**Actividad**

**Programación Orientada a Objetos**

**Unidad 2**

**Estudiante:**

**Alvaro Jesús Perez Menendez**

**Código:**

**7502420053**

**Universidad de Cartagena**

**Facultad de Ingeniería**

**Ingeniería de Software**

**Introducción**

**Se hará la documentación de la actividad que se ha dado en la propuesta del tutor, para el ejercicio 36 – Agencia de Viajes, donde se habrán hecho varios cambios, principalmente porque en la actividad numero uno se hizo sin tanto conocimiento y se hizo utilización de varios materiales y videos para poder hacerlo.**

**En este nuevo trabajo, se hizo completamente a pulso la parte de los diagramas de clases y su integración a código a NetBeans, lo considero un gran logro, porque se hizo sin ayuda de algún material, simplemente con el conocimiento de haber practicado varias veces con otros trabajos.**

**En la parte de los cruds su implementación es clara, mas aun no se capta completamente la estructura de como va creado, valga la redundancia, el crud, para poder utilizarlo en el main.**

**Habiendo dicho esto, se hará un breve resumen de las clases, ya que, la falta de conocimiento y la falta de leer, no me dejaron saber como iba estructurado la Actividad 1, y no explique cómo se debía.**

**Para la actividad 2, se hablara de como realice los cruds y sus debidas pruebas para un buen funcionamiento.**

**Clases**

**Sucursal.java**

* **Función: Esta clase se encargara de atender a los turistas para tener que se vengan a registrar para poner asignarle una o varias reservas, ya sea para un hospedaje en hotel o un vuelo.**
* **Atributos:**
  + **String - código: Las sucursales tendrán un código para poder identificarlo en el sistema.**
  + **String - nombreSucursal: Esto en si es solo un relleno para darle mas simulación al sistema, ya sea “Sede Cartagena” o “Sede Barranquilla”.**
  + **String - ciudad: Indica en que ciudad se encuentra la Sede.**
  + **String - dirección: Indica la Dirección que se encuentra la Sede.**
  + **String - Teléfono: Un numero de contacto para los turistas.**
* **Constructor:** 
  + **Se asignan los atributos que se necesitan para cuando se quiera crear un objeto de esta clase**
* **Métodos:**
  + **Get(para todos los atributos): se hizo get a todos los atributos, para que desde otras clases por si a futuro se necesita, se pueda usar los atributos de esta clase.**
  + **Set(para todos los atributos): Se hace set para todos los atributos dado que al momento de utilizar los archivos crud en el main, es importante el poder editar cualquier información que se guarden en sus atributos.**
  + **mostrarInformacion: Este método mostrara los datos que se guarden en los atributos a la hora de asignarlos en el main.**
  + **realizarReserva: Se hizo dos métodos de realizarReserva, tanto para vuelo como para hotel, cuando el turista ingrese a la sucursal y reserve, este método lo que hará es crear desde las clases TuristaRerservaHotel y TuristaReservaVuelo, un objeto donde muestre los atributos de Turista, y los atributos complementarios de las clases antes mencionadas.**

**Hotel.java**

* **Función: esta clase se encargara de crear objetos, ósea hoteles, para al momento de asignarlos en las reservas de los turistas.**
* **Atributos**
  + **String - código: Los hoteles tendrá un código para identificarlos en el sistema.**
  + **String - nombreHotel: un nombre que se asigna al hotel.**
  + **String - dirección: ubicación del hotel.**
  + **String - ciudad: en que ciudad se encuentra el hotel.**
  + **String - teléfono: Contacto para comunicarse con Servicio al cliente del hotel.**
  + **Int - plazasDisponibles: Este atributo indica cuantas plazas hay disponibles en el hotel.**
* **Constructor:**
  + **Se asignan los atributos que se necesitan para cuando se quiera crear un objeto de esta clase**
* **Metodos:**
  + **Get(para todos los atributos): se hizo get a todos los atributos, para que desde otras clases por si a futuro se necesita, se pueda usar los atributos de esta clase.**
  + **Set(para todos los atributos): Se hace set para todos los atributos dado que al momento de utilizar los archivos crud en el main, es importante el poder editar cualquier información que se guarden en sus atributos.**
  + **mostrarInformacion: Este método mostrara los datos que se guarden en los atributos a la hora de asignarlos en el main.**

