CC30A– Control Nº 1 – viernes 23 de septiembre de 2005

Tpo: 1 hr 45 mins – CON apuntes – SIN Consultas – Contestar en hojas separadas (con nombre)

1.La siguiente clase define los métodos para trabajar con una tabla de objetos, dispuestos en filas y columnas (ambas numeradas de 1 en adelante).

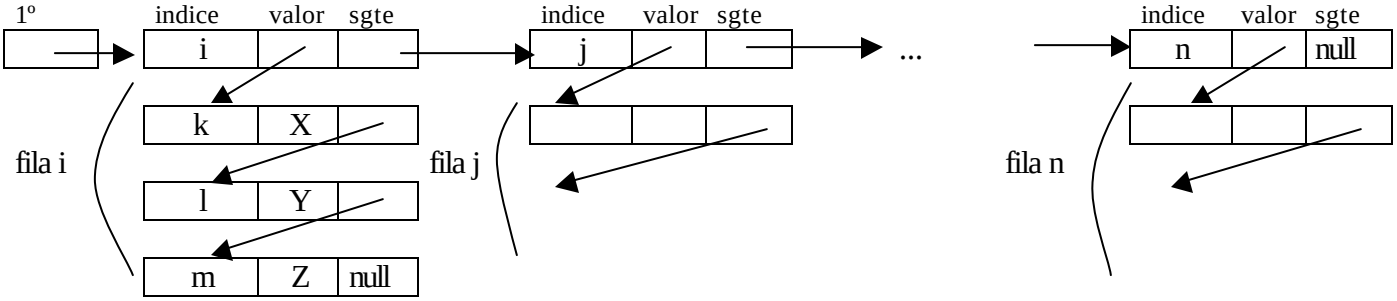
ejemplo	**significado**
Tabla( )	inicializar una tabla que admita cualquier número de filas y columnas
Object getValor (int i,int j)	Entregar el valor del elemento ubicado en la fila i y columna j (si no existe entregar el valor null)
void setValor (Object x,int i,int j) throws SinEspacio	Asignar el valor x al elemento ubicado en la fila i y columna j (si no existe, asignarlo por primera vez)

A)(1 punto) Utilice la clase en un método, de encabezamiento **Tabla crearTabla ()**, que construya y entregue como resultado (sin mostrar en pantalla) la siguiente tabla:

	1	2	3	...	21
1		1	2	...	20
2	A	X			
3	B		X		
...	...			...	
27	Z				X

```
static Tabla crearTabla()throws Exception{

//crear objeto: 0.1
Tabla t=new Tabla();

//primera fila: 0.2
for(int j=2; j<=21; ++j)
    t.setValor(""+(j-1), 1,j);

//primera columna: 0.3
for(int i=2; i<=27; ++i)
    t.setValor("ABCDEFGHIJKLMNOPQRSTUVWXYZ".substring(i-2,i-1),i,1);

//"diagonal": 0.3
t.setValor("X",2,2);
t.setValor("X",3,3);
t.setValor("X",27,21);

//devolver tabla: 0.1
return t;
}
```

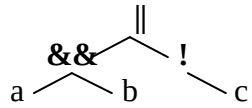
B) (5 ptos) Escriba completamente la clase Tabla considerando que los objetos de la clase se representan a través de una estructura que sólo contiene nodos para los valores presentes en la tabla, en la siguiente forma.



Notas. Las filas y columnas están en orden creciente de índice. Por ejemplo, en la figura se supone que i<j<...<n y k<l<m.

```
class Nodo{  //0.5
int indice; Object valor; Nodo sgte;
Nodo(int x,Object y,Nodo z){
  indice=x; valor=y; sgte=z;
}}
//declaraciones: 0.5
class SinEspacio extends Exception{}
class Tabla{
private Nodo primero;
public Tabla(){primero=null;}
//getValor: 2.0
public Object getValor(int i, int j){ //1 punto
  Nodo r = refIndice(i,primero);
  if(r==null) return null;
  r = refIndice(j,(Nodo)r.valor);
  return r==null ? null : r.valor;
}
protected Nodo refIndice(int x,Nodo r){//1 punto (buscar indice en lista)
  if(r==null) return null;
  if(r.indice==x) return r;
  return r.indice<x ? refIndice(x,r.sgte) : null;
}
//setValor: 2.0
public void setValor(Object x,int i, int j)throws SinEspacio{
  //agregar fila i: 1 punto
  Nodo ri=refIndice(i,primero), r;
  if(ri==null)
    if(primero==null || primero.indice>i)
       ri=primero=new Nodo(i,null,primero);
    else{
       for(r=primero; r.sgte!=null; r=r.sgte)
         if(r.sgte.indice>i) break;
       ri=r.sgte=new Nodo(i,null,r.sgte);
    }
  //agregar x en columna j de fila i: 1 punto
  Nodo p=(Nodo)ri.valor;
  if(p==null || p.indice>j){ri.valor=new Nodo(j,x,p); return;}
  if(p.indice==j){ p.valor=x; return;}
  for(r=p; r.sgte!=null; r=r.sgte){
    if(r.sgte.indice==j){r.sgte.valor=x; return;}
    if(r.sgte.indice>j) break;
  }
  r.sgte=new Nodo(j,x,r.sgte);
}
}
```

