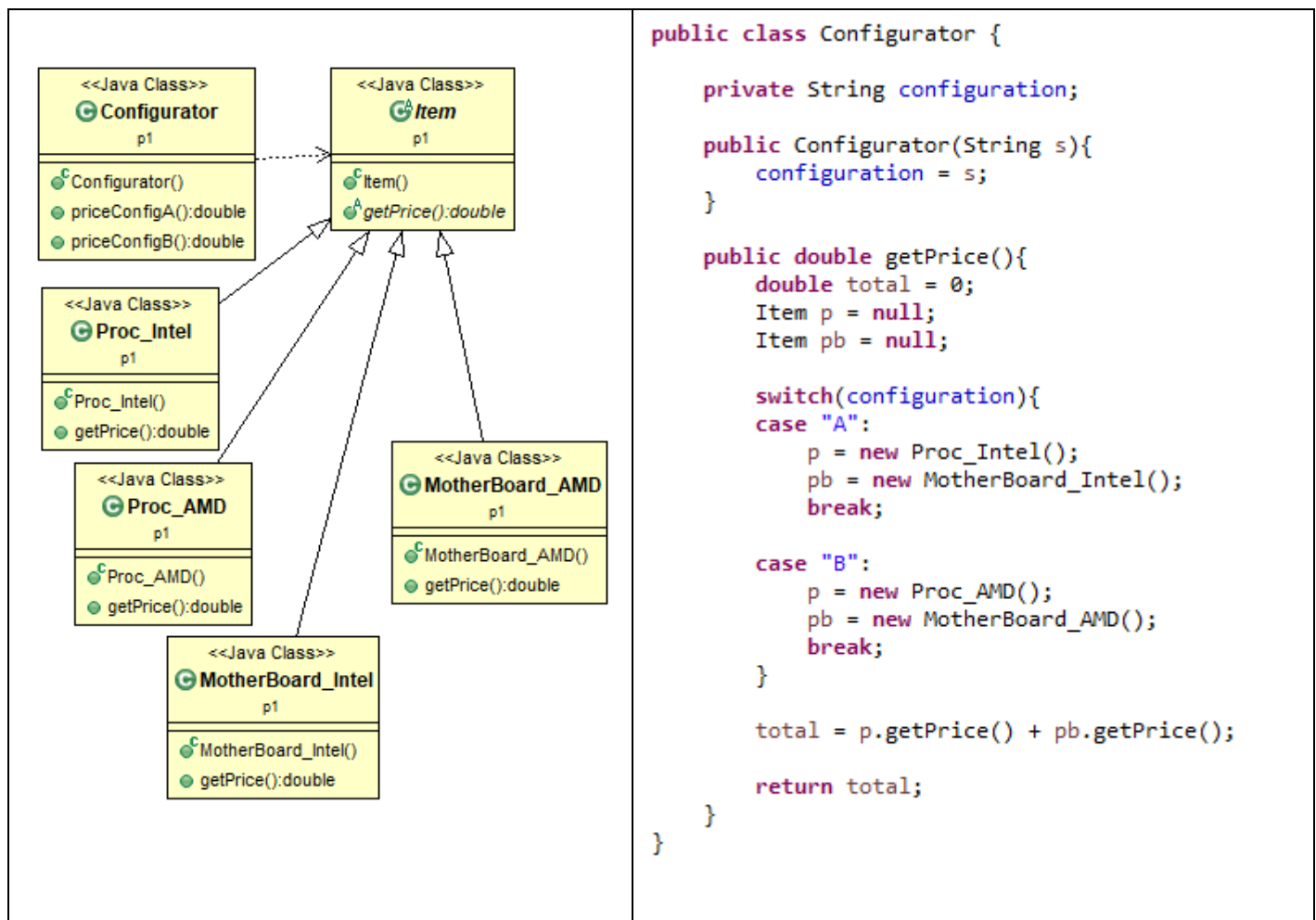


La empresa CholloComponents vende por Internet una gran variedad de componentes de Ordenador, en concreto distintos modelos de placas base y procesadores. Para modelar los componentes en su sistema de información utiliza la siguiente representación de la clase **Item** y sus distintas subclases:



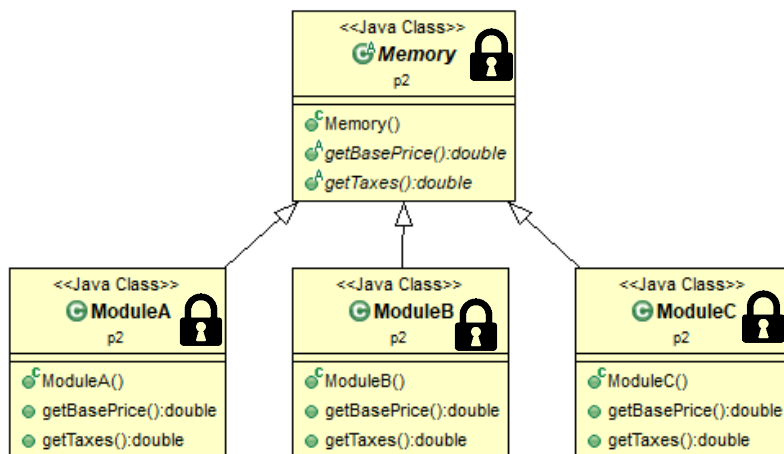
La aplicación web utiliza la clase **Configurator** para ofrecer el precio de distintas configuraciones prediseñadas. En la actualidad hay dos posibles configuraciones de componentes: A (placa base y CPU de Intel) y B (placa base y micro de AMD), las configuraciones no son al azar, sino que los micros y placas de Intel son incompatibles con los micros y placas de AMD, y viceversa. Así para obtener el precio de una configuración A bastaría con:


```
Configurator cfg = new Configurator("A");
System.out.println(cfg.getPrice());
```

El principal problema de este diseño es que es posible generar nuevas configuraciones con componentes incompatibles (ej: micro Intel y placa AMD). Se necesita un cambio de diseño que permita generar estas o nuevas configuraciones de placa base y procesador pero que siempre sean compatibles entre sí.

