UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

GRUPO:B2

PRÁCTICA 3. Algoritmos Voraces. Star Wars: Reparto de agua

Trabajo realizado por (grado de participación):

Miguel de las Heras Fuentes → 50%

Isaac González del Pozo → 50%

Asignatura: Metodología de la Programación

Grupo: B2

Titulación: Grado en Ingeniería Informática

Fecha: 03/05/2021

Tabla de contenido

1. Documentación sobre el código

Para nuestro proyecto de algoritmos voraces utilizaremos dos clases: por un lado, tendremos la clase Eopie que será la encargada de transportar los contenedores de agua y por tanto se le asignará un contenedor; y la clase contenedor que tendrá un único atributo que será el volumen en litros que transporta.

1.1 Clase eopie

La clase Eopie estará definida por la capacidad de litros que estos pueden soportar y el contenedor que transportaran durante las noches de la simulación. Esta clase nos servirá en la simulación para saber la cantidad que transporta los Eopies cada noche. Dentro de esta clase tendremos los Getters y Setters necesarios para el programa principal y un toString para imprimir el Eopie con su capacidad transportada.

```
package voraz;

public class Eopie {
	private double capacidad;
	private Contenedor contenedor;

	public Eopie(double capacidad) {
		this.capacidad=capacidad;

	}

	public double getCapacidad() {
		return this.capacidad;
	}
	public void setCapacidad(double capacidad) {
		this.capacidad=capacidad;
	}

	public Contenedor getContenedor() {
		return contenedor;
	}

	public void setContenedor(Contenedor contenedor) {
		this.contenedor = contenedor;
	}


	public String toString() {
		return "Eopie [capacidad=" + capacidad + ", contenedor=" + contenedor + "]";
	}



}
```

1.2 Clase contenedor

La clase Eopie estará definida por el volumen que este transporta(en litros). Esta clase constará de los métodos Getters y Setters y un toString.

```
package voraz;


public class Contenedor{
	private double volumen;

	public Contenedor(double volumen) {
		this.volumen=volumen;
	}

	public double getVolumen() {
		return this.volumen;
	}
	public void setVolumen(double volumen) {
		this.volumen=volumen;
	}

	@Override
	public String toString() {
		return "Contenedor [volumen=" + volumen + "]";
	}



}
```

1.3 Programa principal

Nuestro programa principal simulará el ejercicio proporcionado en el enunciado, en el cual dispondremos de N Eopies y M contenedores seleccionados por el usuario, siempre y cuando el número de Eopies sea menor que el número de contenedores (N<M). Como podemos observar en la siguiente imagen, al principio de dicho programa inicializaremos todas las variables que necesitaremos durante la simulación, como pueden ser la cantidad de litros que se han transportado(“litros”), la cantidad de litros que se deben transportar(“litrosInic”),…

Una vez se introducen por teclado la cantidad de Eopies y contenedores que quiere el usuario para toda esa semana, el sistema imprimirá por pantalla el número total de Eopies y contenedores que se van a utilizar. Posteriormente se inicializarán dos ArrayList para los Eopies y contenedores respectivamente. Previo a la simulación del reparto, se ordenará la lista de contenedores para que se puedan repartir entre los Eopies que se disponen.

Por último, se iniciará la simulación del proyecto haciendo uso del método de algoritmo voraz que explicaremos a continuación, simulando así el trascurso de las 7 noches y concatenando el número de litros que se transportan cada noche, para así imprimir el número total de litros transportados en la semana.

```
package voraz;

import java.util.*;


public class main {

	static Scanner teclado= new Scanner(System.in);

	public static void main(String[] args) {

		double litros=0; /*cantidad de litros transportados en la semana*/
		double litrosInic=0; /*cantidad de litros inicial*/
		int noches=7;
		int numEopies;
		int numContenedores;

		System.out.println("Seleccione el número de eopies");
		numEopies=teclado.nextInt();
		System.out.printf("Seleccione el número de contenedores. Debe ser mayor de %s\n", numEopies);
		do {
			numContenedores= teclado.nextInt();
			if (numContenedores<=numEopies) {
				System.out.printf("Error. El número de contenedores debe ser mayor que el de eopies"
						+ "(%s)\n", numEopies);
			}
		} while (numContenedores<=numEopies);

		System.out.printf("El numero total de eopies es %s y el número total de contenedores es %s\n"
				,numEopies, numContenedores);

		//creamos las listas
		ArrayList <Eopie>eopies= new ArrayList<Eopie>();
		ArrayList<Contenedor> contenedores=new ArrayList<Contenedor>();

		//inicializamos la lista de contenedores
		litrosInic=inicContenedores(contenedores, numContenedores);
		ordContenedores(contenedores);
		System.out.printf("La cantidad total de litros que se debe transportar es: %.2f\n",litrosInic);

		//iniciamos simulacion
		for (int i = 0; i < noches; i++) {
			System.out.printf("%sº noche: \n", i+1);
			litros+=simulacion(eopies, contenedores, numEopies, numContenedores);
		}

		System.out.printf("El numero total de litros transportados en las 7 noches es %.2f \n"
				,litros);


	}
```

