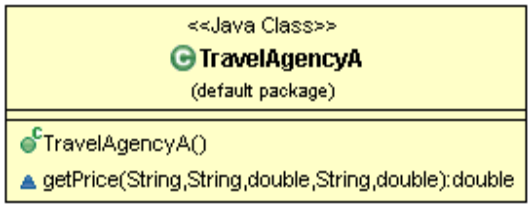
ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

TECNOLOGÍAS Y PARADIGMAS DE LA PROGRAMACIÓN

Control Práctico Patrones-Selección de Patrones. Jueves 28 de mayo de 2020.

Una agencia de viajes A ofrece distintos paquetes vacacionales a distintos lugares del mundo. Para ello se dispone de la clase TravelAgencyA:



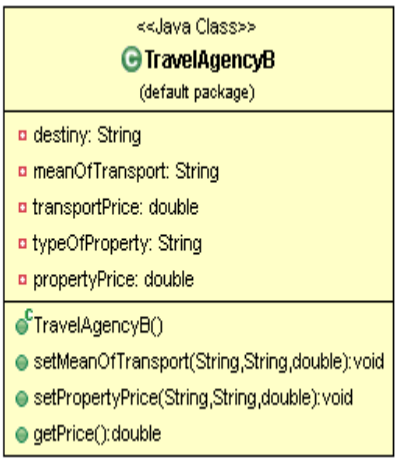
El coste de dicho paquete vendrá dado por el medio de transporte utilizado para llegar al destino, así como el tipo de alojamiento (hotel, apartamento o resort) .

Ejemplo:

```
public static void main(String args[]) {
      //Holidays Package data
      String myDestiny = "Venice";
      String typeOfTransport = "Plane";
      double transportPrice = 350.0;
      String typeOfProperty = "hotel";
      Double propertyPrice = 1250.75;

      //Working out the price
      TravelAgencyA travelAgencyA = new TravelAgencyA();
      double priceA = travelAgency.getPrice(myDestiny,typeOfTransport,
                        transportPrice, typeOfProperty, propertyPrice);
}
```

1. Suponiendo que se dispone de la agencia de viajes TravelAgencyA, se desearía trabajar con una segunda agencia de viajes B TravelAgencyB, con el fin de buscar el paquete vacacional que resulte más atractivo al cliente desde el punto de vista económico. La definición y uso de la nueva clase sería la siguiente:



El precio proporcionado por la agencia B se calcularía como:

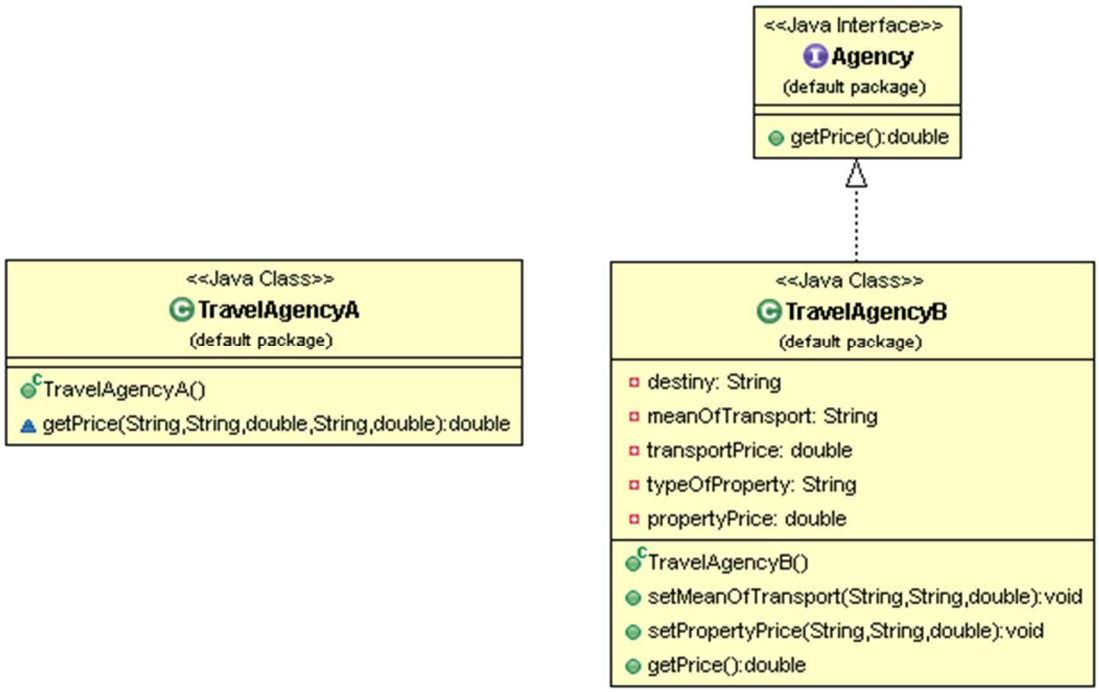
```
//Working out the price for Agency B
TravelAgencyB travelAgencyB = new TravelAgencyB();

//Setting the destiny, transport and its price
travelAgencyB.setMeanOfTransport(myDestiny,
          typeOfTransport,  transportPrice);

//Setting the destiny, type of property and its price
travelAgencyB.setPropertyPrice(myDestiny,
          typeOfProperty,  propertyPrice);

double priceB = travelAgencyB.getPrice();
```

Se desea trabajar de forma indistinta tanto con objetos de TravelAgencyA como de TravelAgencyB por medio de una interfaz denominada Agency con el método getPrice(), el cual tiene el mismo prototipo que el de TravelAgencyB.



