PROBLEMA 1: Algoritmos Genéticos

Belén cumple 30 años y quiere celebrar una fiesta con un gran número de actividades. Belén dispone de una lista de actividades disponibles y un presupuesto total. Cada actividad tiene información sobre el código, el nombre, el coste que supone, y la valoración que tiene dicha actividad entre los usuarios. Además, existe un conjunto de restricciones de actividades incompatibles, es decir, que no se pueden realizar conjuntamente. Por ejemplo, si contrata la actividad "Charanga", no se puede contratar la actividad "Grupo Flamenco".

El objetivo del problema es encontrar la lista de actividades a contratar (solución al problema) en la que se maximice la valoración. Tenga en cuenta que no se puede pasar del presupuesto disponible y que las actividades deben ser todas compatibles.

Para obtener tal selección de actividades, se pide resolver el problema mediante la técnica de Algoritmos Genéticos. Asuma que los algoritmos genéticos son algoritmos de maximización por defecto.

Ejemplo: Lista de Actividades:

Restricciones:
- Charanga <> Grupo Flamenco
- Fotomatón <> Photocall

<>: Significa que no pueden realizarse en la misma fiesta.

Código	Nombre	Coste(€)	Valoración
0	Charanga	750,0	8
1	Grupo Flamenco	600,0	5
2	Dj	500,0	10
3	Fotomatón	600	10
4	Photocall	200	8
5	Buffet Postre	350	10

Una posible solución con un presupuesto de 2000€ sería:

Valoración 36, Lista Actividades: {Charanga, Dj, Photocall, Buffet Postre}

Una posible solución con un presupuesto de 2500€ sería:

Valoración 38, Lista Actividades: {Charanga, Dj, Fotomatón, Buffet Postre}

SE PIDE

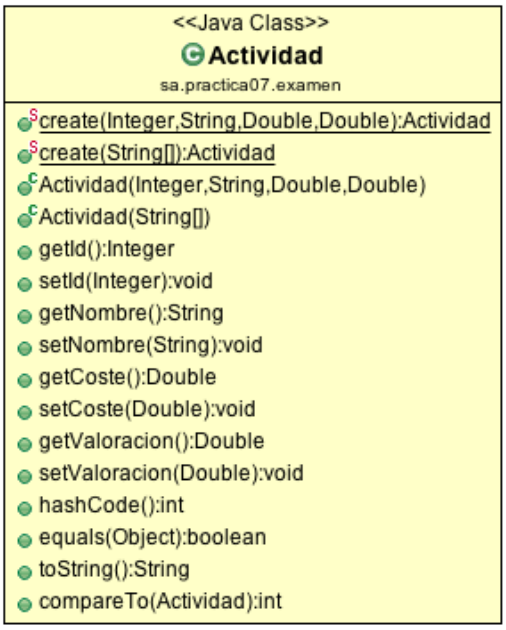
1. ¿Qué tipo o tipos de cromosomas son los más adecuados para resolver el problema y por qué?

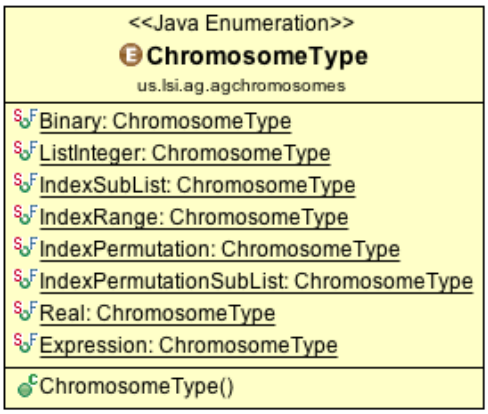
2. Defina y explique cuál sería la función fitness.

