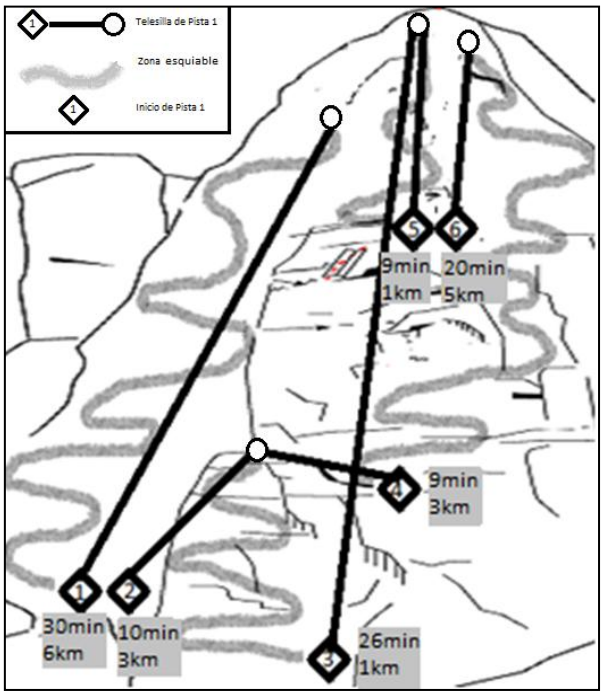
Problema Voraz: El esquiador

Descripción del problema

En una estación de esquí, un esquiador pretende aprovechar al máximo el tiempo que estén los telesillas abiertos de manera que pueda esquiar la máxima distancia por las pistas de nieve. Para poder descender por una pista, primero hay que subir por un telesilla. En este problema se supondrá que cada telesilla lleva a una sola pista, por lo que se hablará indistintamente del tipo de dato *Pista* que recoge los dos conceptos. **Una *Pista* comienza en la entrada de un telesilla y termina en el final de la zona esquiable.** En el inicio de cada *Pista* hay un cartel que informa de su ***longitud*** esquiable y del ***tiempo*** medio que se tarda en subir el telesilla y bajar esquiando. La fracción ***longitud/tiempo*** le indica al esquiador con qué *Pista* aprovecharía mejor el tiempo, por lo tanto, cuando tenga que elegir entre varias, elegirá que haga mayor dicha fracción.



El esquiador, cuando llega a la montaña tiene disponible ciertas *Pistas* por las que comenzar. Cada *Pista* le llevará a un lugar diferente permitiéndole acceder a otras *Pistas*.

En la imagen, considerando 1 y 2 como *Pistas* iniciales, se observa que, para acceder a la *Pista* 4 , habría que coger la 2 , después la 3 y, por último, la Pista 6 .

Se modelará lo arriba descrito para resolver el problema del esquiador, que consiste en planificar el tiempo disponible para esquiar maximizando la distancia recorrida en las pistas.

Modelado:

Para simplificar el problema, se tendrán en cuenta las siguientes suposiciones:

