




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:

<<Java Class>>	
	TravelAgencyA
(default package)	
<hr/>	
	TravelAgencyA()
	getPrice(String,String,double,String,double):double











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 = travelAgencyA.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:

<<Java Class>>	
	TravelAgencyB
(default package)	
<hr/>	
	destiny: String
	meanOfTransport: String
	transportPrice: double
	typeOfProperty: String
	propertyPrice: double
<hr/>	
	TravelAgencyB()
	setMeanOfTransport(String,String,double):void
	setPropertyPrice(String,String,double):void
	getPrice():double

El precio proporcionado por la agencia B se calcularía como:

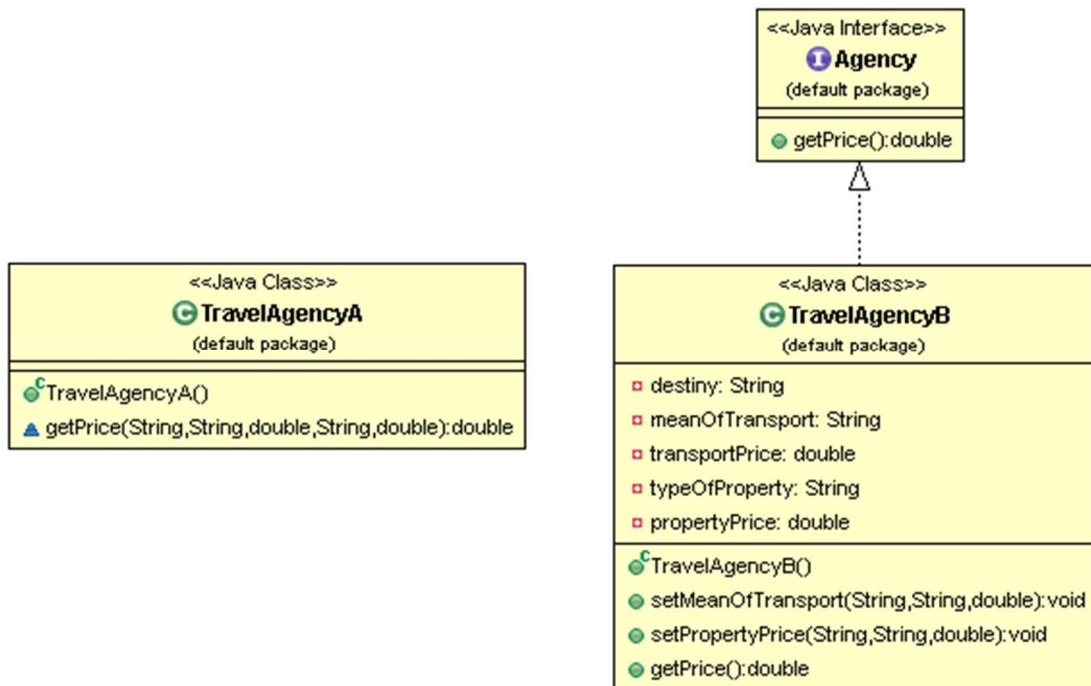
```
//Working out the price for Agency B
TravelAgencyB travelAgencyB = new TravelAgencyB();

//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();
```

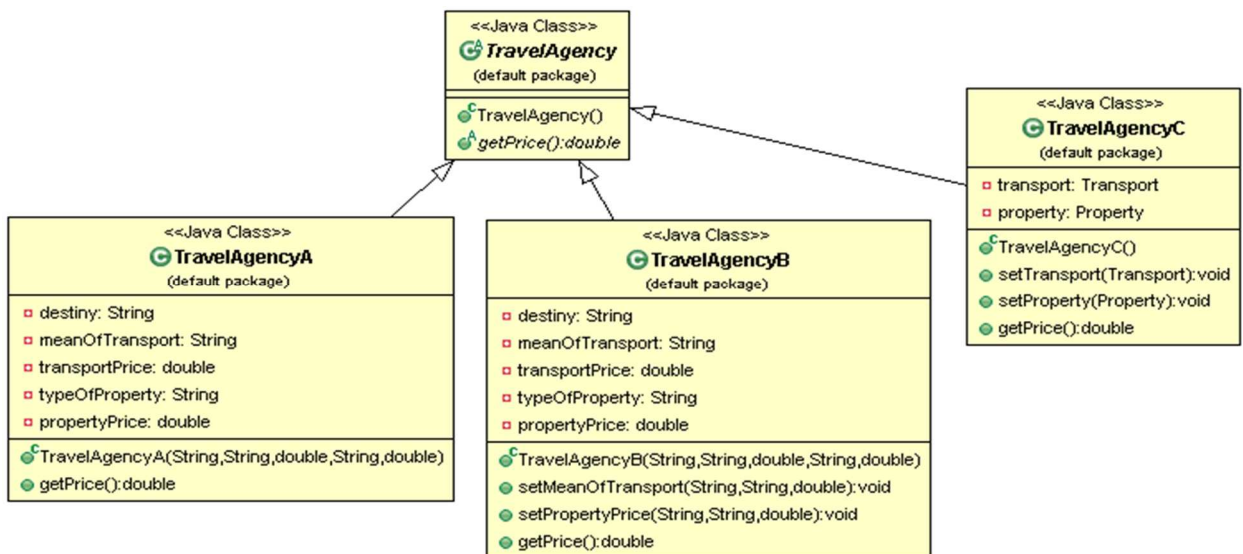
Se desea trabajar de forma indistinta tanto con objetos de TravelAgencyA como de TravelAgencyB por medio de una interfaz denominada Agency con el método getPrice(), el cual tiene el mismo prototipo que el de TravelAgencyB.



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

Define e Implementa las clases necesarias de acuerdo al patrón Adapter de clases y cuál sería la representación de su diagrama de clases correspondiente (proyecto TravelAgencyAdapter). [3 puntos]

2. 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();
```

Define e Implementa las clases necesarias de acuerdo al patrón FactoryMethod y cuál sería la representación de su diagrama de clases correspondiente (proyecto travelAgencyFactoryMethod). [3 puntos]