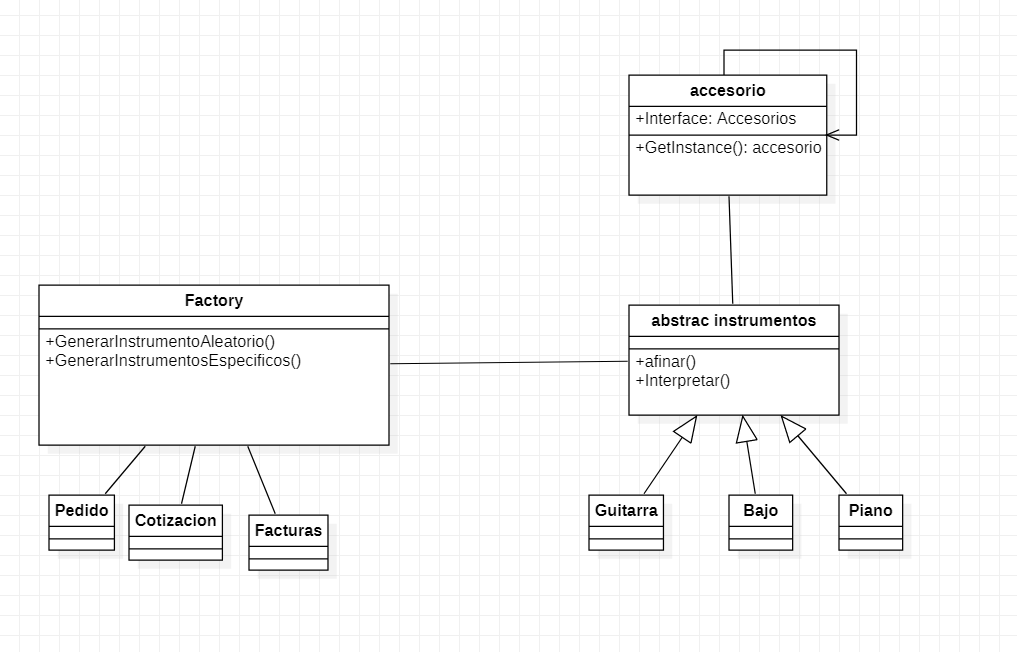
* + Herencia



* Usted hace parte de un equipo de desarrollo de software que está realizando la sistematización de una empresa encargada de producir instrumentos musicales (Guitarra, Bajo y Piano) con la acción de afinar y la de interpretar para ellos, y estos se pueden crear de forma directa en los módulos de pedido, orden de compra, de facturación y cotizaciones, pero se presenta la necesidad de que puedan crear de forma aleatoria, también se pueden crear de forma específica según el usuario haga el requerimiento, adicionalmente se estarán produciendo elementos complementarios a estos instrumentos que son accesorios como calcomanías y afinador de instancia única, se presenta la necesidad de que se puedan tener estos de tipo eléctricos y no eléctricos. Plantea el o los patrones que puedes aplicar entre Singleton y Factory, elabora el diagrama de clases para la solución e impleméntalo en el lenguaje de programación de tu preferencia. Documenta el código y donde apliques cualquier pilar de la programación orientada a objetos destácalo.

Requerimientos.

* + producir instrumentos musicales (Guitarra, Bajo y Piano) con la acción de afinar y la de interpretar.
  + Generar elementos de forma aleatoria.
  + módulos de pedido, orden de compra, de facturación y cotizaciones.



* Describa tres situaciones del mundo real donde puede aplicar el patrón de diseño de software Singleton.

1. La creación de un código de acceso de una persona a una entidad, se podría manejar con este patrón debido a que podemos crear o instanciar este código si no existe si existe no ay que crearlo.
2. Una forma de aplicar este patrón en la vida real seria la manera de ingresar a un sitio tomaremos como ejemplo una biblioteca y ya existe la puerta de entrada de la biblioteca no ay necesidad de crear otra puerta con el mimo uso, esto haciendo referencia a uno de los ejemplos mas utilizados para este patrón como lo es la apertura de las conexiones a bases de datos.
3. Un ejemplo significativo seria si nos ubicamos dentro de una empresa o entidad para el caso de ejemplo usare un bufete de abogados los cuales sacan las impresiones desde la misma impresora por cada departamento, usaría singleton para gestionar la impresora para procesar las consultas sin modificarlas y, en última instancia, imprimirlas.