**Implementación de Patrones de Diseño Creacionales, Estructurales y de Comportamiento**

|  |
| --- |
| Integrantes |
| Edwin Romeo Rivas Díaz |
| Luis Ernesto Figueroa Vásquez |
| Kevin Alexander Aquino Vásquez |
| Luis Alexis Velázquez Godoy |
| Guillermo Alberto Asensio Jiménez |
| Moisés Roberto Hernández Hernández |
| Jonathan Alexander Ramírez Vázquez |

Tabla de contenido

[Solucion tecnica 3](#_Toc182764231)

[Notification 4](#_Toc182764232)

[Observer interface 4](#_Toc182764233)

[Notifier class 4](#_Toc182764234)

[Customer 5](#_Toc182764235)

[Customer class 5](#_Toc182764236)

[Reservation 6](#_Toc182764237)

[Reservation 6](#_Toc182764238)

[ServiceDecorator 7](#_Toc182764239)

[BaseReservation 7](#_Toc182764240)

[BreakFastService 8](#_Toc182764241)

[SpaService 8](#_Toc182764242)

[TransportService 9](#_Toc182764243)

[Room 9](#_Toc182764244)

[Factories 10](#_Toc182764245)

[RoomType 11](#_Toc182764246)

[Room 11](#_Toc182764247)

[RoomFactory 12](#_Toc182764248)

[StandardRoom 12](#_Toc182764249)

[DoubleRoomFactory 13](#_Toc182764250)

[PresidentialRoom 13](#_Toc182764251)

[System 14](#_Toc182764252)

[ReservationSystem 14](#_Toc182764253)

[Clase Main 15](#_Toc182764254)

[Ejecucion de la aplicación 18](#_Toc182764255)

[Creacion de un usuario 18](#_Toc182764256)

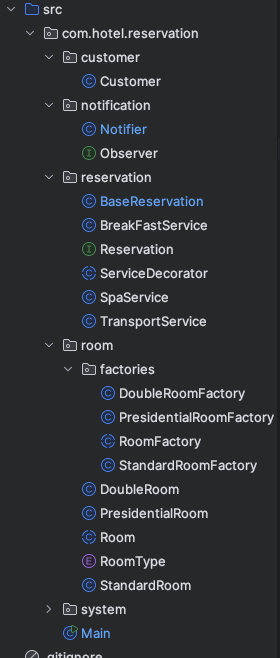
[Agregar Habitacion 18](#_Toc182764257)

[Crear reserva 19](#_Toc182764258)

[Ver reservas 19](#_Toc182764259)

# Solucion tecnica

Como solucion tenica a nuestro proyecto llamado ‘**Hotel Reservation**’ se ha implementado la siguiente arquitectura, la cual consta de:

* customer
  + Customer (Clase)
* notication
  + Notifier (Clase)
  + Observer (interface)
* reservation
  + BaseReservation
  + BreakFastService
  + Reservation
  + ServiceDecorator
  + SpaService
  + TransportService
* room
  + factories
    - DoubleRoomFactory
    - PresidentialRoomFactory
    - RoomFactory
    - StandardRoomFactory
  + DoubleRoom
  + PresidentialRoom
  + Room
  + RoomType
  + StandardRoom
* system
  + ReservationSystem
* Main

## Notification

Paquete creado para implementar el patron observador (observer), el cual se crearan los siguientes archivos:

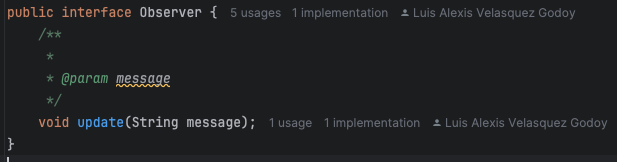
* Notifier (clase)
* Observer (interface)

### Observer interface

Interfaz que contiene el método que se utilizará para notificar nuevas reservas.

