2014

Programación 3

Trabajo Práctico 3, Mundial

Documentación General del trabajo práctico.

Francisco Tobar, Juan Ferreyra, Maximiliano Dahn

6/24/2014

PROGRAMACION 3

INFORME DEL TRABAJO PRACTICO DEL MUNDIAL

CONTENIDO

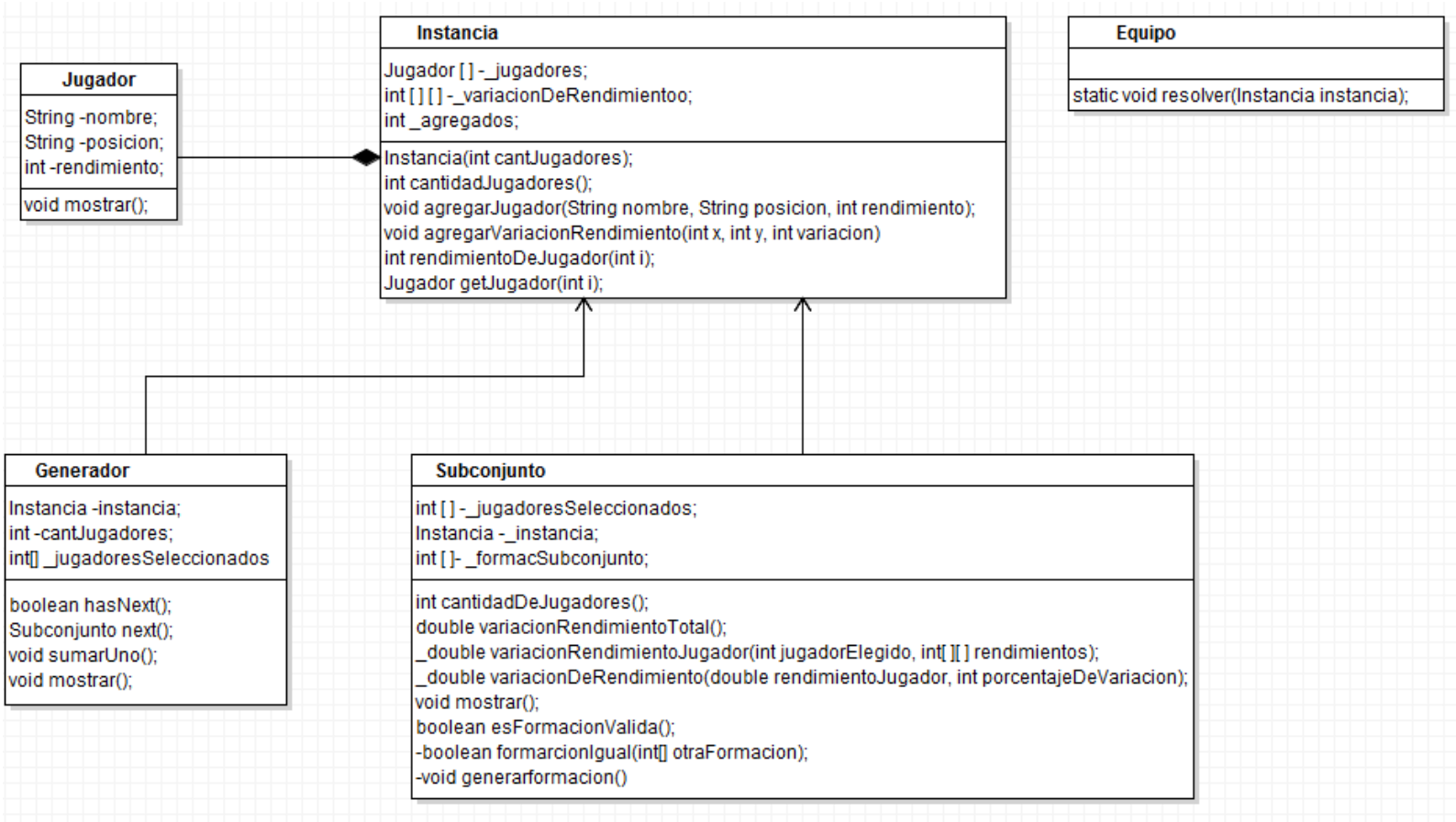
DESCRIPCIÓN GENERAL DEL TRABAJO PRÁCTICO

OBJETIVO GENERAL

Este informe presentará los métodos y funciones del código de un algoritmo que debíamos implementar con fuerza bruta trabajando principalmente sobre el equipo argentino de futbol. Se debía obtener el mejor equipo para el DT del seleccionado argentino teniendo en cuenta que cada jugador tiene un rendimiento personal y un rendimiento dependiendo de con cuál de sus compañeros juega.

DISEÑO

Para realizar el algoritmo el diseño utilizado fue el siguiente con cajas bien definidas.



CLASES

JUGADOR:

- **¿Qué Hace?:**

  La clase jugador es un modelado de lo que los jugadores representan en un partido de futbol. Básicamente se utilizaran para crear un instancia donde se ubicaran los 23 jugadores seleccionados y así poder representar el seleccionado en total sin la elección de los 11 jugadores
  Este objeto tiene como variables locales:

```
1. private String _nombre;
2. private String _posicion;
3. private int _rendimiento;
```

  Representando así un jugador.
  Quizás lo que hay que dejar en claro aquí es que cada jugador tiene un rendimiento personal donde luego en la siguiente clase vamos a describir la función de esta variable. Por otra parte tiene una _posición marcada como String donde pueden ocupar una de las siguientes:

  - AR = arquero
  - DF = defensor
  - MD = mediocampista
  - DL = delantero

  De esta manera se diferencia cada jugador convocado.

INSTANCIA:

- **¿Qué Hace?:**

  La clase instancia es la encargada de representar al equipo de Sabela con una lista contenida de 23 jugadores y una matriz que representa la variación de rendimiento. La variación de rendimiento es aquella matriz donde dependiendo la posición del jugador dentro de la lista con otro, indica como se ve afectado el rendimiento positiva o negativamente para con el equipo. Esto quiere decir que por ejemplo si un jugador se molesta con otro el rendimiento del equipo debería ser peor en función del equipo. Pero en definitiva la instancia representa a los jugadores con su variación de rendimiento.

