PRACTICA FINAL POO

Detalles de la implementación

DIEGO JIMENO DE LA CALLE
DIEGO BORJA FARINANGO

Implementación

Atributos:

En la clase Hotel, tenemos un ArrayList de Habitacion, llamado lista_habitaciones, donde almacenaremos las habitaciones pertenecientes al hotel, debido a los métodos internos del ArrayList, elegimos este formato para facilitar el trabajo. También hay un hashmap, de integer a string, que nos facilita obtener los datos String del tipo para su uso posteior.

Por ultimo tenemos un objeto del tipo LeerObjetos, leer, que utilizaremos para serializar los objetos.

En Habitación, tenemos los atributos propios de las habitaciones, añadiendo el atributo tipo que nos facilita obtener el tipo de la habitación. En cada tipo de habitación, damos ciertos valores por defecto a la habitación, como son el precio por día y de camas supletorias, el tipo y en el caso de individual, suite y familiar si comparten o no baño.

En la clase Reserva, tenemos como atributos los días inicial y final de la reserva, el dni del titular, si van a usar camas supletorias, y si la reserva está pendiente de cobro. Además, con el atributo vip podemos indicar si la reserva es vip o no en caso de reservar una suite.

Métodos:

Dentro de la clase hotel, que sigue el patrón Singleton, encontramos todos los métodos de operación que utilizaremos en el main, Gestion_Hotel:

- Registrar_Habitacion():

En este método agregamos habitaciones a la lista de habitaciones seleccionando sus atributos en función del tipo de habitación que sea.

Para esto utilizamos el método agregar_Habitacion, que nos permite optimizar código. Y utiliza métodos de la clase Habitacion.

Consideramos que el hotel tiene 6 pisos, con 20 habitaciones en cada uno, siguiendo un modelo CDU, siendo C (centenas) el número de piso, y DU (decenas y unidades) el número de habitación dentro de cada planta.

- Modificar_Habitacion():

Este método permite modificar datos de una habitación encontrada a través de su número, para esto utilizamos el método buscar_Habitacion().

A través del método actualizar_Habitacion, teniendo en cuenta el tipo de habitación, modificamos los valores de la habitación, pudiendo cambiar valores por defecto como el precio por día de la habitación o el precio de las camas supletorias. Consideramos que el número de habitación no cambia porque sería otra habitación.

- Listado_Habitaciones ():

Mediante un bucle nos devuelve un listado de todas las habitaciones registradas dentro del hotel, haciendo uso del método mostrar_habitación() al que le pasamos un objeto de tipo Habitacion como parámetro y nos muestra los datos de esa habitación en concreto.

- Info_Habitación():

Este método nos muestra la información de una única habitación con la ayuda del método mostrar_habitacion() antes descrito.

- Eliminar_Habitacion():

Este método elimina una habitación de la lista de habitaciones de Hotel con la ayuda de buscar_Habitacion(), que busca la habitación a eliminar.

- Guardar_Configuracion():

Este método usa el paquete serializable para guardar los objetos de forma permanente en un archivo permanente llamado “base_datos_hotel”, dentro de la carpeta del proyecto.

Nota: Los métodos de operación tanto de habitaciones como reservas están en la clase Hotel pero los métodos relacionados con reserva están dentro de la clase Habitación.

- Registrar_Reservas():

En este método creamos nuevas reservas para una habitación existente, comprobando que esté libre entre las determinadas fechas que se quiere realizar la reserva.

Para esto, utilizamos el método de la clase Habitacion generar_reserva() al que le pasamos como parámetro las fechas de inicio y de final. generar_resesrva(), utiliza un método de la clase Habitación que es obtener_datos_reserva que nos sirve para optimizar código y pasándole una reserva, le asigna valores a sus atributos y nos devuelve el objeto con los valores ya añadidos.

Al generar una reserva, la consideramos no vip por defecto, pero se puede cambiar si está en una habitación de tipo suite en modificar reserva.

- Modificar_reserva():

A través del método buscar_Habitacion, buscamos la habitación por su número, y dentro de habitación, usamos el método de Habitacion actualizar_reserva, que utiliza un método de la misma clase buscar_reserva() para encontrar la reserva según el día de inicio. Después, se muestran los datos de la reserva con el método obtener_datos_reserva() al que se le pasa como parámetro un objeto de tipo reserva. Luego utilizamos el método obtener_datos_reserva() que pide y da nuevos valores a la reserva.

Consideramos que las fechas de inicio y final no se pueden cambiar, porque sería como agregar otra reserva.

- listado_reservas():

Con el método get_Listado_reservas() de la clase Habitacion, mostramos la información de todas las reservas de todas las habitaciones, usando el método de la misma clase mostrar_info_reseserva().

- info_Reserva():

Con el método buscar_reserva() de la clase Habitacion, buscamos la reserva por la fecha de entrada y utilizamos mostrar_info_reserva() para obtener sus datos.

- Eliminar_Reserva():

Con los métodos buscar_Habitacion() y buscar_reserva(), encontramos la reserva, y la borramos con el método de la clase Habitacion borrar_reserva().

- generar_factura():

Con el método generar_factura() de la clase Habitacion, comprobando que la reserva no esté pendiente de cobro y en función del tipo de la habitación, el número de días, las camas supletorias y si la reserva es vip, generamos el precio total e indicamos los datos de la reserva.

Después de esto, no borramos la reserva para poder mantenerla a modo de historial.

Paquetes:

Entrada: MyInput.

HotelExcepciones: Usado para crear excepciones propias, implementadas en toda la aplicación.

Validaciones: Para validar los dni y las fechas introducidas, utilizamos los métodos del paquete validaciones, en la clase Validar, los métodos validar_dni() y validar_fecha(). Estos métodos los utilizamos en otro paquete ya que su cohesión con el resto de clases en baja y para poder utilizarlos en las distintas clases sin repetir código.

Serializable: Usado tanto para leer y escribir objetos en un fichero.