Tanto **TravelAgencyA como TravelAgencyB NO SE PUEDEN MODIFICAR!!**

¿Cuál sería el patrón más adecuado para poder usar TravelAgencyA a través de la interfaz Agency? Justificar la respuesta. [1 punto]

2. Se incorpora nueva agencia de viajes TravelAgencyC, de acuerdo al diagrama de clases de la siguiente figura:

El cálculo del precio se debería de realizar de la siguiente forma:

```
//Working out the price for TravelAgencyC
Transport t = new Transport("plane", 250.0);
Property p = new Property("hotel", 2500.0);

TravelAgencyC travelAgencyC = new TravelAgencyC();
travelAgencyC.setTransport(t);
travelAgencyC.setProperty(p);

double priceC = travelAgencyC.getPrice();
```

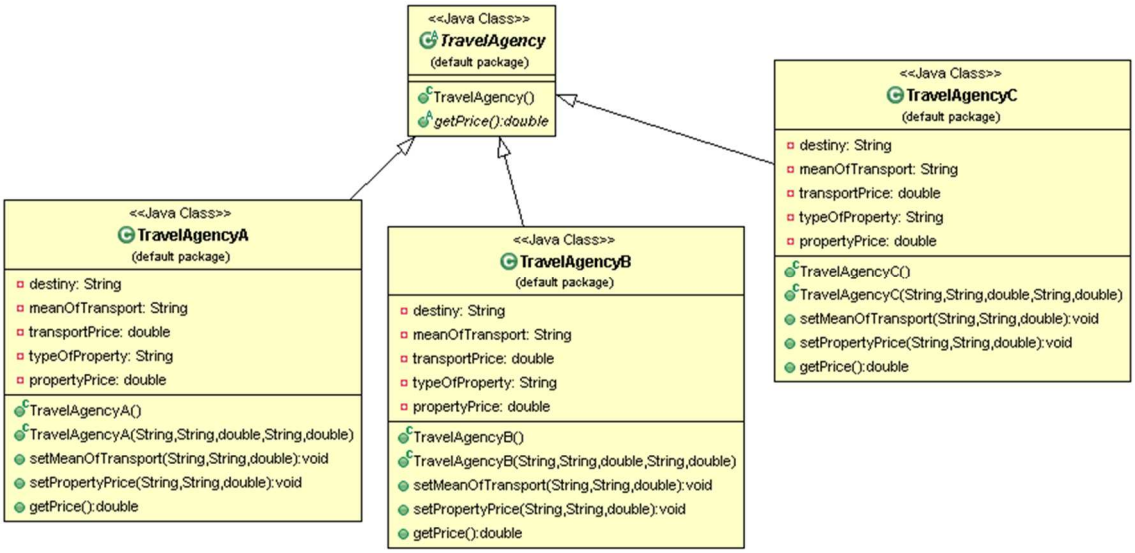
La clase TravelAgencyC **NO PUEDE SER MODIFICADA!!!!**

Se pide:

a) ¿Se podría simplificar el uso de TravelAgencyC, Transport y Property, para que fuese igual de simple que en la clase TravelAgencyA? Justificar la respuesta. [1 punto]

b) En el caso de que se desease que el medio de transporte fuese único, ¿qué tipo de patrón debería de ser usado para la citada clase?. [1 punto]

3. Si las agencias de viajes se modelasen de la siguiente forma:



En la clase TravelAgency, el método getPrice() debería de ser abstracto y ser definido en cada clase. Por otro lado, cada vez que se necesitase una agencia de viajes concreta, entonces se tendría que crear el siguiente código:

```
TravelAgency travelAgency; //setting the travel agency

//type is used for setting the type of travel agency
if(type == 1)
      travelAgency = new TravelAgencyA();
  else if (type == 2)
            travelAgency = new TravelAgencyB();
      else if(type == 3)
            travelAgency = new TravelAgencyC();

travelAgency.getPrice();
```

De acuerdo al código anterior, si apareciesen más agencias de viajes nuevas, entonces sería necesario añadir más sentencias condicionales.

¿Se podría pensar en algún tipo de patrón alternativo que evitase añadir sentencias condicionales cada vez que apareciese una nueva agencia de viajes? Justificar la respuesta. [1 punto]