- **¿Cómo lo hace?:**

  Para crear una instancia primero se debe pasar la cantidad de jugadores debe tener el equipo para crear internamente la matriz y la lista de jugadores. Luego se pueden agregar jugadores y su variación de manera dinámica. Para que se puedan agregar dinámicamente se tuvo que agregar a la clase una variable denominada "_agregados" donde como su nombre lo indica tiene un recuento de los jugadores que se van agregando para así poder agregar un jugador con el índice siguiente luego de haber agregado.
  Luego una función que se utilizara más adelante es la que devuelve el rendimiento individual de un jugador pasándole el índice de la lista.

SUBCONJUNTO:

- **¿Qué Hace?:**

Esta abstracción representa una posible solución al problema que queremos resolver. En su interior contiene una lista binaria donde representa a los jugadores seleccionados en ese instante. En esta clase es la que decide la cantidad de rendimiento del equipo para poder así definir si es una posible solución o no como mencionamos previamente. Además de ello esta clase define la puntuación de rendimiento que tiene el equipo con esta posible solución. Para ello necesita calcular jugador por jugador el rendimiento personal con sus compañeros y sumar todos sus rendimientos para así obtener el rendimiento en equipo.
Por último también se encarga de definir si la formación del subconjunto es válida para un partido de futbol, entonces realizamos una función que chequea esto comparando las posiciones de los jugadores para armar una pequeña lista, en los índices de la lista hay enteros que representan la suma en cantidad de jugadores que juegan en dichas posiciones. Veamos un ejemplo grafico:

Supongamos que el diseño de a continuación es una lista.

