Ejercicio 3

Se dispone de un fondo de inversión que dispone de una cantidad de dinero M para invertir en acciones. Al invertir la cantidad X en una acción el fondo espera recibir un cierto beneficio o interés. Se desea diseñar un algoritmo que intente repartir la cantidad M entre N acciones distintas de forma que se obtenga el máximo beneficio esperado posible. Para poder resolver el problema se dispone de una función beneficio(i,x) que devuelve el beneficio o interés que obtiene el fondo al invertir la cantidad x en la acción i.

En el caso de que estemos considerando la acción I y nos reste por invertir la cantidad de C. Una posible estrategia voraz de inversión sería invertir la cantidad $X \in \{0, 1, \ldots, C\}$ que haga máximo el ratio beneficio(I,X) / X. Este ratio valdría cero para cualquier cantidad X=0.

Ejemplo: Suponiendo la siguiente tabla de beneficios (ej: el beneficio de invertir en la acción A2 la cantidad de 3 millones es de 6), podemos calcular las siguientes soluciones:

DISPONIBLE		1	2	3	…	M	BENEFICIO
10	A1	**2**	4	5	…	5	2
9	A2	3	4	**6**	…	**6**	8
6	A3	2	3	**7**	…	**7**	15
3	A4	1	3	**5**	…	**5**	20
0							

UNA SOLUCIÓN ÓPTIMA

DISPONIBLE		1	2	3	…	M	BENEFICIO
10	A1	**2**	4	5	…	5	2
9	A2	**3**	4	6	…	**6**	5
8	A3	2	3	**7**	…	**7**	12
5	A4	1	3	**5**	…	**5**	17
2							

SOLUCIÓN VORAZ

Se pide a) rellenar la siguiente ficha para que resuelva el problema descrito utilizando la técnica voraz. Para ello, dispone de los siguientes tipos (notación compacta):

Tipo Acción: $b_i^{k,f}$
i: Integer **// Identifica a la acción**
k: Integer **// Cantidad a invertir en la acción**
f(Integer x): Integer **// Beneficio al invertir la cantidad x en la acción**

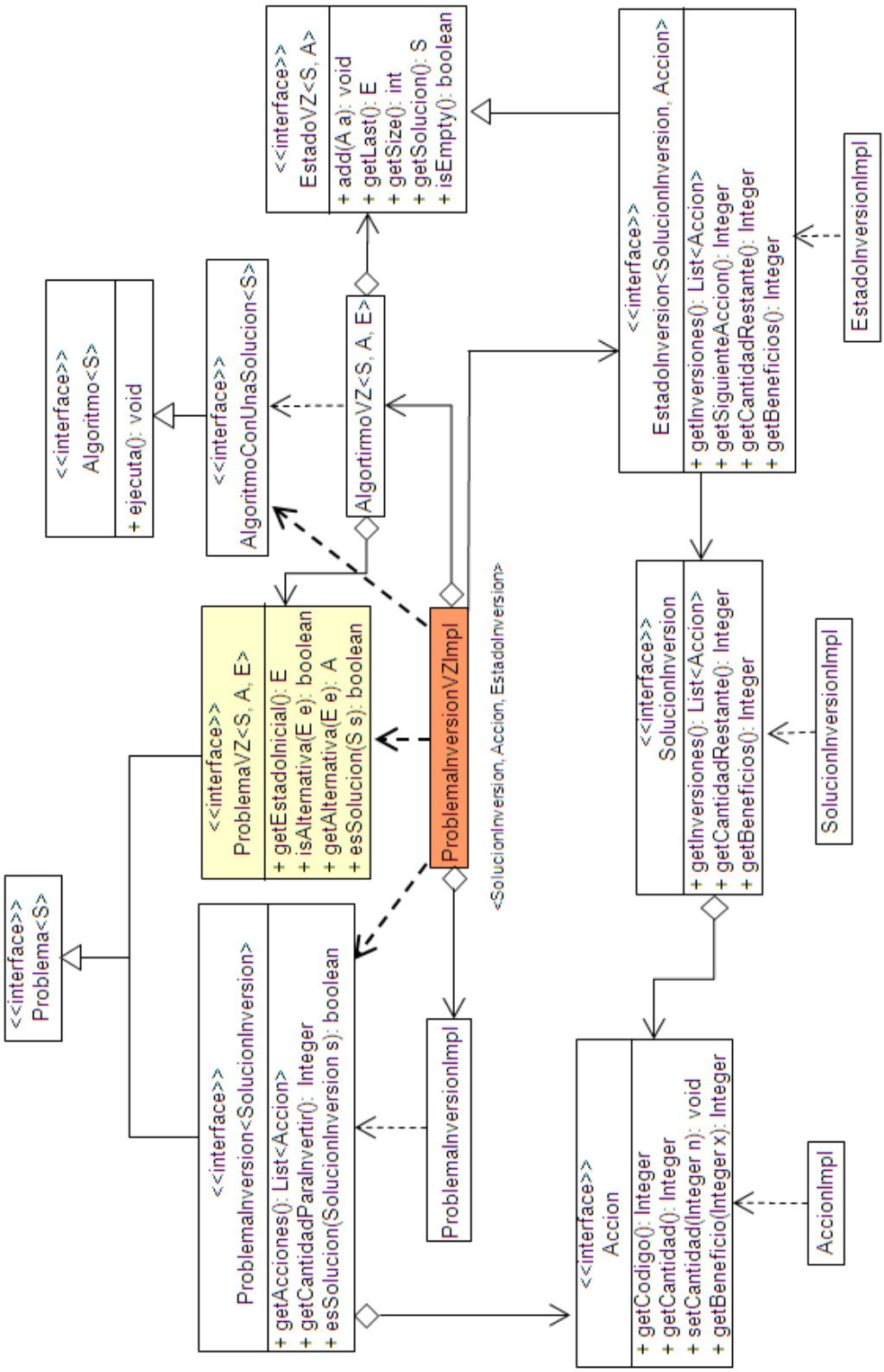
Tipo SolucionInversiones: $s^{lacciones,\ ben}$
lacciones: Lista<Accion> **// Lista de inversiones**
ben: Integer **// Beneficio de la solución**

