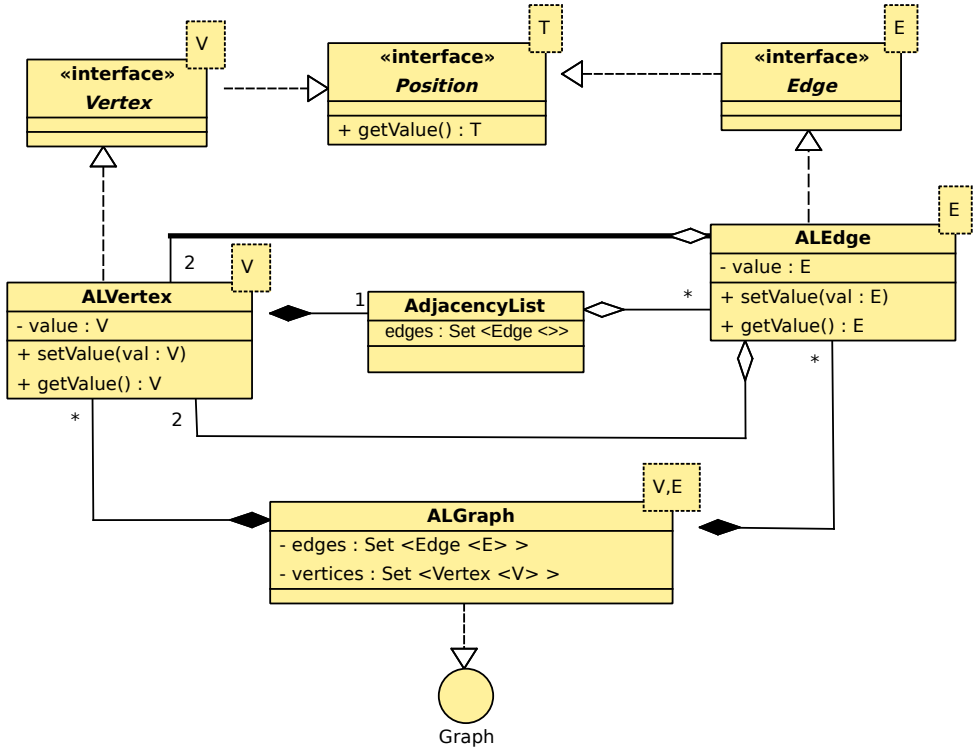
Práctica 4: Grafos

Normas:

- Cada alumno debe realizar los ejercicios de manera individual, aunque pueden compartir información oral.
- Este enunciado va acompañado de un fichero ZIP que contiene el código necesario para hacer la práctica. Los test que acompañan a las prácticas son orientativos y pueden ser ampliados.

Ejercicio 1 – Grafo con Listas de adyacencia

Derivando de `Graph`, implementar la clase `ALGraph` que permitirá construir grafos con listas de adyacencia. Esta clase será similar a su homologa `ELGraph`.



Se recomienda crear un fichero de test que pruebe dicha clase similar al que existe para `ELGraph`.

Ejercicio 2 – Cierre transitivo de un digrafo

Implementar una clase llamada `GraphClosure` que permita realizar el cierre transitivo de un digrafo. La clase tendrá un único método público, llamado `transitiveClosure`, que recibirá un objeto derivado de `Digraph` y devolverá un `Digraph` que sea el cierre transitivo del digrafo

original. La clase podrá tener otros métodos privados. Se proporcionan algunos test de ayuda.

Ejercicio 3 – El parchís

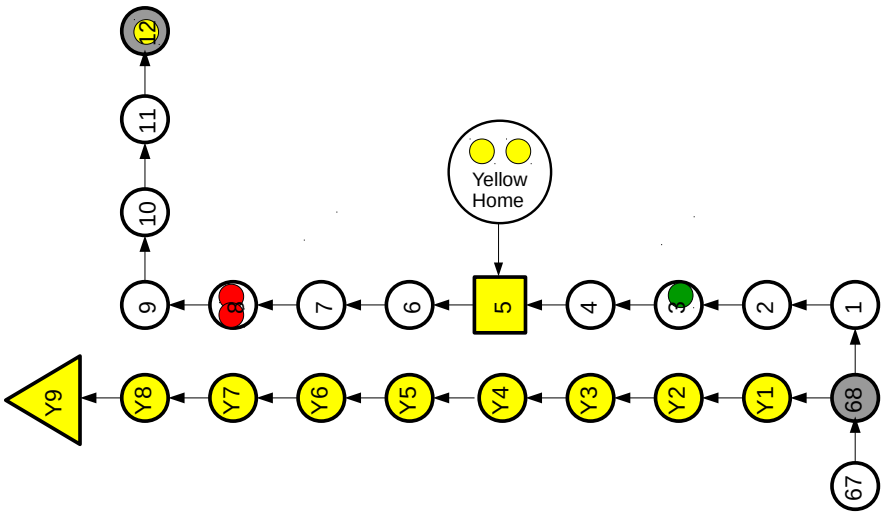
Implementar la clase `ParchisChip` y la clase `ParchisBoard` para facilitar la creación de un juego de parchís.

Para implementar dicha clase se recomienda utilizar internamente una propiedad de tipo digrafo. Los nodos de dicho digrafo deberán almacenar información como el número de la casilla, el color, tipo de casilla (seguro, salida, casa...), y una lista con las fichas que contiene.

También se recomienda que la clase `ParchisChip` contenga una referencia el vértice en el que se encuentra.

Se proporciona un conjunto de test que deberá cumplir dicha clase, aunque se recomienda mejorar dichos test para comprobar el correcto funcionamiento.

Las siguientes figuras ilustran el problema y un caso de uso.



Sobre la figura anterior:

- `a=getChip(0,home,yellow)`: Devolverá una referencia a una ficha de la casa amarilla.
- `canMove(a,1)`: devolverá true, porque un paso desde yellow home lleva a la casilla 5 y está vacía.
- `move(a,1)`: devolverá `null`, porque en 5 no hay otra ficha de otro color.
- `canMove(a,3)`: devolverá false, porque en 8 hay un puente.
- `b=getChip(8,normal,red)`: Devolverá una referencia a una ficha roja de la casilla 8.
- `canMove(b,4)`: devolverá true, porque en 12 hay sitio.
- `move(b,4)`: moverá la ficha roja a 12 y devolverá `null`, porque la ficha amarilla de 12 no puede ser comida al estar en seguro.
- `canMove(a,3)`: devolverá true.
- `c=move(a,3)`: moverá la ficha amarilla de 5 a 8 y devolverá una referencia a la ficha

roja que queda en 8.