**Vuelo.java**

* **Funcion: Esta clase se encarga de crear objetos, ósea vuelos, los cuales van a ser asignados a los turistas cuando reserven en la sucursal.**
* **Atributos:**
  + **String - numeroVuelo: Es el código como tal que identifica el vuelo.**
  + **String - fechaHoraVuelo: Dirá la fecha que se asigna el vuelo.**
  + **String - origen: Desde donde se está tomando el vuelo**
  + **String - destino: A donde se dirige el vuelo.**
  + **String - plazasDisponibles: Plazas disponibles en total**
  + **String - plazasDisponiblesTurista: Plazas que utilizan los turistas, como el plan normal, lo que quede será primera clase (EN MOSTRAR INFORMACION SE HACE UN ARREGLO PARA RESTAR LOS DOS ATRIBUTOS DE PLAZAS PARA MOSTRAR UNA INFORMACION DE CUANTAS PLAZAS DE PRIMERA CLASE HAY)**
* **Constructor**
  + **Se asignan los atributos que se necesitan para cuando se quiera crear un objeto de esta clase**
* **Métodos:**
  + **Get(para todos los atributos): se hizo get a todos los atributos, para que desde otras clases por si a futuro se necesita, se pueda usar los atributos de esta clase.**
  + **Set(para todos los atributos): Se hace set para todos los atributos dado que al momento de utilizar los archivos crud en el main, es importante el poder editar cualquier información que se guarden en sus atributos.**
  + **mostrarInformacion: Este método mostrara los datos que se guarden en los atributos a la hora de asignarlos en el main.**
  + **En mostrar informacion hacemos un pequeño arreglo por decisión propia de estética donde, se crea una variable llamada PlazasPrimeraClase, esta se usa el entero de this.plazasDisponibles y lo resta del entero this.plazasDisponiblesTurista, hacemos un printf donde se pone un texto, y se utiliza la variable creada.**

**Turista.java**

* **Función: Esta clase se encargara de crear Turistas lo cuales son los que van a ir a las sucursales, OJO, aun no tienen la reserva, simplemente su informacion.**
* **Atributos:**
  + **String – codigo: codigo con el cual se identifican en el sistema, ya que estos iran a la sucursal.**
  + **String – nombreCompleto: el nombre completo del turista**
  + **String – id: en este atributo iran datos como cedula, documento de intentidad o algo que les identifique.**
  + **String – dirección: Direccion de la turista.**
  + **String – teléfono: teléfono del turista.**
  + **Sucursal – sucursal: lo utilizaremos para extraer un atributo de la Clase sucursal, en este caso el codigo, ya que necesitamos saber en que sucursal fue atendido.**
* **Constructor:**
  + **Se asignan los atributos que se necesitan para cuando se quiera crear un objeto de esta clase**
* **Métodos:**
  + **Get(para todos los atributos): se hizo get a todos los atributos, para que desde otras clases por si a futuro se necesita, se pueda usar los atributos de esta clase.**
  + **Set(para todos los atributos): Se hace set para todos los atributos dado que al momento de utilizar los archivos crud en el main, es importante el poder editar cualquier información que se guarden en sus atributos.**
  + **mostrarInformacion: Este método mostrara los datos que se guarden en los atributos a la hora de asignarlos en el main.**
  + **Para mostrar el código de la sucursal en mostrarInformacion, se utiliza sucursal.getCodigo, esto lo que hace es que con el get que aplicamos en Sucursal, esta clase pueda usarla.**

