# Introducción

Este documento se creó para dejar constancia por escrito de las especificaciones que debía cumplir la parte del prototipo implementada en Java (llamado SimuMatch dada que su función era la de simular partidos).

El prototipo consta de dos partes, como se ha indicado en el documento “Reglas del juego de cartas” y la implementación en Java, quedando ésta última explicada a lo largo de este documento.

Nota: las imágenes son completamente originales y creadas expreso para el documento.

# Índice

[1. Propósito del programa](#h.sldm2p2z0n7u)

[2. Definición del programa](#h.ptkuv1xe6bh5)

[2.1 Clases](#h.e085annvovj9)

[2.1.1 Class GraphicInterface](#h.dgo74i9j0etr)

[Definición esquemática de la interfaz gráfica](#h.f6xls0iowq1l)

[2.1.2 Class AbilitiesData](#h.88yzapupcknu)

[2.1.3 Class Engine](#h.it54ptt38495)

[2.1.4 Class MatchSimu](#h.p49yukf3l2z)

# 

# 

# 

# 

# 1. Propósito del programa

El programa servirá para **simular el sistema de partidos** introduciendo las variables que condicionan un partido y mostrando paso por paso los resultados de estos.  
Las variables de entrada comprenderán el abanico de habilidades o acciones que interfieren en el desarrollo de los partidos.

Los resultados mostrados comprenderán

* Un registro de las tablas de estados/probabilidad asociada a cada estado de todos los turnos jugados
* El estado actual

# 2. Definición del programa

## 2.1 Clases

El programa, en principio, contará con cuatro clases. Los nombres de las clases así como su funcionalidad quedan definidas en este documento; no así la distribución de paquetes que se deja a elección de los programadores. Tendremos:

* Class GraphicInterface
* Class AbilitiesData
* Class Engine
* Class MatchSimu

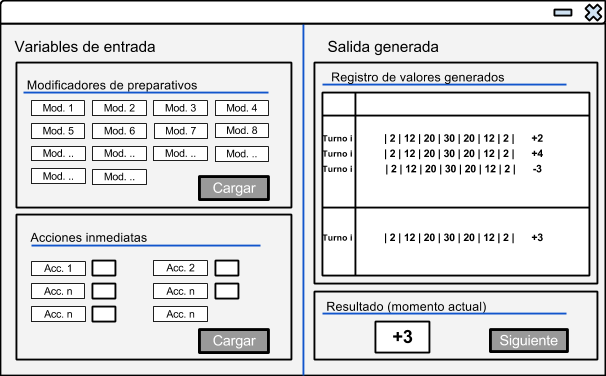
### 2.1.1 Class GraphicInterface

La interfaz gráfica se hará por JSwing. Se busca la funcionalidad, no la estética ya que el programa es exclusivo para el equipo. Recomiendo hacer la interfaz gráfica con NetBeans.

Un detalle de funcionamiento: una vez se carguen los modificadores de preparativos, el panel quedará bloqueado y no se podrán modificar ni cargar nuevas opciones para ningún equipo

##### Definición esquemática de la interfaz gráfica

* Panel contenedor
  + Panel izquierdo
    - Panel de modificadores de preparativos
      * Matriz de selectores, botones o entradas de texto para activar o fijar el valor de todas las acciones preparatorias.
      * Botón para cargar las acciones preparatorias
    - Panel de acciones inmediatas
      * Matriz de selectores, botones o entradas de texto para activar o fijar el valor de las acciones inmediatas
      * Botón para cargar las acciones inmediatas en el siguiente turno
  + Panel derecho
    - Panel registro de valores generados por turno
      * Tabla con los valores y resultado que se obtuvo en cada turno; el turno actual se resaltará sobre los demás
    - Panel de resultado
      * Resultado del turno actual
      * Botón para calcular siguiente turno



### 2.1.2 Class AbilitiesData

Esta clase representará el acceso a una “base de datos” para el programa. No tiene sentido en estos momentos de desarrollo conectar el programa con una base de datos real; así que optaremos por almacenar la información en ficheros estáticos que leeremos y procesaremos en esta clase.   
Todos los ficheros de datos tendrán que ir en un **paquete data** diferente al paquetede código.

La finalidad de esta clase es encapsular al resto de clases el acceso a datos estáticos del estilo <Flag activo en el modificador 1 significa una distribución X de probabilidades>

En el siguiente enlace he dejado indicaciones en Java para hacer el [procesamiento de ficheros](http://pastebin.com/8fr1rdjH). No es obligatorio usar esa forma de procesar ficheros ya que podría haber formas mejores o más eficientes.

### 2.1.3 Class Engine

Será la clase encargada de hacer los cálculos para determinar la tabla de estados y las probabilidades asociadas a los estados. Por lo tanto, usará un objeto de la tabla AbilitesTable para obtener los datos.

Tendrá una estructura de datos para almacenar la tabla de estados/probabilidades. Podría ser un HashMap

Contendrá el método **calculateNextMatch** que generará un número aleatorio y consultará la tabla que previamente habrá generado para determinar el siguiente estado.

### 2.1.4 Class MatchSimu

Será la clase principal, la clase orquesta. Los eventos generados en la interfaz gráfica llamarán a métodos públicos de esta clase.

Usará un objeto **engine** para hacer las operaciones, por ejemplo recibir el resultado de un turno y dárselo a la interfaz para que lo muestre.

Los eventos más importantes son

* El evento “cargar modificadores de preparativos” llamará al método initial Mod para cargar los datos a engine y bloqueará el panel correspondiente.
* El evento “cargar modificaciones inmediatas” llamará al método actualMod para cargar los datos a engine.
* El evento “calcula el próximo partido” llamará al método nextMatch de SimuMatch <<que a su vez hará una llamada a engine.calculateNextmatch()>>