Examen de Algorítmica y Complejidad (Plan 2014)

18 de enero de 2022

Nº matrícula:_____ **Nombre:** _____

Apellidos: _____

Problema. El I.E.S. Almudena Grandes necesita crear los nuevos grupos de alumnos de 1º de la ESO. Para ello debe distribuir un grupo desordenado de **N** nuevos estudiantes, que provienen de diferentes centros escolares de la zona, en grupos de un tamaño máximo de **L** alumnos/grupo (*numMaxAlumnos*). De cada estudiante se dispone del nombre y apellidos, el género (femenino/masculino) y su nota media (expresada como un número entero entre el 5 y el 10). Se desea implementar un algoritmo, **basado en una estrategia voraz**, que permita crear los nuevos grupos de alumnos, de forma que tengan una nota media de grupo lo más parecida posible.

Implementar el método *MezclaEstudiantes* así como los métodos auxiliares que se consideren necesarios.

```
ArrayList<Grupo> mezclaEstudiantes(ArrayList<Estudiante> listaEst,
                                   int numMaxAlumnos)
```

```
public class Estudiante {
    private String nombre, apellidos; //nombre apellidos alumno
    private double nota; //nota media de primaria
    ...
    //constructor, getters, setters
}
public class Grupo {
    // Un Grupo es un conjunto de N estudiantes
    private ArrayList<Estudiante> alumnos;
    private int maxAlumnos;
    public Grupo(int maxAlumnos){
       alumnos = new ArrayList<Estudiante>();    this.maxAlumnos=maxAlumnos;
    }
    public Grupo(ArrayList<Estudiante> lista, int maxAlumnos){
       alumnos = new ArrayList<Estudiante>(lista);   this.maxAlumnos=maxAlumnos;
    }
    //constructores, getters, setters
    public int getAlumnosRestantes(){ return maxAlumnos-alumnos.size();}
    public void aniadeAlumno(Estudiante e){ alumnos.add(e);}
}
```

```
ArrayList<Grupo> mezclaEstudiantes(ArrayList<Estudiante> listaEst, int maxAlumnos) {
     boolean mezclar = true; //alterno estudiantes buenos = true/malos=false
     ArrayList<Grupo> solucion = new ArrayList<Grupo>();
     Estudiante est;
     if (maxAlumnos >= listaEst.size()) { //Todos los alumnos entran en un grupo
         solucion.add(new Grupo(listaEst, maxAlumnos));
     } else { //hay que crear g.size()/maxAlumnos grupos
         while (!listaEst.isEmpty()){
             if (mezclar)  est = seleccionarCandidatoMejor(listaEst);
             else          est = seleccionarCandidatoPeor(listaEst);
             listaEst.remove(est); //borro el estudiante
             aniadirCandidato(est, solucion, maxAlumnos);
             mezclar = !mezclar; //cambio
         }
     }
   return solucion;
}
```

```
void aniadirCandidato(Estudiante est, ArrayList<Grupo> grupos, int maxAlumnos){
   boolean encontrado = false;
   Grupo g;
   if (grupos.size()>0){
      g = grupos.get(grupos.size()-1);
      if (g.getAlumnosRestantes() > 0){
         //el estudiante cabe en el último grupo
         encontrado=true;
         g.aniadeAlumno(est);
      }
   }
   // Si todavía no hay grupos o no caben más alumnos en el último grupo, entonces
   // creamos otro grupo, añadimos al alumno al grupo y añadimos el grupo a grupos
   if (!encontrado){ // creamos un nuevo grupo
       g = new Grupo(maxAlumnos);
       g.aniadeAlumno(est); //añadimos el estudiante
       grupos.add(g);
   }
}

Estudiante seleccionarCandidatoMejor(ArrayList<Estudiante> candidatos){
   // selecciona el estudiante que MEJOR NOTA tenga
   Estudiante mejor = candidatos.get(0);
   for (int i=1; i<candidatos.size();i++)
       if (mejor.getNota() < candidatos.get(i).getNota())
           mejor = candidatos.get(i);
   return mejor;
}

Estudiante seleccionarCandidatoPeor(ArrayList<Estudiante> candidatos){
   // selecciona el estudiante que PEOR NOTA tenga
   Estudiante peor = candidatos.get(0);
   for (int i=1; i<candidatos.size();i++)
      if (peor.getNota() > candidatos.get(i).getNota())
           peor = candidatos.get(i);
   return peor;
}
```