Esta interface cuenta con las siguientes propiedades:

* Metodo update

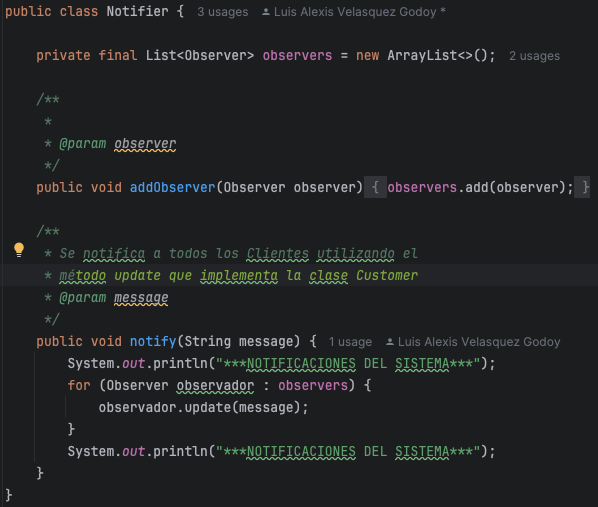


### Notifier class

Clase que almacena todos los observadores (Clientes) que serán notificados al momento de crearse una reserva.

Esta clase cuenta con las siguientes propiedades:

* observers : propiedad que contiene un listado de observadores
* addObserver: metodo que recive por parametro un observador y lo agrega a la propiedad **observers**
* notify: metodo que notificara a todos los Clientes utilizando el método update que implementa la clase Customer



## Customer

Paquete creado para implementar la clase customer y su funcionalidad.

### Customer class

Clase que define el comportamiento y atributos de los clientes. Se implementa Observer para que los clientes puedan ser notificados siempre que hagan una reserva

Esta clase cuenta con las siguientes propiedades:

* id: propiedad de tipo int con sus metodos de acceso
* name: prpiedad de tipo String con sus metodos de acceso
* update: metodo que recive por parametro la propiedad message y retorna el valor del mensaje. Método que se utiliza Observer para enviar mensajes.

Ademas, en dicha clase se sobre escribe el metodo toString, esto según a la necesidad del desarrollo.



## Reservation

Paquete creado para implementar la logicade las reservas, el cual se crearan los siguientes archivos:

* Reservation (interface)
* ServiceDecorator (class)
* BaseReservation (class)
* BreakFastService (class)
* SpaService (class)
* TransportService (class)

### Reservation

Interface que define el comportamiento básico de una reserva

Esta interface cuenta con las siguientes propiedades:

* calculatePrice: metodo que retorna un valor de tipo double
* getSummary: propiedad de tipo String

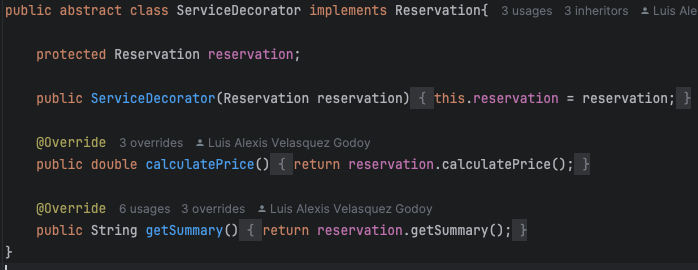


### ServiceDecorator

Clase decoradora que permite agregar un servicio extra a la reserva.

