Dept. d'Informàtica i Telecomunicacions	**Curs 2018 – 2019**
Grup: DAM2T	
M03 Programació	
UF5 Pràctica 5	
Professor Héctor Florido	
Data 11/01/2019	

2Práctica 5: 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, saltelite 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

El comando **SPEED** explicado más adelante.

```
SPEED 2000
```

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 001 3 tv,jacuzzi,balcón
--> new Room added 001 <--
ROOM 002 2 tv,camadoble
--> new Room added 002 <--
```

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 56473847 Antonio limpieza,mantenimiento
--> new Worker added 56473847K <--
WORKER 78876847 Rodrigo piscina,mantenimiento,bar
--> new Worker added 78876847L <--
```

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.

```
RESERVATION 47655768 1 balcon
--> Assigned 47655768F to Room 001 <--
```

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.

El comando **HOTEL** pintará la distribución de habitaciones de una manera visual y de color rojo, donde indicará el número de habitación y si no está vacía, indicará el DNI del cliente y el número de personas que hay.

```
HOTEL

==============================
== ROOM:001  CUSTOMER:233(1) ==
==============================
== ROOM:002       EMPTY      ==
==============================
== ROOM:003       EMPTY      ==
==============================
```

PETICIONES

Una vez está toda la información del hotel cargada, se deberá realizar un programa que gestione la gestión del hotel día a día, para ello recurriremos a la creación de un **Thread** de lectura encargado de la lectura de un fichero de Peticiones de los clientes.

Este thread deberá leer de una en una todas las peticiones del fichero de lectura de peticiones y deberá tratarlas debidamente.

COMANDOS

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

```
PROBLEM 001
--> Assigned Antonio to Room 003 <--
--> Room set as 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.

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 003 piscina,bar,bar
--> Worker Rodrigo assigned to Room 003 <--
--> Worker Maria assigned to Room 003 <--
--> No Worker avaliable for this Service. Added to Customer pending Requests_<--
```

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 003 500E
--> Room 003 free and set to UNCLEAN <--
--> Worker Rodrigo desasigned<--
--> Worker Maria desasigned<--
--> Unsatisfied clients. You loose 250 E  <--
```

El comando **RESERVATION** funciona del mismo modo que hemos visto anteriormente. Siempre que unos clientes no se asignen a una habitación por no cumplir con sus requisitos, se perderán 100€ del dinero del Hotel.

```
RESERVATION 47655768 5 satelite
--> Customer not asigned. You loose 100<--
```

También, como complemento a este Thread, se deberá marcar un **tiempo de lectura** para que entre cada línea del fichero pase un tiempo estipulado en el fichero de carga del hotel. Este tiempo de lectura vendrá marcado por el comando SPEED, y marcará, en milisegundos, el tiempo entre cada petición. Cada petición marcará un día distinto.

```
SPEED 2000
```

El código de cada una de las lecturas deberá incluir los siguientes campos:

```
========================================
    DAY: 1  ===  Money: 1000
========================================
PROBLEM 001
--> Assigned Antonio to Room 003 <--
--> Room set as BROKEN <--
```

Donde tenemos un contador de días y La variable Money con los beneficios del Hotel de manera muy visible para poder gestionar las Peticiones. A continuación aparecerá la petición y como se

ha tratado. Si en algún punto de la lectura de las peticiones el Hotel llega a 0 o menos de dinero, se deberá parar la lectura del fichero del Thread e imprimir el final.

```
========================================
    YOU'VE LOST ALL YOUR MONEY
========================================
```

O en el caso de completar todos los comandos del fichero, se imprime el dinero final del Hotel:

```
========================================
    FINAL MONEY: 650€
========================================
```

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:

- 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.).

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 compilar 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.