Redictado OO1 - 2024
Parcial simulado 29/5

Nos contratan de la agencia de recaudación de la provincia de Buenos Aires para hacer un sistema para el cálculo del impuesto que deben pagar los contribuyentes. Un contribuyente paga impuestos por los bienes que tiene a su nombre, que pueden ser bienes inmuebles (casas, terrenos, etc.) y bienes muebles (hoy se registran motos y autos, en el futuro habrá otros, cada uno con su propio cálculo de impuestos). En esta primera etapa, el sistema debe ofrecer la siguiente funcionalidad:

Dar de alta un contribuyente: Se provee nombre, dni, email y localidad. El sistema da de alta al contribuyente y lo retorna. El contribuyente no tiene ningún bien a su nombre.

Dar de alta un inmueble: se provee el número de partida, el valor del lote, el valor de la edificación, y el contribuyente (propietario). El sistema da de alta el inmueble y lo retorna.

Dar de alta un automotor: se provee patente, marca, modelo, fecha de fabricación, valor y el contribuyente (propietario). El sistema da de alta al automotor y lo retorna.

Calcular el impuesto que debe pagar un contribuyente: dado un contribuyente, se debe calcular cuánto debe pagar de impuestos, según la siguiente especificación:

- Por cada inmueble que posea, el contribuyente debe pagar un 1% del valor del mismo, que se calcula sumando el valor del lote y el valor de la edificación.
- Por cada uno de los otros bienes, se debe pagar un porcentaje del valor de los mismos, en función de su fecha de fabricación.
    - En caso de **superar los 10 años**, no deben pagar nada.
    - En caso contrario, el porcentaje para un automotor es el **5%**.

Su tarea es diseñar y programar en Java lo que sea necesario para ofrecer la funcionalidad antes descrita. Se espera que entregue lo siguiente:

1. Diseño de su solución en un diagrama de clases UML.
2. Implementación en Java de la funcionalidad requerida.
3. Implementación de los tests para **la funcionalidad de calcular el impuesto para los bienes muebles**, justificando su elección en base a valores de borde y particiones equivalentes.

Notas:

- Para calcular los años entre dos fechas puede utilizar la siguiente expresión

```
ChronoUnit.YEARS.between(fecha1, fecha2);
```

Donde fecha1 es anterior a fecha2. La expresión retorna la cantidad de años entre ambas fechas

- Implemente **todos** los constructores que considere necesarios.
- Puede implementar un getter y un setter, y asumir la existencia del resto.

Diseño

- Se espera una clase Contribuyente con la responsabilidad de dar de alta inmuebles, automotores y embarcaciones. También debe presentar la responsabilidad de calcular el impuesto a pagar.
- Se espera una interfaz Valuable (o similar) con el protocolo para calcular impuesto a pagar. Podría ser también una clase abstracta Bien, hay que ver en detalle si se justifica en la solución, por ejemplo, definiendo una variable en común "valor" (no es ideal, pero es analizable).
- Se esperan una clase Inmueble que implementa Valuable y una clase abstracta Vehículo (o similar) con subclase Automotor que define de forma individual la lógica de calcular el impuesto, La clase abstracta Vehículo define atributos comunes como patente, fecha de fabricación,
- BIEN puede ser tanto una clase abstracta como una interface

Corrección

Desaprueba si

- No tiene las clases que esperamos (la clase que generaliza Automotor debe estar presente para poder agregar otros bienes en futuro)
- Le falta el método calcular impuesto
- No resuelve polimórficamente las distintas formas de calcular impuesto, o tiene varias listas, una con inmuebles, otra con autos y otra con embarcaciones. Le hace repetir código.
- No implementa constructores o lo hace de forma inconsistente (por ejemplo, en el setup del test llama a constructores que no existen y no se puede ver cómo se forma el "grafo" de objetos
- Si hizo una clase "Sistema/Arba/Agencia" que implementa todos los métodos de alta de bienes no delegando en el contribuyente que se recibió como parámetro.

Corregimos pero es un error leve si:

- Le falta algún caso borde en los tests, pero están bien planteados los otros casos.
- Calcula mal el porcentaje (error en la operación algebraica 15*valorInmueble/100).
- Errores de sintaxis java.
- Relación "doble" entre Contribuyente y Bien sin consistencia
- Repite IF antigüedad > 10