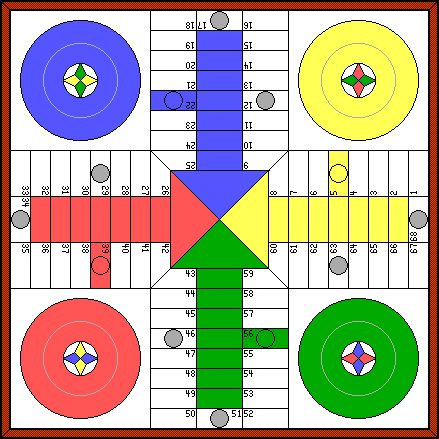
# REGLAS DEL JUEGO



El juego comienza con todas las fichas de cada color en sus respectivas casas. Los jugadores tiran el dado por turnos e irán incorporando una ficha al tablero por cada 5 que saque el dado. La ficha que sale de casa se colocará en la casilla del color del jugador y será una ficha movible.

En cada tirada del dado el jugador que tenga el turno podrá mover una sola ficha tantas casillas como número haya salido en el dado.

/\*

\* 0 - Casa amarilla

\* 1 - Casa azul

\* 2 - Casa roja

\* 3 - Casa verde

\* 12..71 - Tablero

\* 72..79 - Pasillo amarillo

\* 80..87 - pasillo azul

\* 88..95 - pasillo rojo

\* 96..103 - pasillo verde

\* además, las casillas seguras para todos son:

\* 4,5,6,7,8,9,10,11\*/

# CONJUNTO DE REGLAS CONSIDERADAS

# ESTRUCTURA DEL PROGRAMA

Algoritmo Minimax:

* formulation.jar:
  + Action
  + Problem
  + State
* minimax.jar:
  + MinimaxAlgorithm
  + Node
* minimaxAlgorithm:
  + ParchisProblem
  + actions
    - MoveChip
      * isApplicable(State, int)
        + Si la ficha no está en el pasillo final.
        + Si no hay barreras de otro color en la distancia a recorrer.
        + Si la ficha está fuera de casa
        + Si la ficha no está en la casilla final
      * effect(State, int)
        + La ficha se mueve hacia delante el número de casillas correspondiente
    - GetOutOfHome
      * isApplicable(State, int)
        + Si la ficha está en casa
        + Si el dado ha sacado 5
        + Si no tengo barrera de mi color en la casilla de salida
      * effect(State, int)
        + La ficha sale y se queda en la casilla de salida
        + Si hay dos fichas de otro color en la casilla se come la última que ha llegado
        + Si hay una ficha de mi color hago barrera
    - EatChip
      * isApplicable(State, int)
        + Si en la última casilla del movimiento que toque hay una ficha de otro color distinto al nuestro y no está en seguro.
      * effect(State, int)
        + Se mueven 20 casillas con la ficha que se quiera
        + La ficha comida vuelve a casa
    - MakeABarrier
      * isApplicable(State, int)
        + Si en la última casilla del movimiento que toque hay SÓLO una ficha de nuestro color
        + Si la ficha no está en la casilla final
      * effect(State, int)
        + Las fichas que no sean del color de la barrera no pueden pasar
    - ReturnChipToHome
      * isApplicable(State, int)
        + Si ha sacado tres 6 seguidos
        + Si está en el circuito
      * effect(State, int)
        + La ficha vuelve a la casa del principio
    - DestroyABarrier
      * isApplicable(State, int)
        + Si hay una barrera hecha
        + Si en la tirada de dados sale un 6
        + Si no quedan más fichas en el tablero de ese color
      * effect(State, int)
        + La barrera se rompe moviendo una ficha
        + Todas las fichas pueden pasar
        + Las fichas que constituian la barrera pueden moverse
    - EnterFinalCorridor
      * isApplicable(State, int)
        + si con el movimiento entramos en el pasillo
      * effect(State, int)
        + la ficha no se puede comer
        + si la tirada es mayor que las casillas que faltan para llegar la ficha no se mueve
        + las barreras no afectan
        + no le afecta sacar tres 6 seguidos
        + la ficha entra en el pasillo
    - EnterFinalSquare
      * isApplicable(State, int)
        + si con la tirada del dado entra en la casilla final
      * effect(State, int)
        + la ficha ha terminado, queda fuera del juego
    - MoveChipInCorridor
      * isApplicable(State, int)
        + si la ficha está en el pasillo
        + si la tirada es menor que las casillas que le quedan hasta la última casilla (incluida)
      * effect(State, int)
        + la ficha se mueve dentro del pasillo
    - JumpABarrier
      * isApplicable(State, int)
        + si las ficha es del mismo color que las fichas de la barrera
        + si con el movimiento no se queda en la casilla de la barrera
      * effect(State, int)
        + salta la casilla de la barrera
  + Dice
    - public int ThrowDice()
    - integer throws //veces que ha tirado
    - getThrows()

Cambiar funcion expand para que pase menos parametros ¿?

la information del state es la partida

en el effect hay que poner el rating y el movimiento de las fichas!!!!

104 significa que esta en casa