Normas generales:

- La duración del examen es de 2 horas (más 10 minutos para iniciar el entorno informático).
- Esta parte práctica puntúa sobre 7 puntos. Los otros 3 puntos del examen corresponden al test.
- El alumno debe ejecutar la aplicación **ExamWatcher** (disponible en la pestaña otros recursos del campus virtual) durante el examen. Dicha aplicación monitoriza las acciones del usuario y por ello estará activa durante todo el examen. Cuando termine el examen, la aplicación tiene un botón para generar un fichero ZIP con el directorio en el que esté el código del alumno. El alumno debe escribir su nombre completo en el campo correspondiente de la aplicación antes de generar el ZIP. El alumno debe subir dicho ZIP a la actividad examen práctico de junio del campus virtual.
- Desde el campus virtual hay que descargar un archivo ZIP que contiene el **esqueleto** de los dos ejercicios del examen así como los test unitarios correspondientes. Se recomienda crear un proyecto de Netbeans vacío; copiar dentro de la carpeta del proyecto la carpeta src y la carpeta test proporcionadas; añadir las librerías `JUnit` y `Hamcrest`; y finalmente cerrar y volver a abrir Netbeans.
- Al inicio del examen debe firmarse el examen y rellenarse el nombre y los apellidos en la cabecera de este documento.
- Justo antes de entregar el examen deberá rellenarse el campo "hora de entrega" en la cabecera de este documento.

Ejercicio 1 – Creación de estructura de datos [3 puntos]

En clase se ha visto cómo la función `rehash`, de la clase `OAHashDictionary`, debe duplicar el tamaño de la tabla hash que contiene, recalculando la función de hash para todas las entradas existentes. Esta función se invoca cuando el número de claves almacenados en la tabla hash supera cierto umbral, de manera que se prevengan colisiones. Sin embargo, en dicha clase no existe un equivalente que permita ahorrar espacio cuando, tras haber crecido previamente, las claves se eliminan mediante el método `remove`.

Si el método `rehash` recibiese mediante un entero la nueva capacidad, sería posible cambiar `put` (y `remove`) para que al insertar (o borrar) un valor de la tabla se comprobase el tamaño y se invocase a `rehash` con un valor doble (o mitad).

En este ejercicio **se pide implementar en Java un nuevo método `rehash`** en la clase `OAHashDictionary` que reciba un parámetro con la nueva capacidad deseada (2 punto).

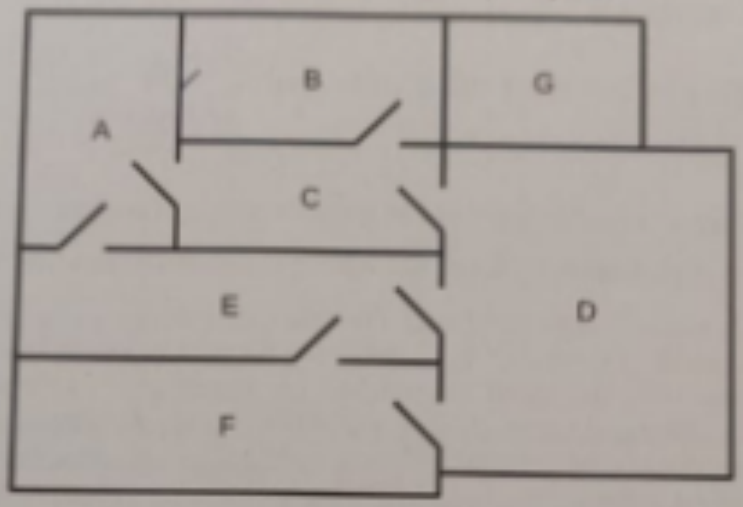
También **se pide implementar los métodos `put` y `remove`**, que deberán implementar el comportamiento descrito para posibilitar el crecimiento y el decrecimiento de la estructura (1 punto).

En el código suministrado hay test para comprobar el correcto crecimiento y decrecimiento de la tabla hash.

Ejercicio 2 – Aplicación [4 puntos]

Para gestionar las visitas al Palacio Real se desea crear una aplicación que permita diseñar un itenerario de una sala a otra.

La descripción del palacio se realiza mediante la enumeración de las puertas. Por ejemplo, la figura adjunta responde a la siguiente enumeración: A-C, B-C, C-D, D-E, D-F, A-E, E-F.



Si en la figura adjunta se pide un itinerario de la sala A a la sala F hay dos posibilidades: [A,C,D,F], [A,E,F]. Sin embargo, se desea que el itinerario atraviese el menor número de puertas posibles. Por ello, para este caso, el único itinerario correcto de la sala A a la sala F sería [A,E,F].

Para construir dicha aplicación se propone crear la clase `SmartPalaceMap`. Dicha clase deberá tener los siguiente métodos:

- `Room insertRoom(List<Room>)`.- Crea una habitación, indicando todas las habitaciones concetadas a ella, devolviendo la referencia a la nueva habitación.
- `List<Room> getPath(Room, Room)`.- Devuelve la lista de habitaciones.

La interfaz de la clase `SmartPalaceMap` se proporciona junto con algunos test unitarios.

Se pide:

a) Añadir las propiedades que se consideren oportunas a la clase `SmartPalaceMap` para cumplir el objetivo deseado (1 punto).

b) Implementar el método `insertRoom` de acuerdo con las propiedades añadidas (1 punto).

c) Implementar el método `getPath` de acuerdo con las propiedades añadidas (2 punto).