Sistema de Contratación de Servicios de Seguridad

Programación III - 2023

Leonardo Motylicki - Agustin Perea - Tomas Larsen - Dario Funes

30/04/2023

Facultad de Ingeniería - UNMDP

INTRODUCCIÓN

-El siguiente proyecto tiene como finalidad, modelar y programar un sistema de contratación de servicios de seguridad con las siguientes características:

1. Los servicios ofrecidos son: Monitoreo de alarmas en viviendas o Comercios.
2. Se cuenta con servicios adicionales: Cámaras de seguridad, botones antipánico y móvil de acompañamiento. Estos mismos se añaden al servicio base, por una diferencia en la suma de dinero
3. Cada abonado puede contratar varios servicios, uno para cada uno de los domicilios de los que sea titular.
4. La empresa puede aplicar dos tipos de promociones a sus servicios: Dorada o Platino
5. Los abonados pueden ser personas fisicas o juridicas
6. El pago de los servicios puede ser en: Efectivo, cheque o tarjeta.

-El sistema debe poder realizar la gestión de todos los datos del mundo del problema. El sistema debe poder generar un reporte de todos la facturas emitidas con el detalle correspondiente de cada una. O sea, para cada factura se debe listar cada una de sus contrataciones, con su descripción y valor.

IMPLEMENTACIÓN

Class Sistema Seguridad:

Clase con una única instancia, se encarga de contener y gestionar las facturas y clientes de la empresa mediante arrays.

Interfaz IFactura:

Interfaz encargada de modelar el comportamiento de la clase Factura y su respectivo decorador.

Class Factura:

Se extiende de IFactura. Establece, además, una relación de composición con Sistema, siendo que éste en sus atributos tiene un array de facturas. Es una clase clonable por lo que también implementa Cloneable.

Como su nombre lo indica, representa una factura correspondiente a un cierto abonado en función de los servicios contratados.

En cuanto a métodos, además de su constructor y getters, contiene el método para crear y asignar un pago a sí misma, siendo estas unas de sus responsabilidades. No obstante, la creación dependerá de la forma de pago, por lo cual se utiliza el decorator.

Abstract Class Persona:

Modela mediante herencia el comportamiento que tendrán los tipos de persona. Es de tipo clonable ya que por cascada, al clonar una factura se deberá también clonar diferentes clases más, entre ellas, Persona. Esto es porque mantiene una relación de asociación con la clase Factura. Asimismo tiene una relación de composición con Sistema siendo que Sistema tiene como atributo un ArrayList de esta clase. También tiene una composición con Domicilio, siendo que el Domicilio no existiría sin una Persona. Eliminar la Persona, elimina el Domicilio. Esta composición se traduce en el atributo ArrayList <Domicilio> domicilio.

Persona representa a un abonado del sistema. Es una clase padre de los diferentes tipos de persona que pueden ser clientes del sistema (física o jurídica) ya que mantiene una relación de herencia con esas clases, modelando su comportamiento.

Los métodos que tiene son abstractos y serán sobreescritos en sus clases hija. En particular tiene un método que recibe un ArrayList de doubles llamado recibeDescuento. Este método recibe como parámetro otro ArrayList, pero de contrataciones. Su función es la de crear un Array que guarde en cada componente i un coeficiente. Este coeficiente representa cuánto se ha de cobrar por la contratación i según se trate o no de una contratación de persona jurídica n° 4 o más. Es una forma de resolver cómo aplicar el descuento que reciben las personas jurídicas a partir de su tercer contratación. Este Array tiene tantas componentes como contrataciones tiene la persona.

Class PersonaJurídica:

Clase hija de Persona. Hereda los atributos de ella y sobreescribe sus métodos. No es clonable, lo que representa una situación para la clonación de factura siento que PersonaJuridica es hija de Persona y Persona tiene una relación con Factura, una clase que es clonable.

El método para construir el ArrayList de recibeDescuento en este caso lo que hace es completar el array con 1s en los primeros 3 componentes y luego 0.5 del 3ero en adelante. Lo que significa que a partir del 4to servicio, el mismo saldrá solo la mitad de su valor. Al calcular el monto de la factura se deberá multiplicar el valor de cada contratación por el coeficiente guardado en este array.

Class PersonaFísica:

Clase hija de Persona. Hereda los atributos de ella y sobreescribe sus métodos. Esta clase sí es clonable.

En este caso el método para construir el ArrayList de recibeDescuento, como no va a haber nunca un descuento para las personas físicas, no importa cuántas contrataciones tenga, el coeficiente siempre va a valer 1.

Class Domicilio:

Clase concreta que representa el domicilio de un abonado. Mantiene una relación de composición con Persona y una agregación con Contratacion.

