Práctica 11 – Programación con Restricciones – FdI - UCM

Preparación:

1.- Crear un proyecto Java en Eclipse.
2.- Añadir en Build Path el jar externo ***choco-solver-3.3.3-j7-with-dependencies.jar***

Problema: tenemos un columpio tipo "balancín" con 4 posiciones para sentarse en cada lado (que suponemos que están a distancias equidistantes):

Nuestro objetivo es equilibrar el balancín con pesas de distintos valores. Si llamamos a1, a2, a3 y a4 a los pesos que se colocarán en un lado del columpio y b1, b2, b3 y b4 a los que se situarán al otro lado, tenemos que el columpio está en equilibrio si se cumple:

$$1*a1 + 2*a2 + 3*a3 + 4*a4 = 1*b1 + 2*b2 + 3*b3 + 4*b4$$

Modelo

- Declararemos un solver de la siguiente forma:

```
Solver solver = new Solver("Práctica 10");
```

- A continuación dos arrays as y bs usando `VariableFactory.enumeratedArray`
Los pesos en el array as oscilarán entre 0 y 3 (0 significa no se pone peso), mientras que en el lado del array bs se colocarán pesos de entre 2 y 5 kg.
- Debe suceder que a1 != b1, … a4!=b4 (usar `IntConstraintFactory.arithm`)
- Declaramos un array Java coeffs con 4 valores [1,2,3,4]
- Finalmente declaramos dos variables enteras (IntVar) ar y br, ambas entre 0 y 20 con `VariableFactory.bounded`
- Usaremos el constraint **`IntConstraintFactory.scalar`** para asegurar que ar es el producto escalar de as*coeffs es ar, y que bs*coeffs es br.
- Finalmente, un constraint `IntConstraintFactory.arithm` comprobará que ar=br.
- Para resolver usaremos `solver.findSolution()`.
- Para mostrar los resultados ver el apartado "6.1.2 Enumerating solutions" del manual.

```
Sol1.: a0: 1 a1: 3 a2: 3 a3: 1 b0: 2 b1: 2 b2: 2 b3: 2. Peso total: 20
Sol2.: a0: 3 a1: 1 a2: 1 a3: 3 b0: 2 b1: 2 b2: 2 b3: 2. Peso total: 20
```