Entonces al comparar dos listas, una lista con la formación valida y otra con la del subconjunto, ya nos informa si esta posible solución tiene una formación real.

- **¿Cómo lo hace?:**

Veamos como realizan su trabajo las funciones más importantes de esta clase.

En primer lugar detallaremos las funciones referidas al cálculo de rendimiento total del equipo algún subconjunto postulado.

```
public double variacionRendimientoTotal();
```

Aquí nos encontramos con la función que retorna un double con el rendimiento total del equipo. Pero para poder obtener esto primero se debe obtener el rendimiento personal de cada jugador ya que varía dependiendo de qué compañero tenga en el equipo.

Para ello implementamos otra función denominada

```
variacionRendimientoJugador(int jugadorelegido, int[][] rendimientos);
```

Donde en su interior calcula exactamente la variación que tiene el jugador teniendo en cuenta el porcentaje que suma o resta dependiendo del jugador con el cual se compara.

En segundo lugar fue necesario programar `cantidadDeJugadores();` para determinar la cantidad de jugadores seleccionados de los 23.

Esta función agarra la lista de `_jugadoresSeleccionados` y cuenta cuando encuentra un valor 1 ya que representa un jugador en el equipo. Retorna el contador. Fue necesaria para valida si era un equipo valido dentro de la cancha.

Por último están las funciones que determinan si es válida la formación de tal subconjunto:

Primero comenzamos con un método que lo denominamos

```
public boolean esFormacionValida();
```

Esta función genera la formación del subconjunto y la compara con las formaciones validas de un partido de futbol. Para poder realizar su trabajo utiliza otra función que compara índice por índice de la lista como formación para así concluir si es válida o no lo es.

GENERADOR:

- **¿Qué Hace?:**
  El generador se encarga de pasar cuando se lo solicite un subconjunto siguiente dentro de un límite establecido de datos. Es decir, tiene siempre una lista binaria de jugadores seleccionados (donde los "1" representa los que están y los "0" los que no están) y cuando se lo solicita modifica esa lista de seleccionados haciendo un cambio de un bit en ella para entregar otro posible subconjunto. Para esto implementa funciones como hasNext() que indica si existe un próximo subconjunto a generar; next() donde entrega el próximo subconjunto. De esta manera le es útil a la siguiente clase static que explicaremos a continuación.

EQUIPO:

- **¿Qué Hace?:**
  Esta es la clase encargada de revisar todos los subconjuntos, uno por uno, que el generador le entrega e ir revisando que tenga 11 jugadores (método `cantidadJugadores()` ya mencionado) y si la formación es válida, para que no quede ninguna opción descartada a evaluar. De esta manera siempre se queda con el equipo que tenga mejor rendimiento total y así asegurar que es la mejor solución.

- **¿Cómo lo hace?:**

  Una vez teniendo la estructura anterior esta clase realiza las funciones importantes pareciéndolo simple. Veamos

```
public static void resolver(Instancia instancia)
{
	Generador generador = new Generador (instancia);
	Subconjunto elMejor = null;

	//Mientras halla otro subconjunto por generar
	while(generador.hasNext())
	{
		//Genera un subconjunto nuevo
		Subconjunto actual = generador.next();

		if(actual.cantidadDeJugadores()==11 && actual.esFormacionValida())
		{
			if(elMejor==null ||
	elMejor.variacionRendimientoTotal() < actual.variacionRendimientoTotal())
				elMejor = actual;
		}
	}

	elMejor.mostrar();
}
```

Como vemos primero se debe generar una instancia. Pasársela al Equipo y su función de resolver. Esta internamente crea el generador con la instancia y mientras sea posible entregarle el próximo subconjunto para comparar si tiene 11 jugadores y una formación valida. Si cumple estas condiciones y tiene mejor variación de rendimiento que la previamente almacenada entonces guardamos ese subconjunto como posible solución hasta encontrar una mejor o se convierta en la mejor solución. Básicamente se fija todas las soluciones posibles hasta encontrar la mejor. De eso se tratan los algoritmos de fuerza bruta.