**TipoPension.java**

* **Función: Esta clase es un tipo Enum, en este caso mostrara opciones para usarlos con if en la clase donde se guarde la reserva de Hotel.**
* **Contenido:**
  + **PENSION\_COMPLETA**
  + **MEDIA\_PENSION**
  + **SIN\_PENSION**

**TipoPlaza.java**

* **Función: Esta clase es un tipo Enum, en este caso mostrara opciones para usarlos con if en la clase donde se guarde la reserva de Vuelo.**
* **Contenido:**
  + **PRIMERA\_CLASE**
  + **TURISTA**

**TuristaReservaHotel .java**

* **Función: Esta clase fue creada para poder utilizar complementos necesarios a la hora de que un turista tenga una reserva en un hotel, lo que nos mostrara información adicional.**
* **Atributos**
  + **Turista turista: Usara atributos de la clase turista.**
  + **TipoPension: Usamos la clase tipo Enum para poder elegir el tipo de pension que tiene el turista a la hora de reservar el hotel.**
  + **Hotel hotel: Usara atributos de la clase Hotel**
  + **Date fechaLlegada: Como lo exige la actividad, necesitamos un atributo que nos diga la fecha de llegada al hotel.**
  + **Date fechaSalida: Como lo exige la actividad, necesitamos un atributo que nos diga la fecha de salida del hotel.**
* **Constructor:**
  + **Se utilizará el constructor usando los atributos de la clase turista, que sería toda su información, el tipo de pensión, en que hotel se hospeda, e estancia en el hotel.**
* **Métodos:**
  + **Get(para todos los atributos): se hizo get a todos los atributos, para que desde otras clases por si a futuro se necesita, se pueda usar los atributos de esta clase.**
  + **Set(para todos los atributos): Se hace set para todos los atributos dado que al momento de utilizar los archivos crud en el main, es importante el poder editar cualquier información que se guarden en sus atributos.**
  + **MostrarInformacion: Acá como decía, se utilizará toda la información del turista, por lo cual se utilizará el get para llamarlas y los datos complementarios que se tienen que saber de la reserva, como el hotel reservado, fecha de llegada y salida.**
  + **Aparte de esto, utilizamos IF para asignarle al turista con la reserva que tipo de pensión tiene.**

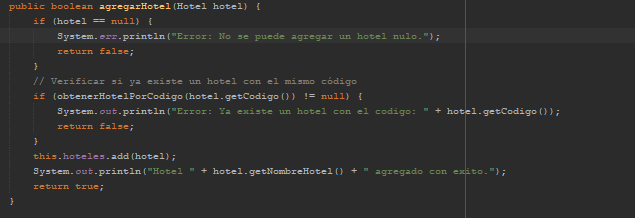
**TuristaReservaVuelo.java**

* **Función: Esta clase fue creada para poder utilizar complementos necesarios a la hora de que un turista tenga una reserva en un vuelo, lo que nos mostrara información adicional.**
* **Atributos**
  + **Turista turista: Usara atributos de la clase turista.**
  + **TipoPlaza tipoPlaza: Usara la clase enum para usarlo en if y poder escoger opciones.**
  + **Vuelo vuelo: Usara atributos de la clase Vuelo.**
* **Constructor:**
  + **Se utilizará el constructor usando los atributos de la clase turista, que sería toda su información, el tipo de pensión, en que hotel se hospeda, e estancia en el hotel.**
* **Metodos:** 
  + **Get(para todos los atributos): se hizo get a todos los atributos, para que desde otras clases por si a futuro se necesita, se pueda usar los atributos de esta clase.**
  + **Set(para todos los atributos): Se hace set para todos los atributos dado que al momento de utilizar los archivos crud en el main, es importante el poder editar cualquier información que se guarden en sus atributos.**
  + **MostrarInformacion: Aca, se utilizara toda la información del turista, por lo cual se utilizara el get para llamarlas, se utiliza la clase enum, para mostrar la opción que se le asigne y hacer get a los atributos de vuelo.**
  + **Utilizamos IF, para decidir si el turista con la reserva tiene Clase Turista, o Primera clase.**