Esta clase cuenta con las siguientes propiedades:

* reservation: propiedad de tipo Reservation
* calculatePrice: propiedad de la implementacion de la interface Reservation
* getSummary : propiedad de la implementacion de la interface Reservation

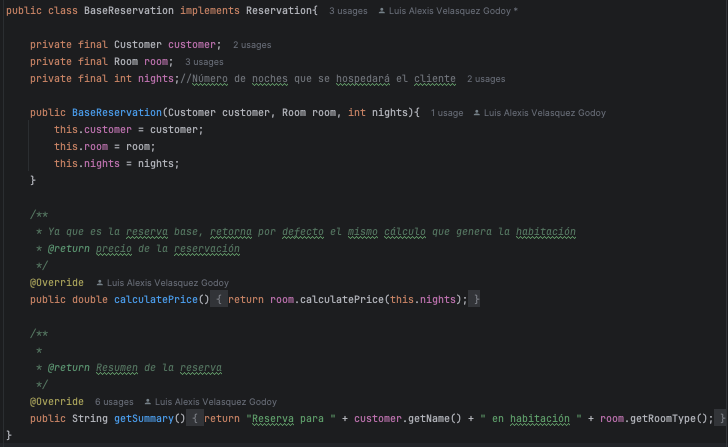


### BaseReservation

Clase base de una reserva. Implementa la interfaz Reservation

Esta clase cuenta con las siguientes propiedades:

* customer: propiedad de tipo Customer
* room: propiedad de tipo Room
* nights: propiedad de tipo int
* calculatePrice: propiedad de la implementacion de la interface Reservation. Ya que es la reserva base, retorna por defecto el mismo cálculo que genera la habitación.
* getSummary : propiedad de la implementacion de la interface Reservation. Resumen de la reserva



### BreakFastService

Decorador que agregar el servicio de desayuno a la reserva.

Esta clase cuenta con las siguientes propiedades:

* calculatePrice: propiedad de la implementacion de la interface Reservation. Se hace un cargo de 20 al solicitar desayuno.
* getSummary : propiedad de la implementacion de la interface Reservation.

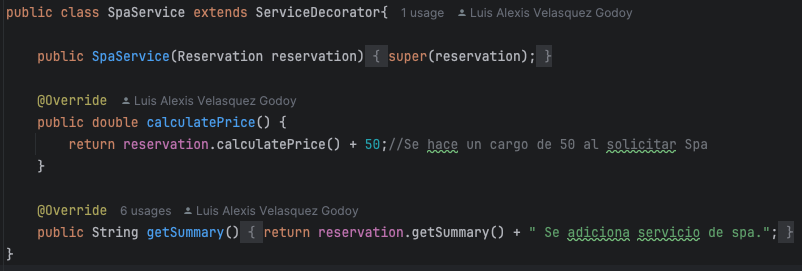


### SpaService

Decorador que agregar el servicio de SPA a la reserva.

Esta clase cuenta con las siguientes propiedades:

* calculatePrice: propiedad de la implementacion de la interface Reservation. Se hace un cargo de 50 al solicitar SPA.
* getSummary : propiedad de la implementacion de la interface Reservation.



### TransportService

Decorador que agregar el servicio de transporte a la reserva.

Esta clase cuenta con las siguientes propiedades:

* calculatePrice: propiedad de la implementacion de la interface Reservation. Se hace un cargo de 100 al solicitar transporte.
* getSummary : propiedad de la implementacion de la interface Reservation.



## Room

Paquete creado para implementar la logicade los cuartos (room), el cual se crearan los siguientes archivos:

* factories
  + RoomFactory (class)
  + DoubleRoomFactory (class)
  + PresidentialRoomFactory (class)
  + StandardRoomFactory (class)
* DoubleRoom (class)
* PresidentialRoom (class)
* Room (class)
* RoomType (enum)
* StandardRoom (class)

### Factories

#### RoomFactory

Clase abstracta que define el comportamiento de las factories de habitaciones.