1.- Selecciona el patrón más adecuado para esta tarea. Justifica tu respuesta. Indica su tipo, y si el patrón utilizado tiene varias versiones justifica cual utilizarías. **[1 punto]**

La empresa se propone incorporar en su página web la venta también de módulos de memoria, para ello llega a un acuerdo con la empresa **FullIRAM** que le ofrece sus dispositivos en venta, pero debe respetar el siguiente diseño de clases para ello: (el precio total de un módulo de memoria sería la suma de su precio base, más el de sus impuestos)







El  indica que el código de la clase no es modificable.

Para no tener que modificar el diseño de la aplicación Web, nuestra empresa necesita incorporar los módulos de memoria a su sistema, sin modificar la jerarquía de **Memory** ni las clases ya existentes de la jerarquía de **Item** de nuestra empresa, para que se pueda manipular cualquier módulo de memoria como otro **Item** más.

2.- Selecciona el patrón más adecuado para esta tarea. Justifica tu respuesta. Indica su tipo, y si el patrón utilizado tiene varias versiones justifica cual utilizarías, y además si el resto de las versiones se podrían utilizar o no. [1 punto]

Nuestra empresa dispone de una *pasarela de pago* contratada con un banco para gestionar todos los pagos con tarjeta de crédito de sus clientes en la Web. Para ello el banco le proporciona la siguiente clase que actúa de pasarela de pago.

<div><div><<Java Class>></div><div><div> CreditCardGateway</div><div>p3</div></div></div> <div><div> CreditCardGateway()</div><div> setup(String):boolean</div><div> payment(CreditCard,double):boolean</div></div>	<pre>public class CreditCardGateway { public boolean setup(String CompanyHashCode) { // In the setup process the company send its HashCode // to login in the bank // and return true if everything is ok return true; } public boolean payment(CreditCard card, double amount) { // if the company is previously logged and // the credit card is correct // the bank make the requested payment and return true return true; } }</pre>
---	---

Esta clase la recibimos en un fichero CreditCardGateWay.class ya compilado, luego no podemos modificar el código de la clase. Necesitamos un desarrollo software que verifique que todas las compras se hagan desde un mismo objeto pasarela, ya que el banco nos cobra en función del número simultaneo de instancias activas en nuestra web. Y adicionalmente verifique los siguientes requisitos:

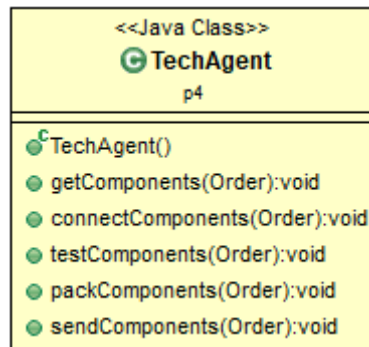
- Para poner en funcionamiento la pasarela es necesario invocar previamente al método setup() proporcionándole el código hash (o clave) a la que debe tener acceso solo las personas autorizadas, por lo que se requiere que en el código fuente solo aparezca en un único fichero.
- Se debe impedir por diseño el que los programadores de la Web tengan acceso a la creación de más pasarelas de pago.
- La pasarela ha de estar directamente accesible desde cualquier módulo del proyecto de la Web.
- Solo se lanzará la pasarela de pago y su procedimiento de configuración en caso de necesitarse un pago por tarjeta de crédito, en otro caso ni si quiera se creará la pasarela.
- Se está negociando con el banco para que nos ofrezca por la misma tarifa disponer de hasta cuatro instancias simultáneas de la pasarela, así que el diseño de software debe cubrir también esta posibilidad.

3.- Selecciona el patrón más adecuado para esta tarea. Justifica en tu respuesta de qué forma se cubrirían los cinco requisitos anteriores. Si el patrón utilizado tiene varias versiones justifica cual utilizarías. **[1 punto]**

Para la gestión de los pedidos realizados por los clientes en su Web, la empresa dispone de una plantilla de técnicos. Todos ellos son capaces de realizar las diferentes etapas de un pedido:

- Acopio de componentes del almacén
- Ensamblaje de los diferentes componentes
- Verificación del correcto funcionamiento de los componentes
- Empaquetado de los componentes ensamblados
- Etiquetado del paquete para su envío

Cada técnico está modelado como un objeto de la clase **TechAgent**.



Para gestionar un pedido es necesario invocar desde la clase **Configurator** a su método **processOrder()**:

```
public class Configurator {

    // ...

    public void processOrder(Order o) {
        TechAgent t1 = new TechAgent();

        // Ensemble components
        t1.getComponents(o);
        t1.connectComponents(o);
        t1.testComponents(o);

        // pack and send
        t1.packComponents(o);
        t1.sendComponents(o);
    }
}
```

Se ha decidido que diferentes técnicos puedan gestionar las diferentes etapas de un pedido, y además que se agrupen en una macrooperación las labores de ensamblado y en otra las labores de empaquetado y envío. También se entiende que en el futuro puedan aparecer o eliminarse diferentes etapas de la gestión de un pedido, sin que esto suponga tener que modificar el diseño de la aplicación.

4.- Selecciona el patrón más adecuado para esta tarea. Justifica tu respuesta. Indica su tipo, y si el patrón utilizado tiene varias versiones justifica cual utilizarías. **[1 punto]**