y de la función $\arg\max_{j \in T}(g(i,j))$ que devuelve el j del conjunto T que hace máximo g(i,j)

Ficha	
Técnica: Algoritmo VZ	
Propiedades Compartidas	N: Integer **// Número de acciones** M: Integer **// Cantidad a invertir**
Propiedades del Estado	L : Lista<Accion> **// Lista de inversiones** B: Integer **// Beneficio total** C: Integer **// Cantidad restante** I: Integer **// ID de la siguiente acción**

Solución: SolucionInversiones	**Objetivo:**
Alternativas:	**Elección:**
Estado inicial:	**Estado final:**
Add(j) :	**Complejidad:**

b) Dado el siguiente esquema UML implementar la clase ProbemaVZImpl, **sólo los métodos de la interfaz ProblemaVZ**. Suponga implementados el resto de clases e interfaces.



Tiempo estimado: 45 minutos **3,5 puntos**

SOLUCION APARTADO A (4 ptos)

Tipo Acción: a_j^k
k: Integer **// Cantidad a invertir en la acción j**

Tipo SolucionAcciones: $s^{lacciones,\ ben}$
lacciones: Lista<Accion> **// Lista de inversiones**
ben: Integer **// Beneficio de la solución**

Ficha		
Técnica: Algoritmo VZ, BT		
Propiedades Compartidas	N: Integer, número de Acción M: Integer, número de millones a invertir	
Propiedades del Estado	L : Lista<Accion>, Lista de inversiones bancarias B: Integer, Beneficio total C: Integer, Cantidad restante I: Integer, Siguiente Acción	
Solución: SolucionAcciones		
Objetivo: encontrar $\mathbf{s^{l,b}}$ tal que b tenga el mayor valor		(0,3 ptos)
Alternativas: A(l, b, i, c) = {0,1,2,...,c}		(0,3 ptos)
Elección: $k = h(A(l,b,i,c)) = \arg\max_{j \in A}(\frac{a_i . f(j)}{j})$		(1 ptos)
Estado inicial: ([], 0, M,0)		(0,4 ptos)
Estado final: i>=N o C<=0		(0,4 ptos)
Add(j) : (l, b, c, i) -> (l + a_i^j, b + a_i.f(k), c – k, i + 1)		(0,8 ptos)
Complejidad: O(N x M)		(0,8 ptos)

SOLUCION APARTADO B (6 ptos)

```
public class ProblemaInversionVZImpl implements
        ProblemaVZ<SolucionInversion, ObjetoInversion, EstadoInversion>
        AlgoritmoConUnaSolucion<SolucionInversion>, ProblemaInversion {

    ProblemaInversion problema;

    public EstadoInversion getEstadoInicial() {  // 1 ptos
        return new EstadoInversionImpl(problema.getCantidadParaInvertir());
    }


    public boolean isAlternativa(EstadoInversion e) {     // 1,5 ptos
        return e.getSiguienteAccion() < problema.getAcciones().size() &&
            e.getCantidadRestante()>0;
    }

    public Accion getAlternativa(EstadoInversion e) {     // 2,5 ptos
        Accion a = problema.getAcciones().get(e.getSiguienteAccion());
        int crMax = 0;
        int rMax = 0;
        for (int cr=1;cr<=e.getCantidadRestante();cr++) {
           r = a.getBenficio(cr)/cr;
           if (rMax<r) {
             crMax = cr;
             rMax = r;
           }
        }
        a.setCantidad(crMax);
        return a;
    }

    public boolean esSolucion(SolucionMochila s) {   // 1 ptos
        return problema.esSolucion(s);
    }

   ...
}
```