STUCOM
Centre d'Estudis
www.stucom.com

Homologat i concertat per
la Generalitat de Catalunya

Dept. d'Informàtica i Telecomunicacions	Curs 2018-2019
Grup: DAM / DAW	
M03 Programació	
UF5 Práctica 3	
Nom professor/a: Mª del Mar Fontana	
Data: 15/01/2019	

Práctica 3: Colecciones

En el Hotel Stucom 5 estrellas precisan de un software que les permita gestionar todos los aspectos del hotel a tiempo real. Para ello, nos han encargado un programa que gestione tanto las reservas de las habitaciones, el estado de las habitaciones y los trabajadores del hotel.

De entrada, se deberá cargar el fichero que nos proporciona el mismo Hotel con la distribución de habitaciones, trabajadores en plantilla y reservas a fecha de hoy. Se debe cargar este fichero de una manera eficiente en memoria, utilizando la colección que mejor cumpla las características.

ESPECIFICACIONES

Para gestionar mejor el Hotel, se trabaja con unos estándares homologado según vemos a continuación:

Estado Habitación:

- CLEAN: Puede ser asignada a un cliente.
- UNCLEAN: No puede ser asignada hasta que se limpie.
- BROKEN: Debe ser reparada antes de poderla asignar.
- RESERVED: Está reservada hasta que los clientes salgan.

Servicio en las Habitaciones: tv, balcon, camadoble, jacuzzi, minibar, teléfono, satélite y sweet.

Habilidades de los Trabajadores: mantenimiento, limpieza, piscina, spa, bar, comida y lavandería.

Money: El hotel empezará con 1000 Euros y deberá gestionarse para generar más ingresos.

COMANDOS

Todos los comandos deberán dar feedback. El mensaje de respuesta a un comando deberá mostrarse en azul. Los comandos de información se mostrarán en magenta. Los mensajes de error deberán mostrarse en rojo.

El comando **ROOM** añadirá una habitación al hotel. Las habitaciones están numeradas con 3 dígitos y se deberá especificar la capacidad máxima y los servicios que tiene incluidos.

```
ROOM 007 2 CAMADOBLE,TV
--> new Room added 007 <--
room 008 1 satelite,telefono,tv
--> new Room added 008 <--
ROOM 001 1 tv
[ Room already exists ]
ROOM 009 2 sauna,tv,satelite
[ Wrong service ]
```

El comando **WORKER** añadirá un nuevo trabajador. Los trabajadores están identificados por un DNI (parte numérica), un nombre y los habilidades que tienen para ofrecer servicios a los clientes.

```
WORKER 33333333 Antonio limpieza,mantenimiento
--> new Worker added 33333333 <--
worker 44444444 Paco piscina,bar
--> new Worker added 44444444 <--
worker 33333333 Pepe piscina,mantenimiento
[ Worker already exists ]
worker 55555555 Sandra piscina,mantenimiento,sauna
[ Wrong service ]
```

El comando **RESERVATION** añadirá una nueva reserva al hotel. Para ello se creará un nuevo cliente (Customer) con los datos introducidos a continuación de la reserva que son el DNI (parte numérica), el número de personas que son y los requisitos para la habitación.

Este comando deberá asignar la **habitación que cumpla todos los requisitos** que los clientes piden y también se deberá asignar la habitación que **más se acerque en número de plazas** al número de personas de la reserva.

```
RESERVATION 11223344 1 tv
--> Assigned 11223344 to Room 003 <--
RESERVATION 33445566 2 camadoble,tv,telefono
--> Assigned 33445566 to Room 004 <--
reservation 55667788 1 tv
--> Assigned 55667788 to Room 006 <--
reservation 88888888 4 jacuzzi,tv,balcon
[ There isn't any room available. Customer not asigned. You've lost 50€ ]
```

El comando **HOTEL** pintará la distribución de habitaciones de una manera visual, donde indicará el número de habitación y si no está vacía, indicará el DNI del cliente y el número de personas que hay.
Además mostrará un listado de los trabajadores y si están en alguna habitación indicará en cuál.

```
==> ROOMS <==
== ROOM 001 BROKEN ==
== ROOM 002 CLEAN ==
== ROOM 003 BROKEN ==
== ROOM 004 BROKEN ==
== ROOM 005 BROKEN ==
== ROOM 006 CUSTOMER:55667788(1) ==
== ROOM 007 CLEAN ==
== ROOM 008 CUSTOMER:11223344(1) ==
=====================================================
==> WORKERS <==
== WORKER 11111111 Juan ROOM:001 ==
== WORKER 44444444 Paco ROOM:003 ==
== WORKER 33333333 Antonio ROOM:001 ==
== WORKER 12345678 Pepe ROOM:003 ==
== WORKER 22222222 Clara ROOM:006 ==
== WORKER 88888888 Maria AVAILABLE ==
=====================================================
```

El comando **PROBLEM** informará de un problema en alguna de las habitaciones, para ello, se deberá buscar la habitación, marcar su estado a estropeada (BROKEN).

