Universidade Vigo
Campus de Ourense
Escola Superior de Enxeñaría Informática

Campus de Ourense
32004 Ourense
España

Tel. 988 387 000
Fax 988 387 001

esei.uvigo.es
sdeuig@uvigo.es

3'25

AED II

Prueba de map y grafo (02/12/2020)

Apellidos: Vila Fonseca Nome: Sergio

DNI: 53979995-K

1.- (2 *puntos*) Dada la clase HashMap<K,V> que implementa la interfaz Map<K,V> (disponible en el anexo):

```
public class HashMap<K,V> implements Map<K,V> {
    private Lista<Par<K,V>> [] tabla;
    private int numElems;
```

implementa en dicha clase el método **modificarValor (K clave, V valorViejo, V valorNuevo)** haciendo uso de la representación o estructura de datos anterior. Si para resolver el ejercicio haces uso de algún otro método de Map o HahsMap, impleméntalo también.

```
public boolean modificarValor (K clave, V valorViejo, V valorNuevo)
//Modifica:this.
```

//*Produce*: Sólo si la clave tiene asociado el *valorViejo* éste se remplaza por el *valorNuevo*. Si se realiza el cambio, el método devuelve cierto. En otro caso, devuelve falso.

3'25

2.- (*4 puntos*) En la teoría de grafos, se conoce como "Lema del apretón de manos" la siguiente propiedad: la suma de los grados de los vértices de un grafo *g* es el doble del número de arcos de *g* (siendo grado de un vértice el número de adyacentes y predecesores que tiene). Escribe un método que compruebe esta propiedad

```
public static <E,T> boolean lemaApretonManos (Grafo<E,T> g)
```

//*Produce*: devuelve cierto si la suma de los grados de los vértices es igual al ~~número~~ doble de arcos del grafo. Devuelve falso en caso contrario

3.- (*4 puntos*) Escoge uno de los siguientes ejercicios de uso de Map.

a) Implementa el método eliminarVertice (Vertice<E> v) de la interfaz Grafo<E,T>, cuya sintaxis y semántica se presenta a continuación:

public void eliminarVertice (Vertice<E> v)

//Modifica: this

//Produce: elimina el vértice del grafo, así como los arcos que llegan y salen de él.

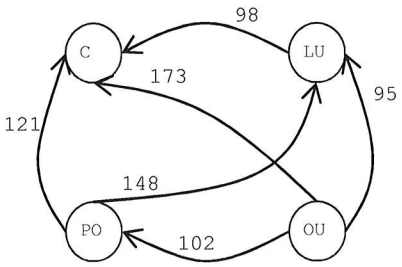
haciendo uso de la siguiente representación interna:

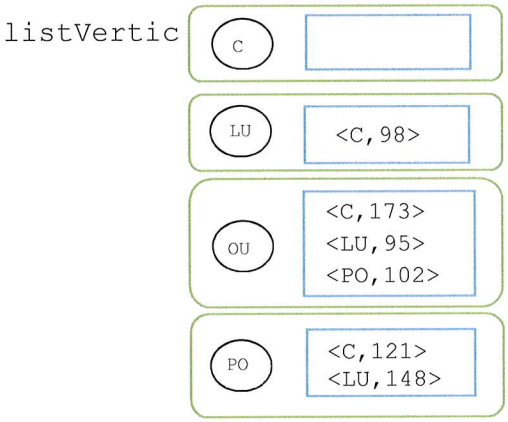
public class MapAdyacencia<E,T> **implements** Grafo<E,T>{

private List<VerticeConMap<E,T>> listVertices;

donde la clase VerticeConMap<E,T> se describe en el anexo.

Ejemplo:





b) Dada la clase MapPolinomio, que implementa la interfaz Polinomio (disponible en el anexo):

public class MapPolinomio implements Polinomio {

private Map<Integer, Double> terminos;

implementa en dicha clase el método resta de polinomios

public Polinomio resta (Polinomio p)

//Produce: un polinomio nuevo que es la diferencia de this - p

Si para resolver el ejercicio haces uso de algún otro método de Polinomio o MapPolinomio, impleméntalo también.

Sergio Vila Fonseca

1.-
```
public boolean modificarValor(Kclave, VvalorViejo, VvalorNuevo){
boolean toRet = false;
while ( getClaves().hashnext() && !VvalorViejo.equals(VvalorNuevo)){
{ K a = getClaves().next();
    if(a.equals(clave)){
        insertar (clave, valorNuevo);
        valorViejo = valorNuevo;
        return true;
    }
}
return toRet;
}
```

¿qué hace?

¿qué hace?

No mas este parámetro.

No mas la estructura de datos que se indica en el enunciado

2.-
```
public static <E,T> boolean lemaApretonManos (Grafo<E,T> g){
Iterator <Vertice<E>> tr = g.vertices();
Iterator <Vertice<E>> itr = g.adyacentes();
int cont=0;
While (tr.hashnext()) {
    Vertice <E> w = tr.next();
    while(xx)
    cont += w.nAdy
    cont += g.nAdy(w);
    cont += g.nPredec(w);
}
int nArcos = int numArcos = g.nArcos();
if ( cont == 2* numArcos)
    return true;
else
return false;
}}
private int nArcos (Grafo<E,T> g ) {
    Iterator <Arco<E,T>> tr = g.arcos();
    int cont = 0;
    while (tr.hashnext){
        cont ++; tr.next();
    }
    return cont; }
```

-0'25/

-0'5/

Sergio Vila Fonseca

2.-

```
public Iterator <Vertice <E>> predecesores (Vertice <E> v){
    ArrayList <Vertice <E>> toRet = new ArrayList ();
    Iterator <Vertice <E>> tr = g.vertices ();

    while (tr.hashnext()){
    Vertice<E>w = tr.next();
        Iterator <Vertice <E>> itr = g.adyacentes (w);
        while (itr.hashnext) {
            Vertice <
            if( v.equals(itr.next()) {
                toRet.add
            Vertice <E> aux = itr.next();
            if(v.equals (aux)) {
                toRet.add(aux);
            }
        }
    }
    Iterator <Vertice <E>> Iterator_toRet = toRet.Iterator();
    return Iterator_toRet;
}

private int nAdy ( Vertice <E> v){
    Iterator <Vertice <E>> tr = g.adyacentes (v);
    int cont=0;
    while( tr.hashnext()) {
        cont++; → itr.next();
    }
return cont;
}
private int nPredec (Vertice <E> v) {
    Iterator <Vertice <E>> tr = predecesores (v);
    int cont=0;
    while(tr.hashnext()){
        cont++; → tr.next()
    }return cont; }
```

Grafo<E,T> g)

Grafo <E,T> g

Grafo<E,T>g)

3.- b)

```
public Polinomio resta (Polinomio p) {   // this - p
int aux = grado();
ArrayList<Par<K,V>> toRet = new ArrayList();
while (aux >= 0) {
if(get(aux) != null && p.get(aux) != null) {
   Par<K,V> a = new Par(aux, (get(aux) - p.get(aux)));
   toRet.add(a);
}
if(get(aux) != null && p.get(aux) == null) {
   Par<K,V> a = new Par(aux, get(aux));
   toRet.add(a);
}
if(get(aux) == null && p.get(aux) != null){
   Par<K,V> a = new Par(aux, (-p.get(aux));
   toRet.add(a);
}
if(get(aux) == null && p.get(aux) == null)
aux--;
}
return toRet;
}
```

método de mapa, lo tendría que llamar un objeto

??? implementarlo (Este ejercicio es de uso de Repo)