Trabajo Práctico Hotel

23/06/2018

Materia: Laboratorio de Programación III

Integrantes: Rodríguez Gonzalo, Paz Ivo, Ferreyra Franco

Desarrollo:

Las primeras semanas de desarrollo se utilizaron para la investigación. Nos hicimos preguntas propias de un Hotel, cómo por ejemplo: ¿Qué hay en un hotel? Diferentes tipos de Habitaciones para una cantidad determinada de clientes. ¿Cómo funciona, quien o cuantas personas lo administran? En un hotel, un cliente pide una habitación si se presenta de manera física, o puede hacer una reserva. Cuestiones que involucran al hotel y a los usuarios del mismo.

Para la realización del proyecto recurrimos a GitHub. Esta herramienta permitió que los integrantes trabajen sobre un proyecto “master” el cual siempre estaba actualizado con lo último que fuéramos haciendo.

Para la realización del diagrama UML utilizamos el plugin ObjectAid el cual nos facilito el diseño del mismo. Gracias a esto pudimos visualizar nuestro proyecto y corregir errores.

Descripción del diagrama:

Como primera instancia tenemos al Controlador. Este tiene como atributo un Hotel ya que se va a encargar de tareas que no son propias de dicha clase, como puede ser el guardado de información mediante archivos.

Dejando las clases utilitarias nos encontramos con la clase Usuario y Hotel. Esta primera es una clase abstracta, de la cual heredan Admin y Conserje. Su única función es contener atributos propios de un Usuario. Dentro de la clase Hotel vamos a encontrar varias colecciones del tipo TreeMap (elementos se encuentran ordenados por orden ascendente de Clave) Donde se van a guardar: Habitaciones Conserjes Clientes y Reservas. También tiene un atributo del tipo double totalIngresos que guarda información de las ganancias. La clase Cliente es una clase básica que guarda información de los clientes. Como agregado tiene un ArrayList<String> historial y es modificado por el Conserje. Una de las clases mas importantes es la clase Reserva. Al igual que el Cliente es generada por un Conserje. Esta se encarga de especificar fechas de ingresos y salidas de los clientes (check-in, check-out) y los asigna a una Habitación. Además tiene un ArrayList<Productos> consumos, que registra los consumos hechos por un cliente. Siendo Producto una Enumeración.

La clase Habitación además de contener atributos trabaja también con una Enumeración llamada TipoHabitacion.

El encargado de crear Habitaciones y dar de alta Conserjes es el Admin.

La clase IOGenericoUtil, como su nombre lo indica, aplica de forma genérica el funcionamiento de Archivos. La cual nos permite trabajar de forma más eficiente y nos ahorra las sobrecarga de métodos según su tipo.

Otra clase Utilitaria es FechaHoraUtil, esta trabaja con LocalDateTime y DateTimeFormatter. En esta se declara las horas específicas de check-in y check-out. También nos permite formatear el tiempo según nos parezca.

Agregamos una clase Menú que desplegará unas opciones por pantalla para el Usuario, donde al principio le pedirá que ingrese su nombre de usuario y contraseña para acceder al sistema. Para esto, nos ayudamos con una clase llamada Password y restringe los datos de dicha contraseña. Verifica que sea alfanumérica y limita su longitud.

Con todas estas clases damos por entendido que son suficientes para la realización de este trabajo.

Diario de trabajo:

02/05/18

Inicio. Comienzo del diseño en papel, determinación de primeras clases fundamentales. Hotel, Admin, Conserje, Reserva, Habitación, Cliente.

Hicimos una lectura de la consigna, y anotamos posibles "Clases" fundamentales para el funcionamiento de nuestro sistema.

03/05/18

Continuación de diseño. Se determinó que tanto admin, como Conserje deben extender de una clase llamada Usuario y debe ser abstracta.

07/05/18

Puesta en común. Declaración de primeros métodos de las clases. Incógnitas: ¿Debemos poder trabajar con varios admin o basta con sólo tener uno?

¿Cómo deberíamos almacenar los datos de las reservas, habitaciones y conserjes? Listas, mapas, ¿algo más?

09/05/18

Se optó por trabajar con un Admin único. Y se descartó que el sistema debe poseer un Login para poder gestionar el hotel. También se acordó en hacer uso de mapas, específicamente TreeMap. Para el almacenamiento de información respecto a conserje, reservas, y habitaciones.

10/05/18

Atributo adicional llamado 'historial' agregado a la clase Cliente. Revisión de métodos en las clases ya diseñadas.

12/5

Primera codificación de las clases ya diseñadas. Investigación de las clases LocalDate y LocalDateTime, parte de java.date. Consideración de añadir una clase adicional llamada 'Ocupación' en el que irían las reservas confirmadas, es decir, las ocupaciones concretas de los clientes en las habitaciones.

