Examen de Algorítmica y Complejidad (Plan 2014) — **2 de diciembre de 2022**

Nº matrícula:_____ **Nombre:** _____

Apellidos: _____

1) **Problema** (**2.5 puntos**). El avión en el que viajabas ha sufrido un terrible accidente y se ha estrellado en las aguas cercanas a una isla desierta. Tú y un reducido grupo de supervivientes habéis conseguido superar la peor parte, pero ahora estáis preocupados por sobrevivir al hambre. Por suerte el avión transportaba alimentos para los pasajeros y tripulantes en varios contenedores, por lo que **necesitáis recuperar todo lo posible antes de que el avión se hunda**. El valor de cada contenedor puede calcularse en función de los días que permite subsistir (debido a la cantidad de comida que almacena) y el tiempo que se tarda en extraerlo del avión (por su tamaño o lugar de almacenaje). Como único experto en algorítmica del grupo de supervivientes, has sido elegido para gestionar el rescate de contenedores.

   SE PIDE: Implementar un algoritmo basado en el **esquema Voraz** que maximice el tiempo que podréis sobrevivir en la isla gracias a los días de subsistencia que os darán los contenedores de alimentos que seáis capaces de rescatar antes de que se hunda el avión. El algoritmo deberá tener la siguiente cabecera:

```
ArrayList<Contenedor> rescate(ArrayList<Contenedor> contenedores, int hundir)
```

   donde *contenedores* es una lista de los contenedores de alimentos que trasporta el avión y *hundir* son los minutos que tardará el avión en sumergirse, a partir de los cuales no podréis recuperar más contenedores. El método devolverá los contenedores rescatados.

   Aclaraciones: Se podrán implementar todos los métodos y clases adicionales que se consideren necesarios, incluyendo añadir a la clase Contenedor.

```
class Contenedor{
    private int tiempo; // tiempo en minutos que se tarda en recuperar el contenedor
    private int dias;   //número de días que proporciona alimento

    public Contenedor(int tiempo, int dias){
        this.tiempo = tiempo;
        this.dias = dias;
    }
    // Métodos get/set de las propiedades de la clase

    public float getBeneficio(){
        return ((float) this.dias) / ((float) this.tiempo);
    }
}
```

```
public static ArrayList<Contenedor> rescate(ArrayList<Contenedor> contenedores, int
hundir){
```

```
        ArrayList<Contenedor> solucion = new ArrayList<Contenedor>();
        int tRestante = hundir;

        while(!recursos.isEmpty() && tRestante > 0){
            Contenedor seleccionado = seleccion(contenedores);
            contenedores.remove(seleccionado);
            if(seleccionado.getTiempo() <= tRestante){
                tRestante -= seleccionado.getTiempo();
                solucion.add(seleccionado);
            }
        }

        if(solucion.size()>0)
            return solucion;
        else
            return null;

    }

private static Contenedor seleccion(ArrayList<Contenedor> contenedores) {
        Contenedor mejor = null;
        for(Contenedor actual : contenedores){
            if(mejor == null) mejor = actual;
            else if(mejor.getBeneficio() < actual.getBeneficio()){
                mejor = actual;
            }
        }
        return mejor;
    }
```