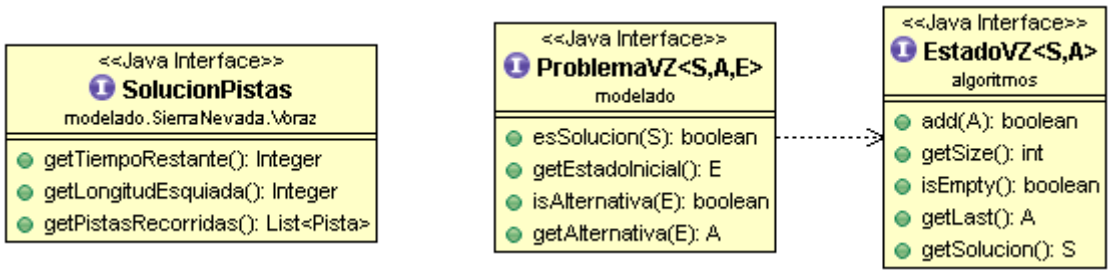
1. El tipo de dato *Pista* representa a un telesilla que lleva hasta una pista de nieve que se va a esquiar. Las propiedades de *Pista* son:
    a. ***tiempo***: es el tiempo que se consume cuando se recorre dicha pista. Este tiempo es el comprendido desde que se pone a la cola del telesilla, hasta que se finaliza la pista después de esquiarla.
    b. ***longitud***: es la distancia que el esquiador recorre cuando esquía la pista.
2. Cada *Pista* da acceso **a una o a varias** *Pistas*. Existirá una propiedad compartida (del Problema), ***accesibilidad,*** de tipo *Map<Pista, SortedSet<Pista>>* que, para cada *Pista*, almacena a las que se pueden acceder una vez terminada, es decir, las **alternativas** del esquiador. En la imagen, la Pista 3 da acceso tanto a la Pista 5 como a la Pista 6 .
3. Para comenzar a esquiar, se dispone de un conjunto de *Pistas* a las que puede acceder inicialmente. Existirá otra propiedad compartida (del *Problema*), ***pistasIniciales***, de tipo *SortedSet<Pista>* que contendrá las *Pistas* que pueden ser accedidas inicialmente.
4. Cada *Problema* tendrá una propiedad ***tiempoTotal*** que hace referencia al tiempo que las *Pistas* permiten **acceder** a los esquiadores.
5. En cada *Estado*, el esquiador tiene que decidir qué *Pista* elegirá de todas las que tiene disponibles en ese *Estado*. Elegirá aquella *Pista* que tenga mayor ***longitud/tiempo***. Tal y como se indica en la ficha proporcionada, el orden que se tiene en cuenta en los conjuntos ordenados está en función de esa fracción, por lo que **primero estará la *Pista* que el esquiador elegirá.** En un *Estado* cualquiera, el esquiador podrá acceder a una *Pista* siempre y cuando le quede algo de ***tiempoDisponible.***

Ejemplo de ejecución:

Dados los datos de ***accesibilidad*** de la imagen, teniendo como ***pistasInciales*** la 1 y la 2 y un ***tiempoTotal*** de 60 min, la solución obtenida utilizando un esquema voraz, debería ser.

- ***pistasRecorridas***: Lista con las pistas 2 , 3 , 6 , 4 .
- ***tiempoRestante***: Entero con valor igual a -5.
- ***longitudEsquiada***: Entero con valor igual a 12.

Lo que representa que se han esquiado 12 km por las pistas indicadas y que, cuando se terminó de esquiar, los telesillas hacía ya 5 minutos que no dejaban entrar a más gente.



Se pide:

1. Rellenar las partes de la ficha que no estén rellenas.
2. Implementar los métodos (Deberá corresponderse con lo que escriba en la ficha):
   a) `private void voraz()`.
   b) `public boolean isAlternativa(EstadoPista e)`. De *ProblemaPistasImpl*.
   c) `public Pista getAlternativa(EstadoPista e)`. De *ProblemaPistasImpl*.
   d) `public boolean add(Pista p)`. De *EstadoPistasImpl*.
   e) `public boolean esSolucion(SolucionPista s)`. De *ProblemaPistasImpl*. Tendrá que comprobar que los 3 atributos de la solución (ver UML) son válidos.

Problema esquiadores	
Técnica: Voraz	
Tamaño: t=tiempoDisponible	
Propiedades Compartidas	***pistasIniciales**: SortedSet<Pista>, ordenada por (tiempo/longitud) de mayor a menor* ***accesibilidad**:Map<Pista,SortedSet<Pista>>, ordenada por (tiempo/longitud) de mayor a menor* ***tiempoTotal**: Integer*
Propiedades del Estado	***pistasRecorridas**: List<Pista>* ***distanciaRecorrida**: Integer* ***tiempoDisponible**: Integer*
Solucion: SolucionPistas	
Alternativa: //TODO: Sólo tendrá en cuenta las pistas accesibles desde la última añadida A_{estado}*={*	
Elección: //TODO: Se tendrán en cuenta las propiedades longitud y tiempo de cada Pista *h(estado,*a_i*)=*	
Estado Inicial: //TODO: Habrá que considerar una de las pistas iniciales	
Estado Final: //TODO: Se tendrá que indicar cuándo el esquiador no puede continuar	
Add(a_i*): //TODO: Los valores del estado tendrán que actualizarse con la nueva Pista*	

Tiempo estimado: 60 min. Puntuación: 5 puntos