15/05/18 //tutorías

Descarte de la Clase Ocupación por ser considerada redundante. Se añadió un boolean en clase reserva para confirmar la misma. Añadimiento de la lista de Productos dentro de clase Reserva.

16/05/18

Creación de repositorio git. Creación de clase utilitaria 'FechaHoraUtil'. Creación de enumeraciones 'tipoHab' 'EstadoHab' y ' Producto'. Primera implementación de clase Password, siendo esta un atributo de Clase Usuario.

17/05/18

Planteamiento de como qué datos almacenar en archivos. Consideración de utilización de tipos genéricos y métodos estáticos. Pruebas en código.

Mejoras en clase Password. Inconveniente: no logramos encontrar una manera viable de esconder el input del teclado para la contraseña.

21/05/2017

Implementación de la clase utilitaria IOGenericoUtil. Uso de tipos genéricos(que extienden de Serializable) y métodos estáticos. Revisión de diagramas y bocetos en papel. Pasado en limpio de diagramas UML.

22/05/2018 //tutorías

Creación de clase 'Controlador' que maneja un objeto Hotel.

23/05/2018

Creación de Menús dentro de su propia clase.

24/05/18

Primera implementación de switches dentro de la clase Controlador. Implementación de login: inconveniente: Password. Primera implementación del método alta Conserje de la clase Admin.

Discusión acerca de qué clases deberían tener cierto tipo de Comportamientos, pensando en la cohesión y el acoplamiento del programa.

//Test en código de cómo funcionaría Controlador

Diseño de clases en mini-hojas (atributos, métodos con sus respectivos tipo-de-dato)

Descripción del funcionamiento de cada clase.

26/5

Implementación del método que verifica si existen conflictos de Fechas con reservas correspondientes a una habitación ya existentes junto con una en creación.

Ubicada en FechaHoraUtil. Ajuste de creación Ubicada en FechaHoraUtil. Ajuste de Modificadores de acceso en varios métodos de varias clases.

28/5

Pasado en limpio de UML. Implementación de método 'yaespasada' en FechaHoraUtil, que determina si una fecha introducida es pasada a la actual.

29/5

Mejoras en métodos de altaReserva y altaConserje. Implementación de altaHabitación. Utilización de arrojo de excepciones dentro de alta Reserva para evitar la introducción por teclado de fechas erróneas.

30/5

Implementación y revisión de métodos de búsqueda para Clientes, habitaciones y reservas. Creación de clase Archivos, que contienen las rutas donde se

guardará la información necesaria producida por Hotel.

31/5

Mejora en la búsqueda de cliente y método altaCliente. Agregado de métodos funcionales a la clase Controlador.

1/6

Ubicación de procedimientos que inicializan el programa, cómo la lectura de archivos dentro del constructor de Hotel. Si no lee éxito, solamente el objeto a almacenar en un atributo, este se inicializa de manera estándar.

2/6

Creación de check-out, dentro de Conserje. Limpieza de lineas innecesarias y simplificación de código (Ej: cambiarEstado ---> habilitado = ! habilitado)

4/6/2018

Problemas con github Desktop, perdimos 2 horas intentando revertir los cambios.

Lo solucionamos con una versión anterior.

5/6/2018

Una descripción en general: corrección de varios métodos modificación de 2 métodos en admin, modificar habitación precio por día y eliminar habitación, modificar datos de usuario listado reserva, clientes.

agregado método de remover elemento de los mapas

cancelar reserva

Problemática con variables estáticas cuando el sistema inicia, estas volverían a un valor por default.

clase admin: problemas con método: setearTipoHabitacion()

7/6/2018

Renombramos métodos de la clase hotel y admin para que haya un mejor entendimiento para nosotros y los que lean el código. Además de renombrar métodos, corregimos el código (renombrar variables, resumir líneas de \*código\* con la creación de nuevos métodos para mejorar la calidad de lectura y evitar la repetición de código).

modificado:

básicamente toda la clase de admin, y agregamos métodos como generarNuevoPrecioHabitacion, generarTipoHabitacion)

En hotel agregamos métodos que retornan habitaciones que carecen de reservas, con el fin de evitar problemas al settear precios/tipohabitacion etc.

Modularizarían de métodos utilizando exceptions.