2. Una condición se puede representar en un árbol binario. Por ejemplo, suponiendo que a, b y c son valores de tipo boolean, la expresión (a && b) | | (!c), se representa por el árbol:



Al respecto, la clase Cond ofrece los siguientes métodos:

Ejemplos.	resultado	encabezamiento
Cond a=new Cond(true); Cond b=new Cond("condición"); Cond c=new Cond(b);	a: referencia a objeto con valor verdadero b: ref a objeto que contiene condición c: ref a objeto que es una copia de b	Cond(boolean x) Cond(String x) Cond(Cond x)
a.evaluar()	entregar resultado de condición	boolean evaluar()
a.and(b)	Condición a && b	Cond and(Cond x)
a.or(b)	Condición a || b	Cond or(Cond x)
a.not()	Condición !a	Cond not()
a.toString()	String que representa la condición	String toString()

Escriba la clase Cond, salvo el 2º constructor (que construye el árbol a partir de un String).

Importante. El método evaluar debe ser eficiente, esto es, debe evaluar concurrentemente los árboles izquierdo y derecho y considerar el resultado del árbol derecho sólo si es pertinente. Por ejemplo, en la condición a && b, si a es false, no es necesario evaluar b.

Indicación. Las hojas deben guardarse como objetos de la clase Boolean (que recubre el tipo boolean).

//declaraciones: 0.5

```
class Nodo{
Object valor; Nodo izq, der;
public Nodo(Object x,Nodo y,Nodo z){valor=x;izq=y;der=z;}
}
class Cond{
private Nodo raiz;
public Cond(){raiz=null;}
public Cond(boolean x){ //0.5
  raiz=new Nodo(new Boolean(x),null,null);
}
public Cond(Cond x){  //0.5
  raiz=new Nodo(x.raiz.valor,copiar(x.raiz.izq),copiar(x.raiz.der));
}
protected Nodo copiar(Nodo r){
  if(r==null) return null;
  return new Nodo(r.valor, copiar(r.izq), copiar(r.der));
}
public Cond not(){ //0.5
  Cond aux=new Cond(this);
  aux.raiz=new Nodo("!",null,aux.raiz);
  return aux;
}
public Cond and(Cond x){ //0.5
  Cond aux=new Cond(),a=new Cond(this),b=new Cond(x);
  aux.raiz=new Nodo("&&",a.raiz,b.raiz);
  return aux;
}
public Cond or(Cond x){ //0.5
  Cond aux=new Cond(),a=new Cond(this),b=new Cond(x);
  aux.raiz=new Nodo("||",a.raiz,b.raiz);
  return aux;
}
public String toString(){//0.5
  return toString(raiz);}
protected String toString(Nodo r){
  if(r==null) return "";
  if(r.izq==null&&r.der==null)return ""+((Boolean)(r.valor)).booleanValue();
  return "("+toString(r.izq)+r.valor+toString(r.der)+")";
}
```

```
public boolean evaluar()throws Exception{ //1.5
  return evaluar(raiz);       //0.1
}
protected boolean evaluar(Nodo r)throws Exception{//0.1
  if(r==null)return false;    //0.1

  if(r.izq==null && r.der==null)
    return ((Boolean)(r.valor)).booleanValue();//0.1

  if(r.valor.equals("!"))     //0.1
    return !evaluar(r.der);   //0.1

  T t=new T(r.der); t.start();//0.2

  boolean a=evaluar(r.izq);   //0.1

  if(r.valor.equals("&&") && a==false) return false;//0.2

  if(r.valor.equals("||") && a==true) return true;   //0.2

  t.join();             //0.1
  return t.resultado(); //0.1
}
//1 punto
class T extends Thread{             //0.1
private Nodo r; private boolean b;  //0.2
public T(Nodo x){r=x;}              //0.2
public void run(){b=evaluar(r);}    //0.3
public boolean resultado(){return b;}//0.2
}

}
```