**Cruds**

**HotelCrud.Java**

* **Agregar**

****

**En este codigo se hace la implementación del método agregarHotel, que tiene como objetivo añadir un objeto Hotel.  
  
Se utiliza el método “public boolean agregarHotel(Hotel hotel)”**

**Esto indica que el método devuelve un valor booleano, ya sea verdadero o flaco, para indicar si la operación se hizo o no.**

**agregarHotel, es el nombre del método que utilizaremos para utilizar el agregar**

**(Hotel hotel), nos dice un parámetro de entrada, donde se espera un objeto tipo Hotel, cuando se utilce el método.**

**Tenemos un if (onbtenerHotelPorCodigo(hotel.getCodigo()) ¡= null:**

**Hotel.getCodigo, es un método que hace que saquemos los datos del atritubo codigo de la clase Hotel, para que devuelva ese valor.**

**ObtenerHotelPorCodigo(..) es un método que que busca en los hoteles guardados en el crud, el codugo proporcionado, si lo encuentra, devolverá el objeto hotel, si no, devolverea un null  
!= null, si el resultado de obtenerHotelporCodigo no es null, entonces esto significa que ya se creo un objeto con ese mismo codigo**

**Por ultimo el return false, es el método donde termina y devuelve false, ya que no se permite agregar hoteles duplicados por codigo**

**Ahora si nuestro método tiene éxito, entonces se agragara, tienendo un print donde nos diga que el hotel a sido agregado, obteniendo de ella un hotel.getNombreHotel(), que nos dira que el hotel x fue agregado.**

**Y se utiliza el return true, para decir que el método fue verdadero, ósea que el hotel fue creado.**

* **Leer**

**Texto

El contenido generado por IA puede ser incorrecto.**

**Tenemos un método llamado obtenerTodosLosHoteles(), este método se encarga d devolver una lista de todos los hoteles que están almacenados en el crud**

**Empezamos con un public List<Hotel> obtenerTodosLosHoteles():**

**El list<Hotel> nos indica que nos devuelve una lista de objetos del tipo Hotel.**

**Cuando utilizamos el return new ArrayList<>(this.hoteles);**

**Quiere decir que this.hoteles, es la informacion guardada en el crud,**

**New ArrayList<>(), esto nos permite iniciar una lista con todos los objetos que hay en this.hoteles**

**Ahora cuando utilizamos public Hotel obtenerHotelPorCodigo(String codigo) {}**

**Es que como se dice, se esta utilizando el método, y este recibe un parámetro String llamado codigo, que es el codigo del hotel que hay que buscar.**

**For (Hotel h: | this.hoteles), es bucle donde itera sobre cada objeto hotel presente que esta guardado en el crud**

**Dentro de cada bulcie, el if(h.getCodigo().equals(codigo)), se obtiene su codigo h.getCodigo)**

**El .equals(codigo) es crucial para comparar objetos String en java**

**El return, nos dice que si enceuntra un hotel cuyo codigo coincide con el codigo buscando, el método devuelve inmediatamente ese hotel**

**Y si no, tenemos un return null, el bucle termina, es decir que en toda la búsqueda del crud, no se encontró el codigo y devuelve un null**

* **Actualizar o Editar**

**Texto, Escala de tiempo

El contenido generado por IA puede ser incorrecto.**

**Aquí utilizamos public boolean actualizarHotel(Hotel hotelActualizado)**

**Asi como habíamos dicho, el boolean nos va a tirar un true o un flase, tenemos el nombre del método con el cual lo llamaeremos y diciendonos que de la Clase Hotel, tendremos una variable llamada hotelActualizado, con la informacion editada o actualizada.**

**Tenemos un if (notelActualizado == null): que nos indica que si no se proporciona un objeto valido, este nos dará null**

