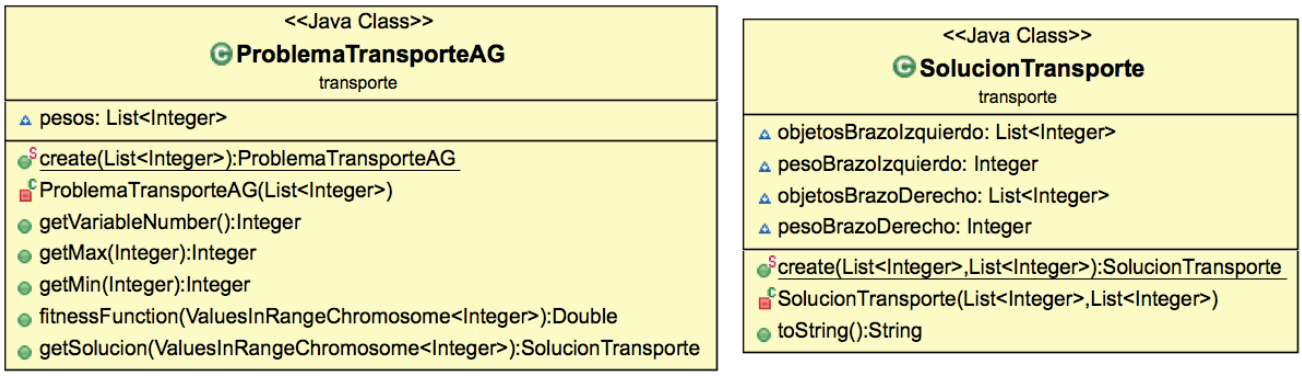
Ejercicio 4 – Algoritmos Genéticos

Se deben transportar n objetos pesados (cada objeto i en [0,n) con peso p_i) en una máquina transportadora que cuenta con 2 brazos. Para ello, cada objeto debe ubicarse en uno y sólo uno de los brazos, de forma que la carga total esté lo más equilibrada posible entre ambos brazos. Se cuenta además con la siguiente restricción: no pueden incluirse en el mismo brazo más de 3n/4 objetos.

Por ejemplo, para pesos = [241, 345, 876, 137, 431, 553] (n = 6), una posible solución sería:

```
Brazo izquierdo (identificado por 0):
      Objetos: [0, 1, 3, 5]
      Peso total: 1276
Brazo derecho (identificado por 1):
      Objetos: [2, 4]
      Peso total: 1307
```

Se desea resolver este problema mediante Algoritmos Genéticos. Para ello, se han modelado las clases ProblemaTransporteAG y SolucionTransporte que se muestran en el siguiente diagrama (*):



(*) el método create de la clase SolucionTransporte crea un objeto solución a partir de:

- un array de ceros y unos de tamaño n (0 indica que ese elemento va en el brazo izquierdo, 1 que va en el brazo derecho), y
- un array de pesos

SE PIDE:

1. Implementar los siguientes métodos de la clase ProblemaTransporteAG:

a) `public Integer getVariableNumber()`
b) `public Double fitnessFunction (ValuesInRangeChromosome<Integer> cr)`
c) `public SolucionTransporte getSolucion(ValuesInRangeChromosome<Integer> cr)`

Solución:

1)

```
@Override
public Integer getVariableNumber() {
	return pesos.size();
}

@Override
public Double fitnessFunction(ValuesInRangeChromosome<Integer> cr) {
	List<Integer> cromosomaDecodificado = cr.decode();
	int n = cromosomaDecodificado.size();
	int i = 0;
	int numObjetosBrazoIzquierdo = 0;
	int numObjetosBrazoDerecho = 0;
	int pesoBrazoDerecho = 0;
	int pesoBrazoIzquierdo = 0;
	while(i<n) {
		if(cromosomaDecodificado.get(i) == 0) {
			numObjetosBrazoIzquierdo++;
			pesoBrazoIzquierdo += pesos.get(i);
		} else {
			numObjetosBrazoDerecho++;
			pesoBrazoDerecho += pesos.get(i);
		}
		i++;
	}

	int diferenciaPesos = Math.abs(pesoBrazoIzquierdo-
pesoBrazoDerecho);
	Double penal = 0.0;
	int umbral = Math.floorDiv(3*n, 4);

	if(numObjetosBrazoDerecho >= umbral)
		penal += numObjetosBrazoDerecho - umbral;
	else if (numObjetosBrazoIzquierdo >= umbral)
		penal += numObjetosBrazoIzquierdo - umbral;

	int pesoTotal =
pesos.stream().mapToInt(Integer::intValue).sum();
	penal = penal*pesoTotal*pesoTotal;

	return (double) -(diferenciaPesos+penal);

}

@Override
public SolucionTransporte getSolucion(ValuesInRangeChromosome<Integer>
cr) {
	List<Integer> sol = cr.decode(); //1 punto
	return SolucionTransporte.create(sol, pesos); // 1 punto
}
```