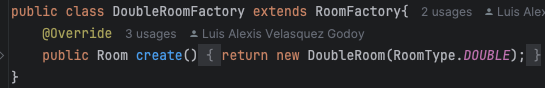
Esta clase abstracta cuenta con las siguientes propiedades:

* Create: Retorna un objeto de tipo Room o que extienda de la clase Room (StandardRoom, DoubleRoom, PresidentialRoom)



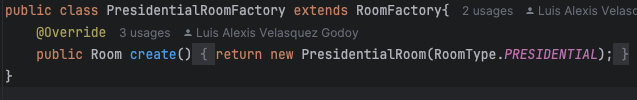
#### DoubleRoomFactory

Factoría que retorna objetos de tipo DoubleRoom



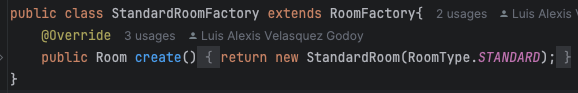
#### PresidentialRoomFactory

Factoría que retorna objetos de tipo PresidentialRoom



#### StandardRoomFactory

Factoría que retorna objetos de tipo StandardRoom

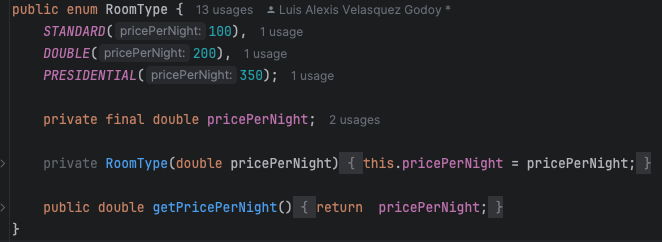


### RoomType

Enum que define el tipo de habitación que ofrece el hotel y costo por noche.

Este enum cuenta con las siguientes propiedades:

* pricePerNight: propiedad de tipo double con sus metodo de acceso (get)

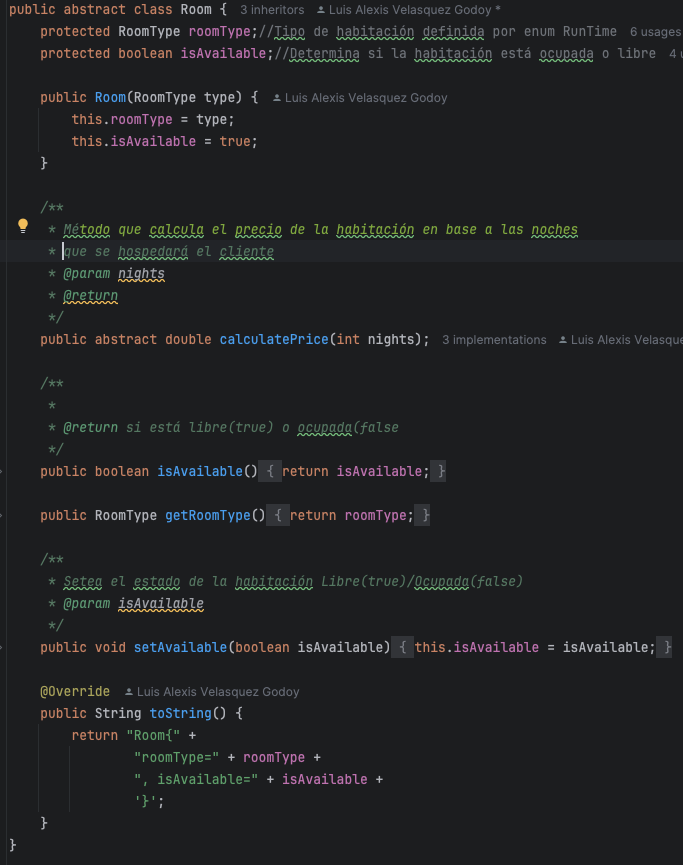


### Room

Clase abstracta que define el comportamiento y atributos de una habitación.

Esta clase cuenta con las siguientes propiedades:

* roomType: propiedad de tipo RoomType con sus metodo de acceso (get)
* isAvailable: propiedad de tipo boolean con sus metodo de acceso (get)
* calculatePrice: propiedad que recive por parametro la propiedad nights de tipo int, y retorna un double.
* toString: Sobre escritura del metodo troString

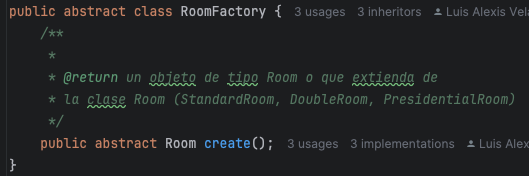


### RoomFactory

Clase abstracta que define el comportamiento de las factories de habitaciones.

