Problema 3: Viaje intergaláctico

Enunciado:

La nave de la flota intergaláctica Enterprise se tiene que preparar para un viaje por el espacio profundo. Una de las tareas más importantes es seleccionar las armas que han de llevar en la nave. Ésta ha de contener todas las características requeridas para la correcta protección de la nave y, además, ha de pesar lo menos posible.

Para llenar la nave, es necesario considerar que hay una lista de características requeridas que deben cubrirse por las armas seleccionadas. Por ejemplo, a continuación, se muestra una lista de características requeridas para un escenario concreto:

Requisito-1	Requisito-2	Requisito-3	Requisito-4
Láser	Paralizante	Cuerpo-Cuerpo	Ligera

Para llenar la bodega de la nave, se dispone de un hangar con las distintas armas que están disponibles, dichas armas se describen por sus características, su peso y el daño que producen. Un arma del hangar puede responder a más de una necesidad, por ejemplo, si se necesita un arma láser ligera y un arma láser, con un arma láser ligera sería suficiente. A continuación, se muestra un ejemplo de hangar donde se dispone de las siguientes armas:

Nombre:	*P-90*	*P-90-2*	*Disruptor Kull*	*Zat*	*OriZouk*
Peso:	150g	250 g	250 g	220 g	100g
Características:	Láser, Ligera	Láser, Ligera	Paralizante, Ligera	Láser, Paralizante, Cuerpo-Cuerpo	Torpedo
Daño:	750	650	500	350	100

Una solución podría ser las armas **P-90** -que cumple la primera y la cuarta características requeridas de la lista previamente indicada- y **Zat** – que cumple la primera, segunda y tercera características requeridas de la lista previamente indicada-. El peso de dicha solución sería 150 + 220 = 370 g. Otra posible solución sería utilizar Disruptor Kull**,** Zat **e** OriZouk cuyo peso sería 570 g. El objetivo **es minimizar el peso que llevamos en la nave**, por lo que la primera solución será la elegida.

Modele y resuelva este problema utilizando grafos virtuales de manera que, de cada vértice salgan dos aristas, una indicando que se coge cierta arma del hangar y otra indicando que dicho arma no se coge.

Para ello considere que **los objetos del hangar (clase ObjetoHangar) tienen identificación** (String), **peso** (Float), **daño** (Integer), y una **lista de características** (List<String>). Las necesidades del viaje se especificarán como una lista de características. Cada arma tendrá como propiedad una lista de características, por lo tanto, **las necesidades son listas de listas de características** (List<String>).

Preguntas

1. Justifique el tipo de grafo virtual que ha de utilizar y diseñe la clase de sus vértices.
2. Indique justificadamente cómo modelará los pesos de los elementos del grafo.

3. Implemente todo lo necesario para resolver el problema mediante el algoritmo de AStar. Puede utilizar el código provisto en el paquete `us.lsi.astar.viajeintergalactico`.
4. Complete la clase `TestViajeInterestelar` de modo que permita lanzar, al menos, el ejemplo indicado.

Ejemplo de requisitos para pruebas

```
List<String> requisitos= Lists.newArrayList("Laser", "Ligera", "Paralizante",
"Cuerpo-Cuerpo");
```

Ejemplo de hangar para pruebas

```
List<ObjetoHangar> hangar = Lists.newArrayList(
new ObjetoHangar("P-90", Lists.newArrayList("Laser", "Ligera"), 150f, 750f),
new ObjetoHangar("P-90-2", Lists.newArrayList("Laser", "Ligera"), 250f, 650f),
new ObjetoHangar("Disruptor kull", Lists.newArrayList("Paralizante", "Ligera"),
250f, 500f),
new ObjetoHangar("Zat", Lists.newArrayList("Laser", "Paralizante", "Cuerpo-Cuerpo"),
```

Function para verificación de objetivo

Dada una serie de necesidades (List<String>) y una clase EstadoNave que contiene una propiedad con una lista de armas seleccionadas (List<ObjetoHangar>), el siguiente trozo de código crea un Function que verifica que dichas armas son adecuadas para el viaje, es decir, que todos los requisitos han sido cubiertas por, al menos, un arma:

```
Function<EstadoNave, Double> function = (e -> {
       if (requisitos.stream().allMatch(req ->
            e.getArmas_seleccionadas().stream().anyMatch(arma ->
            arma.caracteristicas.contains(req)))) {
              return 0.0;
       } else {
              return Double.MAX_VALUE;
       }

});
```