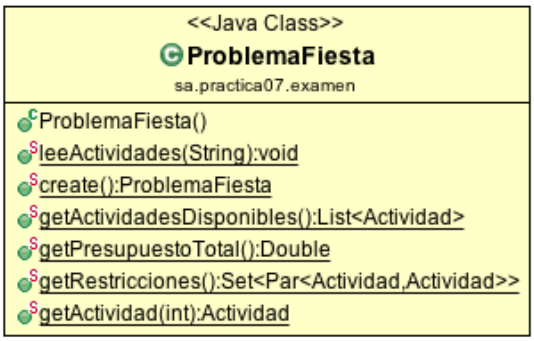
3. Implemente los siguientes métodos pertenecientes a la clase ProblemaFiestaAG para la técnica Algoritmos Genéticos:

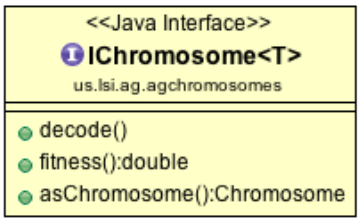
public Double fitnessFunction(… chromosome) { //TODO}

public List<Actividad> getSolucion(… chromosome) {//TODO}









SOLUCIÓN:

1. ¿Qué tipo o tipos de cromosomas son los más adecuados para resolver el problema y por qué?

Respuesta: El cromosoma más adecuado es el Binary. Aunque puede resolverse también mediante los cromosomas *IndexRange* (con delimitación de multiplicidad máxima a 1) o el *IndexSubList*. Cualquiera de ellos define los cromosomas adecuados para resolver el problema, cuya solución es incluir o no una actividad en la solución del problema. En los casos de *Binary* y de *IndexRange*, el valor del cromosoma indica si incluimos la actividad (valor 1) o no (valor 0). En el caso de *IndexSubList* se correspondería con el índice de la actividad.

2. Defina y explique cuál sería la función fitness.

Respuesta: La función fitness a maximizar podría ser la siguiente:

$$valoracionTotal = \sum_{i=0}^{|ls|-1} va(i) \mid dc(i) > 0$$

$$presRest = presupuestoTotal - \sum_{i=0}^{|ls|-1} cost(i) \mid dc(i) > 0$$

$$presRest >= 0$$

$$numActIncomp = \sum_{p \in rest} esIncomp(p)$$

$$Fitness = valoracionTotal - ((presRest>=0?0: presRest^2) + numActIncomp^2)*k$$

$a(i) = ProblemaFiesta.getActividad(i)$

$dc(i) = decode().get(i)$

$va(i) = a_i.getValoracion()$

$cost(i) = a_i.getCoste()$

$rest = ProblemaFiesta.getRestricciones()$

$esIncomp(p) = (a1(i) \&\& a2(i))?1:0$

$a1(p) = (decode(p.p1.id) = 1)$

$a2(p) = (decode(p.p2.id) = 1)$

3. Implemente los siguientes métodos pertenecientes a la clase ProblemaFiestaAG para la técnica Algoritmos Genéticos:

Respuesta:

```
@Override
public Double fitnessFunction(BinaryChromosome2 cr) {
	List<Integer> list = cr.decode();
	Double valoracionTotal = 0.0;
	Double costeTotal = 0.0;
	List<Actividad> actividadesARealizar = Lists.newArrayList();
	Double valoracion = 0.;
	for (int i = 0; i < list.size(); i++) {
		Actividad a = ProblemaFiesta.getActividad(i);
		if (list.get(i) > 0) {
			valoracion += a.getValoracion();
			costeTotal += a.getCoste();
			actividadesARealizar.add(a);
		}
	}
	Integer numActIncomp = 0;
	for(Par<Actividad, Actividad>p:ProblemaFiesta.restricciones){
		if(actividadesARealizar.contains(p.p1) &&
		actividadesARealizar.contains(p.p2)) {
			numActIncomp++;
		}
	}
	Double presRest = ProblemaFiesta.presupuestoTotal-costeTotal;
	presRest = presRest >= 0. ? 0.0 : presRest;
	Double d = numActIncomp * numActIncomp + presRest * presRest;
	Double f = valoracionTotal - 10000000L * d;
	return f;
}

public List<Actividad> getSolucion(IndexChromosome cr) {
	List<Actividad> actividades = Lists.newArrayList();
	List<Integer> ls = cr.decode();
	for(int i= 0;i<ls.size();i++){
		if(ls.get(i)>0){
			actividades.add(ProblemaFiesta.getActividad(i));
		}
	}
	return actividades;
}
```