Esta clase cuenta con las siguientes propiedades:

* create: metodo que retorna una propiedad de tipo Room. Retorna un objeto de tipo Room o que extienda de la clase Room (StandardRoom, DoubleRoom, PresidentialRoom)

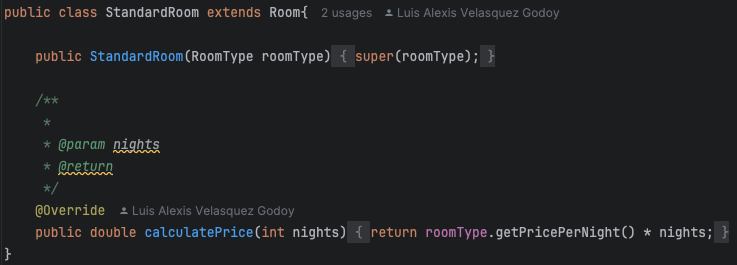


### StandardRoom

Clase que determina el comportamiento de una habitación tipo Standard. Se extiende de la clase Room

Esta clase cuenta con las siguientes propiedades:

* calculatePrice: metodo que recive el parametro nights de tipo int y retorna el precio de total de las noches de reserva por el precio de la habitacion.

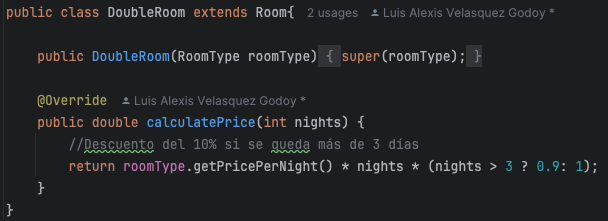


### DoubleRoomFactory

Clase que determina el comportamiento de una habitación tipo doble. Se extiende de la clase Room

Esta clase cuenta con las siguientes propiedades:

* calculatePrice: metodo que recive el parametro nights de tipo int y retorna el precio de total de las noches de reserva por el precio de la habitacion. Si se quede mas de 3 dias se le dara un descuento del 10%

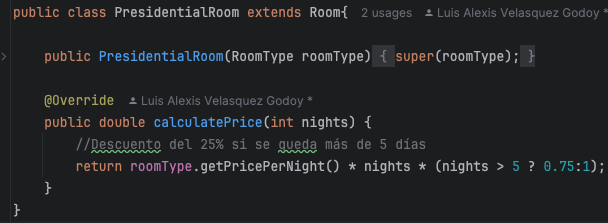


### PresidentialRoom

Clase que determina el comportamiento de una habitación tipo presidencial. Se extiende de la clase Room

Esta clase cuenta con las siguientes propiedades:

* calculatePrice: metodo que recive el parametro nights de tipo int y retorna el precio de total de las noches de reserva por el precio de la habitacion. Si se quede mas de 5 dias se le dara un descuento del 25%



## System

Paquete creado para implementar la logica del almacenamiento de las reservas. En dicho paquete se implementa el patron singleton. Ademas, todo esta implementado en la clase ReservationSystem

### ReservationSystem

Clase general que determina el comportameinto del sistema. Almacena todos los clientes, Cuartos y Reservas. Se utiliza el patrón de diseño Singleton para tener una única instancia de esta clase general.