**Y nos imprime con un mensaje en print que nos dira que este objeto es nulo.**

**Por utilmo el return false, que nos indicara que la operación fallo**

**Ahora, con el for (int i = 0; i < thisl.hoteles.size(); i++) se hará un bucle traidiciona que itera atraves de this.hoteles, utilizando el índice i, asumiendo que this.hoteles es una lista donde hay que buscar**

**Tenemos el if this.hoteles.get(i).getCodigo().equals(hotelActualizado.getCodigo()))**

**Que dentro del bucle en cada iteración, se obtiene un hotel del la lista (this.hoteles.get(i).**

**Entonces cuando se encuentra el codigo, this.hoteles.set(.i, actualizado); nos remplaza el objeto existente y se imprimirá con un println, que el hotel con codigo xxx que se pide con un get de la variable actualizada, ha sido actualizada con éxito**

**Osea que cerraríamos con un return true, señalando que la operación fue un éxito.**

* **Eliminar**
* **Texto

  El contenido generado por IA puede ser incorrecto.**

**Llegamos al utimo método que es eliminarHotel(String codigo)**

**Este método se encarga de buscar y eliminar un hotel guardada en la crud, por medio de su codigo. Aca se utiliza el iterator, ya que me parecio una forma fácil de usar para eliminar elementos**

**Tenemos el**

**Public boolean eliminarHotel(String codigo):**

**Donde sabemos que el boolean nos devuelve un true o un false**

**El nombre del método que es eliminarHotel**

**Y que se eliminiara con un String codigo**

**Utilizamos Iterator<Hotel> iterator = this.hoteles.iterator()**

**Para usar el iterator en lo que se guarda en this.hoteles, esto nos hace eliminar elementos de lo que esta guardado mientra se esta iterando en ella,**

**Utilizamos while (iterator.hasNext()) para que le diga a iterarot que mientras haiga mas elementos por ver, siga revisando.**

**Hotel h = iterator.next(); significa que se obtiene el siguiene elemento Hotel de la informacion guradada en la crud, atraves del iterator y se asigna a la variable h**

**Se compara el codigo actual con getCodigo con el que se desea eliminar y utilzamos el .equals para comprar String**

**Por ultimo deceimos que iterator.remove nos elimine el elemento , porque si los códigos coinciden, este actuara y marcara un print diciendo que el hotel con el codigo x se elimino con éxito, terminando con un true, de caso contrario nos marcara un false, diciendo que no se encontró ningún codigo para eliminar.**

**Texto

El contenido generado por IA puede ser incorrecto.**

**Texto

El contenido generado por IA puede ser incorrecto.**

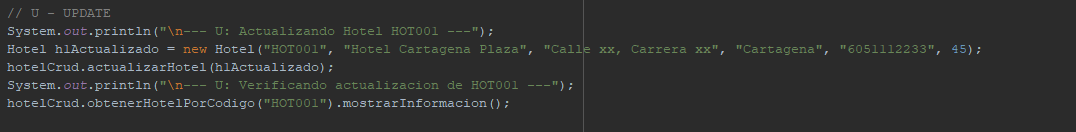
**Podemos ver el funcionamiento de los métodos que se utilizan el el crud.**

**Texto

El contenido generado por IA puede ser incorrecto.**

**Texto

El contenido generado por IA puede ser incorrecto.**

****

**Texto

El contenido generado por IA puede ser incorrecto.**

**Se puede ver como creamos objetos de las Clases y las agregamos a espacio del crud para que esten guardadas como informacion por medio de los métodos.**

**Video**

[**https://drive.google.com/file/d/1sOvgaoRcJF9yu5N-otsvwP3FbYU\_2BmL/view?usp=sharing**](https://drive.google.com/file/d/1sOvgaoRcJF9yu5N-otsvwP3FbYU_2BmL/view?usp=sharing)

**Github**

[**https://github.com/aperezm15/ActividadUnida2.git**](https://github.com/aperezm15/ActividadUnida2.git)