En el método main haremos uso de los siguientes métodos:

```
,
//inicializador de eopies
public static void inicEopies(ArrayList<Eopie>eopies,int numEopies) {
        double capacidad;
        Eopie e;
        System.out.println("Lista de Eopies:");
        for (int i = 0; i < numEopies; i++) {
                capacidad=Math.random()*50;
                e= new Eopie(capacidad);
                eopies.add(e);
                System.out.printf("%sº Eopie: capacidad= %.2f litros\n",i+1,capacidad);
        }

}

//inicializador de contenedores
public static double inicContenedores(ArrayList<Contenedor> contenedores, int numContenedores) {
        double capacidad;
        double litrosInic=0;
        Contenedor c;
        System.out.println("Lista de contenedores:");
        for (int i = 0; i < numContenedores; i++) {
                capacidad= Math.random()*50;
                litrosInic+=capacidad;
                c= new Contenedor(capacidad);
                contenedores.add(c);
                System.out.printf("%s. volumen= %.2f litros\n",i+1, capacidad);
        }
        return litrosInic;
}

//métodos para ordenar la lista de eopies y de contenedores
public static void ordEopies( ArrayList<Eopie> eopies) {

        eopies.sort(Comparator.comparing(Eopie::getCapacidad).reversed());

}
public static void ordContenedores( ArrayList<Contenedor> contenedores) {

        contenedores.sort(Comparator.comparing(Contenedor::getVolumen).reversed());

}
```

- El primer método **inicEopies** nos servirá para inicializar la lista de Eopies con las capacidades que estos pueden transportar.

- El método **inicContenedores** servirá de igual modo que el anterior, inicializando la lista de contenedores con su respectivo volumen medido en litros. Este método retornará un numero double que será la cantidad de litros que se deben transportar durante la semana.

- Tanto el método de **ordEopies** como el de **ordContenedores** nos servirán para ordenar de mayor a menor capacidad nuestra lista de Eopies y contenedores.

2. Elementos principales del enfoque voraz seguido

Se tienen **N Eopies** con una capacidad de carga C y **M contenedores** con un peso en litros L.

La estrategia que seguiremos será asignarle al Eopie de mayor capacidad disponible, el contenedor de mayor peso.

Candidatos:

- **A estudiar**: Los M contenedores
- **Seleccionados**: Los contenedores a los que ya se les ha asignado un Eopie

Funciones:

- **Selección**: Contenedor de mayor volumen; si hay dos iguales, cogemos el que se lea primero
- **Factibilidad**: El volumen del contenedor elegido no puede superar la capacidad de carga del Eopie elegido
- **Objetivo**: Maximizar el número de litros transportados
- **Solución**: Si se han completado ya los N Eopies o ya se han asignado todos los contenedores.
- **Inserción**: Asignar un contenedor a un Eopie.

3. Explicación breve de la estrategia seguida para distribuir los contenedores entre los Eopies

La estrategia seguida para distribuir los contenedores se basa en la utilización de dos métodos; el método “simulación” y el método “voraz”, los cuales explicaremos a continuación:

```
public static double simulacion(ArrayList<Eopie> eopies, ArrayList<Contenedor> contenedores,
                        int numEopies) {

                double litros=0;
                eopies= new ArrayList<Eopie>();

                inicEopies(eopies, numEopies);
                ordEopies(eopies);
                Iterator <Eopie>it= eopies.iterator();
                Eopie e;
                System.out.println("Lista de eopies que no han transportado ningun litro esta noche:");
                while (it.hasNext()) {
                        e= it.next();
                        e.setContenedor(voraz(e, contenedores));

                        if (e.getContenedor()!=null) {
                                litros+=e.getContenedor().getVolumen();
                        }else {
                                System.out.printf("Eopie: capacidad= %.2f litros\n", e.getCapacidad());
                        }
                }

                return litros;
        }


public static Contenedor voraz(Eopie e, ArrayList <Contenedor> contenedores) {
                Contenedor S=null;
                Contenedor aux;

                Iterator <Contenedor> it= contenedores.iterator();

                while(S==null && it.hasNext()) {
                        aux= it.next();
                        if (aux.getVolumen()<=e.getCapacidad()) {
                                S=aux;
                                contenedores.remove(aux);
                }
                }
                return S;
        }
```

La primera imagen corresponde a un método que simula una de las noches y por tanto será llamado desde el método main siete veces; una por noche.
El método consiste en recorrer la lista de Eopies con un iterador, y en cada iteración llamar al método voraz para asignarle un contenedor.
En caso de que después de esa llamada el eopie siga teniendo el atributo “contenedor”=NULL, mostraremos por pantalla la información de ese eopie para mostrar que no ha podido transportar ningún contenedor esa noche.

La segunda imagen corresponde al método voraz. A este método se le pasará como parámetros el eopie seleccionado en el método anterior, y la lista de contendores disponibles.
El método consistirá en recorrer la lista de contenedores y asignarle al eopie el primer contenedor que encontremos con un volumen menor que la capacidad que puede soportar el eopie.

4. Complejidad teórica del algoritmo

Como hemos dicho en el apartado anterior, nuestro algoritmo voraz consta de dos métodos. Ambos métodos encuentran su mayor complejidad en sus respectivos bucles while. El primer método tendrá una complejidad de $O(n)$, siendo n el número de Eopies. Dentro de su bucle while se hará la llamada al otro método que tendrá otro bucle while que se ejecutará M veces, siendo M el número de contenedores disponibles. Este método tendrá, por tanto, una complejidad de $O(m)$.

Por lo tanto, la complejidad de nuestro algoritmo voraz será:

$$\boldsymbol{O(n * m)}$$

5. ¿Es el algoritmo óptimo en cuanto al resultado obtenido?

No es un algoritmo óptimo porque o bien se desperdicia mucho espacio en los Eopies por la condición de que solo pueden llevar un contenedor o bien porque al no poder fragmentar los contenedores, algunos Eopies no podrán transportar ningún contenedor porque todos tienen un volumen mayor del que puede transportar.