18/6

Acomodamos los métodos de todas las clases según para que o quienes son usadas (por ej: las que hacen salida por pantalla estaban debajo de un comentario que dice /\*salida por pantalla\*/)

19/6

Terminación de la documentación Javadoc. Avanzamos con el informe. Se testeo a fondo el programa para la detección de errores. Finalización del registro del proyecto

Manual del Usuario:

1. Como primera instancia se deberá ingresar al sistema, para ello tiene que escribir su usuario y su contraseña. Si es la primera vez que ingresa debera acceder con el usuario admin predeterminado. Usuario: *admin* Password: *password*.
2. Una vez adentro, para su mayor seguridad es necesario que cambie sus datos de ingreso. Para ello vaya a *4. Administrar Información Propia->2. Modificar Informacion.*
3. Dentro de los menús si usted quiere volver a las opciones anteriores solamente debe presionar la opción (cero).
4. Menú administrador:

* *Administrar conserje:*

* + *Dar alta conserje:* Se procede a la creación de un conserje. Pedirá un dni, una contraseña(alfanumérica, mayor a 8 dígitos y menor a 20) y un nombre.
  + *Habilitar/Deshabilitar conserje:* se listarán todos los conserjes existentes en el sistema y deberá ingresar el dni del cual se va a modificar.
  + *Eliminar conserje:* se listarán todos los conserjes existentes en el sistema y deberá ingresar el dni del cual se espera eliminar.
  + *Ver Listado de conserjes:* se listarán todos los conserjes existentes en el sistema. Se mostrara dni, nombre y su estado (habilitado o no habilitado).
* *Administrar habitaciones:*

* + *Creación de habitación:* Se procede a la creación de una habitación. Se pedirá un numero de habitación, especificar de que tipo es (matrimonial, familiar o individual) y el valor de la misma. Este valor tiene que ser mayor a $200.
  + *Modificar tipo de habitación:* se listarán todas las habitaciones existentes en el sistema y deberá ingresar el numero de la cual se va a modificar.
  + *Modificar precio de la habitación:* se listarán todas las habitaciones existentes en el sistema y deberá ingresar el numero de la cual se va a modificar. El precio ingresado deberá ser mayor a $200.
  + *Eliminar habitación:* se pedirá el número de habitación a eliminar. Si la encuentra se procederá a pedir una confirmación.
  + *Ver todas las habitaciones:* se listarán todas las habitaciones existentes en el sistema.
* *Administrar clientes:*

* + *Ver registros de clientes:* Se listarán todos los clientes dentro del sistema.
  + *Eliminar cliente del sistema:* se listarán todos los clientes existentes en el sistema y deberá ingresar el número de dni a eliminar.
* *Administrar información propia:*

* + *Ver información propia:* se mostrará nuestro usuario y nuestro nombre.
  + *Modificar información:* se pedirá que ingrese un nuevo usuario, una nueva contraseña y un nombre. Se procede a una confirmación de datos.
* *Cerrar sesión:* para cerrar sesión o cambiar de usuario solo tiene que presionar la opción (cero).

1. Menú conserje:

* *Gestión de reservas:*

* + *Generar reserva:* 
    - *Crear Cliente:* deberá introducir nombre del nuevo cliente, apellido, dni, nacionalidad y dirección. Deberá confirmar los datos o volver a ingresarlos.
    - *Buscar Cliente:* ingrese numero de dni del cliente ya creado previamente en el sistema.

Una vez asociado al cliente tiene que seleccionar el tipo de habitación que requiera. Deberá ingresar el numero de la misma para asociarla y las fechas de ingreso y salida de la misma en formato dd/MM/yyyy.

* + *Check- in:* se listarán las reservas cuya fecha de ingreso coincide con la de hoy. Seleccione la que desea realizar el check-in.
  + *Check-out:* se listarán las reservas validas para realizar esta acción. Deberá seleccionar una y se procederá a realizar la factura de la misma.
  + *Ver reservas:* se listarán todas las reservas existentes en el sistema.
  + *Cancelar reserva:* introduzca el id de la reserva que desea cancelar.
  + *Agregar consumo en una habitación:* deberá asociar una habitación y se procederá a listar los productos disponibles. Seleccione el producto que desea agregar a la cuenta.
* *Gestión de clientes:*

* + *Ver lista de clientes:* se listarán todos los clientes existentes en el sistema
  + *Modificar datos de un cliente:*  debera ingresar el dni del cliente y debera seleccionar que dato desea modificar.
* *Gestión de habitaciones:*

* + *Ver habitaciones libres:* se listarán todas las habitaciones libres dentro del sistema.
  + *Ver habitaciones ocupadas:*  se listarán todas las habitaciones ocupadas dentro del sistema.
* *Modificar información propia:*

* + *Ver información propia:* se mostrará nuestro usuario y nuestro nombre.
  + *Modificar información:* se pedirá que ingrese un nuevo usuario, una nueva contraseña y un nombre. Se procede a una confirmación de datos.
* *Cerrar sesión:* para cerrar sesión o cambiar de usuario solo tiene que presionar la opción (cero).

Fuentes de información consultadas:

* <https://stackoverflow.com>
* <https://lawebdelprogramador.com>
* <https://javahispano.org>