Sobreescribe los métodos de Object equals y hashCode ya que el sistema necesita

ver si un domicilio es igual a otro porque de ser así, habría un error. Cada domicilio debe ser distinto y cada domicilio debe llevar una contratación determinada.

Abstract Class MedioDePagoDecorator:

Clase abstracta para poder implementar métodos del patrón Decorator que modelará de manera dinámica la factura en función de su forma de pago. Esto es porque, según la consigna, si el pago es en efectivo se realiza un descuento del 20%; si el pago es con cheque se realiza un incremento del 10%; y si el pago es con tarjeta de crédito se incrementa un 5%.

Class Efectivo:

Clase hija de la clase abstracta MedioDePagoDecorator. En su constructor, establece el atributo descuento en 0,2 que es el coeficiente a multiplicar cuando, al pagar factura, se elija esta forma de pago.

Class Cheque:

Clase hija de la clase abstracta MedioDePagoDecorator. En su constructor, establece el atributo descuento -0,1. Esto significa que no es un descuento, sino un recargo que se le hace al importe al elegir esta forma de pago.

Class Tarjeta:

Clase hija de la clase abstracta MedioDePagoDecorator. En su constructor, establece el atributo descuento -0,05. Esto significa que no es un descuento, sino un recargo que se le hace al importe al elegir esta forma de pago.

Class MedioDePagoFactory:

Clase que implementa el Patrón Factory para un Medio de Pago.

El método getMedioDePago, recibe como parámetro un string que describe el nombre de algún medio de pago, y retorna un objeto de esa Clase.

Abstract Class Contratacion:

Modela mediante herencia el comportamiento de una Contratación. Es una clase clonable, esto se da como consecuencia de la clonación en cascada de Factura.

Es de tipo autoincremental, controlado por el campo lastId. Además cuenta con un campo Domicilio, que también es único para cada objeto de tipo Contratación.

Cuenta con métodos para calcular el valor de sus servicios, inclusive cuando tiene servicios adicionales y cuando se le aplica una promoción a la misma.

Class MonitoreoComercio:

Clase hija de Contratacion, para el caso particular en que la contratación es para un comercio.

Al llamar al método calculaPromo, por Double Dispatch, le está pidiendo a la subclase de Promoción (Dorada o Platino) correspondiente que utilice su método PromoComercio para calcular el valor del descuento correspondiente para la promo en particular en el monitoreo de comercios.

Class MonitoreoVivienda:

Clase hija de Contratacion, para el caso particular en que la contratación es para una vivienda.

Al llamar al método calculaPromo, por Double Dispatch, le está pidiendo a la subclase de Promoción (Dorada o Platino) correspondiente que utilice su método PromoVivienda para calcular el valor del descuento correspondiente para la promo en particular en el monitoreo de comercios.

Abstract Class ServicioAdicional:

Modela mediante herencia el comportamiento de los servicios adicionales que tendrá una contratación.

Cuenta con un constructor que recibe la cantidad de unidades de este servicio que son agregados a la contratación.

Tiene también el método abstracto obtenerPrecio, que se encargará de calcular el precio de este servicio, dependiendo de quién extienda la clase, y la cantidad del mismo.

Class Camara:

Clase hija de ServicioAdicional, hereda sus atributos y sobreescribe sus métodos.

Implementa el método obtenerPrecio, teniendo en cuenta la cantidad de cámaras que deseen agregarse, y el precio base de las mismas.

Class BotonAntiPanico:

Clase hija de ServicioAdicional, hereda sus atributos y sobreescribe sus métodos.

Implementa el método obtenerPrecio, teniendo en cuenta la cantidad de botones anitpanico que deseen agregarse, y el precio base de los mismos.

Class MovilDeAcompaniamiento:

Clase hija de ServicioAdicional, hereda sus atributos y sobreescribe sus métodos.

Esta clase, además tiene la particularidad de que cuenta con un horario de entrada y otro de salida, en los que el móvil acompaña al cliente.

Implementa el método obtenerPrecio, teniendo en cuenta la cantidad de móviles de acompañamiento que deseen agregarse, y el precio base de los mismos.

Interface Promocion:

Interfaz encargada de modelar el comportamiento que debe tener una promoción. Los métodos que define se usan para implementar el Double Dispatch.

Class Dorada:

Implementa la interfaz Promocion. Dependiendo del tipo de vivienda, la promoción devuelve el importe, al cual se le aplica un descuento.

Class Platino:

Implementa la interfaz Promocion. Dependiendo del tipo de vivienda, la promoción devuelve el importe, al cual se le aplica un descuento.

DIAGRAMA DE CLASES