Cuando una habitación se marca con problemas, se deberá mover a los clientes a otra que cumpla sus requisitos o desasignarlos en caso de que no se tengan disponibles.

```
PROBLEM 001
--> Room set as BROKEN <--
PROBLEM 888
[ There isn't room with this number ]
problem 004
--> Room set as BROKEN <--
--> Assigned 33445566 to Room 005 <--
problem 005
--> Room set as BROKEN <--
[ There isn't any room available. Customer not asigned. You've lost 50€ ]
```

El comando **REQUEST** será una petición de una habitación por un servicio. Se buscará entre los trabajadores disponibles y se les asignará uno por cada una de las peticiones que tengan. En caso de no cumplir con alguna de las peticiones se guardarán para poder satisfacerlas antes de que se vayan del Hotel.

```
request 001 mantenimiento,limpieza
--> Worker Antonio assigned to Room 001 <--
--> Worker Juan assigned to Room 001 <--
request 003 piscina,mantenimiento,limpieza
--> Worker Paco assigned to Room 003 <--
--> Worker Pepe assigned to Room 003 <--
--> No Worker available for this service <--
request 006 comida,bar
--> Worker Clara assigned to Room 006 <--
--> No Worker available for this service. Added to customer pending request <--
request 888 bar
[ There isn't room with this number ]
request 001 informatica,limpieza
[ Wrong service ]
```

El comando **FINISH** se utiliza para indicar que se han finalizado las peticiones de una habitación. Quedarán liberados los trabajadores y la habitación quedará en estado CLEAN, en caso de estar vacía, o RESERVED, en caso de tener cliente.

```
FINISH 001
--> Services finished in room: 001 <--
FINISH 004
--> There aren't workers in room <--
finish 003
--> Services finished in room: 003 <--
finish
[ Wrong number of arguments ]
finish 888
[ There isn't room with this number ]
```

El comando **LEAVE** informará de cuando unos clientes se van de su habitación. Esta quedará libre, los trabajadores asignados también y además pasará a estar UNCLEAN para futuras reservas. También informará del dinero que los clientes pagan siempre que tengan todas sus peticiones (Requests) cumplidas, de lo contrario, el hotel perderá la mitad de este valor.

```
LEAVE 888
[ Wrong number of arguments ]
LEAVE 888 500
[ There isn't room with this number ]
leave 006 400
-> Room 006 free and set to UNCLEAN <--
--> Unsatisfied clients. You loose 200 € <--
leave 008 300
-> Room 008 free and set to UNCLEAN <--
--> Satisfied clients. You win 300 € <--
```

El comando **MONEY** informará del dinero actual del hotel. Si en algún momento durante la ejecución de la aplicación el hotel se queda sin dinero deberá dar el mensaje correspondiente y finalizar.

```
MONEY
=======================================================
==>    MONEY : 800 €    <==
=======================================================
```

```
money
=======================================================
==>   MONEY : 100 €   <==
=======================================================
reservation 1 8 tv
[ There isn't any room available. Customer not asigned. You've lost 100€ ]
=======================================================
==========    YOU'VE LOST ALL YOUR MONEY    ===========
=======================================================
```

El comando **EXIT** finaliza la ejecución de la aplicación.

Debe respetarse el formato de los mensajes de salida de la aplicación tal y como indica el enunciado.

COLLECTIONS

Para la práctica se deberá escoger la mejor colección para cada tipo de almacén de datos según lo que se ha visto en la teoría. Destacamos que en el controlador se deberá tener, entre otras muchas cosas, un seguido de collections que permitan:

- Almacenar que Customer está en cada Room.
- Almacenar que Worker está en cada Room.
- Almacenar todas las Rooms.
- Almacenar todos los Wokrers.
- Almacenar todos los Customers.
- Almacenar de cada Customer/Room/Worker sus colecciones de atributos (servicios, requisitos, habilidades, etc.).

Condideraciones

La entrega deberá realizarse con un único fichero .zip con la carpeta completa del proyecto. El fichero zip debe tener nombre y primer apellido del alumno. Ejemplo: marfontana.zip.

El formato de entrada/salida de la aplicación deberá seguir el patrón indicado en el enunciado y en el ejemplo aportado en el campus. Si no pasa el juego de prueba básico subido en el campus la práctica quedará suspendida.

Criterios de corrección

- La entrega se deberá hacer en un archivo comprimido con todos los ficheros necesarios para su correcta ejecución (ficheros de entrada, .java, etc.).
- Es obligatorio el uso de collections correspondientes a cada necesidad. En caso de duda se pedirá una justificación de la elección.
- Se debe utilizar la lectura de fichero anteriormente estudiada en clase del mismo modo que se pide un trato correcto de Excepciones y su propagación.
- Documentar el código.

1-5 => Si el código compila y pasa los ficheros de prueba.
5-7 => Correcta elección de collections, uso de excepciones y de ficheros.
7-8 => Código limpio y ordenado. Fácil de seguir.
8-9 => Código documentado y sintetizado. Agradable al a vista.
9-10 => Llevas la práctica un paso más lejos.