Esta clase cuenta con las siguientes propiedades:

* instance: propiedad que almacena la instancia de la clase
* customers: propiedad que almacena el listado clientes
* rooms: propiedad que almacena el listado de cuartos
* reservations: propiedad que almacena el listado de reservas
* customerId: lmacena el último id de cliente
* getInstancia: retorna una instancia única de clase ReservationSystem.
* getCustomerId: Retorna el último ID de cliente que se ha utilizado y posteriormente se suma 1 ya que ese será el nuevo ID que se asigne a un nuevo cliente que se cree.
* addCustomer: Se registra el cliente en el sistema.
* addRoom: Se registra la habitación en el sistema
* createReservation: Se registra la reserva en el sistema
* getCustomerById: retorna el cliente con ID @param id. Si no existe cliente con el ID que proporciona el usuario, retorna null.
* getAllCustomers: retorna una lista de todos los clientes registrados en el sistema.
* getAllRooms: retorna una lista de todos las habitaciones registradas en el sistema.
* getAllReservation: retorna una lista de todas las reservaciones registradas en el sistema.
* getAllAvailableRooms: retorna una lista de todas las habitaciones disponibles registradas en el sistema.



## Clase Main

La clase main contiene la funcionalidad de recibir los valores al usuario que esta corriendo el aplicativo.

Como punto de partida, se obtiene la instancia de la clase Notifier, luego se muestra el menu con las siguientes opciones:

1. Registrar Cliente
   1. Se ingresaran los datos del cliente
2. Agregar Habitación
   1. Se solicitara que ingrese el tipo de habitacion
3. Crear Reserva
   1. Se solicitara el id del cliente para anexar la reserva.
   2. Se verifica las habitaciones disponibles
   3. Se muestra el listado de habitaciones disponibles
   4. Se selecciona una habitacion
   5. Se solicita el numero de noches que se requieren para la reserva
   6. Se le muestra el menu de opciones adicionales a la habitacion
   7. Se ingresa la reserva
   8. Se notifica la creacion de la reserva
4. Ver Reservas
   1. Se muestra el listado de las reservas creadas
5. Salir
   1. Finaliza la ejecucion de la aplicacion

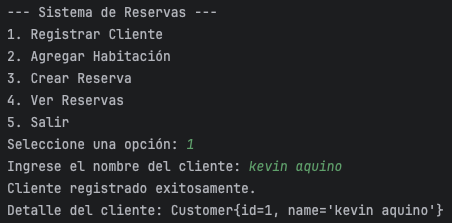




## Ejecucion de la aplicación

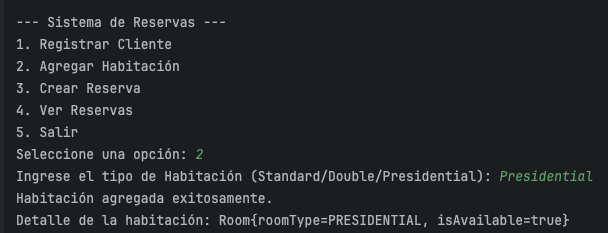
### Creacion de un usuario

Al usuario dentro del aplicativo, se le solicitara registrar a un nuevo cliente, dicha opcion es la numero uno, se le solicitara el nombre, y si todo esta correctamente, se ingresara el registro, y se podra observar el identificador del cliente, que en este caso es 1.



### Agregar Habitacion

Al usuario dentro del aplicativo, se le solicitara registrar a una nueva habitacion, dicha opcion es la numero dos, se le solicitara el nombre, dicho nombre debe de ser una de las opciones que aparece en el enunciado (Standard/Double/Presidential), y si todo esta correctamente, se ingresara el registro.



### Crear reserva

Al usuario dentro del aplicativo, se le solicitara registrar a una nueva reseva, dicha opcion es la numero tres, se le solicitara el id del cliente, en este caso es 1, luego se le solicitara la habitacion, el numero de noches de la reserva, y por ultimo si tiene algun extra para agregar a la reserva, y si todo esta correctamente, se ingresara el registro, y se podra observar el resumen de la reserva.



### Ver reservas

Al usuario dentro del aplicativo, podra observar el listado de reservas, el cual podra ver el detalle de cada reserva que hay.

