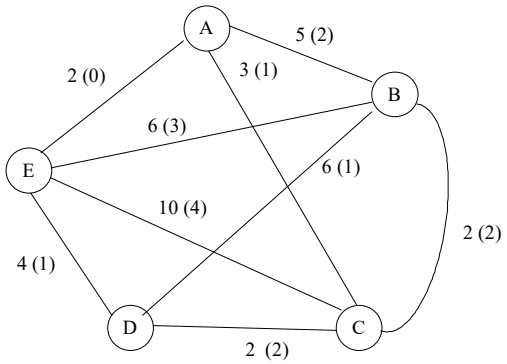
Paradigmas de Programación
Paradigma de Objetos

Modelado y Desarrollo

Ejercicio 1

Dado un mapa como el siguiente, en el cual se indican las ciudades, las distancias entre ellas y cuantos kilómetros son de camino de tierra (la cantidad entre paréntesis).



Se le solicita que defina la clase principal **Mapa** y los siguientes métodos:

De clase:

new
Crea una instancia de Mapa.

De instancia

conexion: *ciudad1* **con:** *ciudad2* **distancia:** *distancia*

ciudad1,ciudad2: son las ciudades que se relacionan.
distancia: Arreglo de dos elementos, que representa la distancia total y de tierra (Total ,Tierra).

caminoElegidoDe: *ciudadI* **a:** *ciudadF* **maximoTierra:** *máximo*

Dará como resultado el listado de ciudades que hay que recorrer para ir de *ciudadI* a *ciudadF* por el camino de mínima longitud, y que por el mismo no sea necesario transitar una longitud total de camino de tierra que supere *máximo*.

Ejercicio 2

a.) Se le solicita que genere una clase **ArregloHomogeneo**. La misma debería tener el mismo comportamiento que Array, pero además se requiere que todos los elementos de un arreglo dado sean instancias de la misma clase que el primer elemento que se adicione (el primer elemento que se adicione no tiene por que ser el de la posición 1).

Por ejemplo:

A := ArregloHomogeneo new: 10. A at: 2 put: 1. A at: 3 put: Object new. "Error. Se pretende ingresar un elemento de tipo no homogeneo"	En este caso los elementos que se incorporen a continuación deberán ser todos enteros. Al pretender adicionar como elemento de A una instancia de Object, se presenta un mensaje de error.

Por lo tanto se le pide que describa la nueva clase , incorporando si se requiere nuevas variables.

Deberá definir los métodos:

at: put:

tipo (retorna la clase de los elementos del arreglo)

declararTipo: *tipo* (permite establecer un tipo determinado en el caso en que no se haya definido por la inclusión de un elemento).

b.) Defina una subclase de Array denominada **MatrizHomogénea**, la cual deberá representar una matriz cuadrada (nxn), y todos sus elementos deberán ser instancias de la misma clase. Se le solicita que utilice en la definición de esta clase, el ArregloHomogéneo definido en el punto a.. Deberá definir al menos los métodos: **new:** *(x@y)* , **at:** *(x@y)* **put:** *valor* y **at:** *(x@y)*. Tenga especial cuidado en el empleo de los métodos redefinidos.

Ejercicio 3

Se le pide que defina en SmallTalk la clase Matriz, la cual deberá representar a una matriz de NxN. Describa claramente el significado de la representación seleccionada. La clase definida, además de los métodos básicos necesarios, de creación e inserción, deberá contener los siguientes métodos:

incorporarFila: *fila* **en:** *posición*
el resultado de la evaluación de este método, modifica la matriz receptora incorporando *fila* en *posición*.

incorporarColumna: *columna* **en:** *posición*
el resultado de la evaluación de este método, modifica la matriz receptora incorporando *columna* en *posición*.

eliminarFila: *posición*
el resultado de la evaluación de este método, modifica la matriz receptora eliminando la fila ubicada en *posición*.

eliminarColumna: *posición*
el resultado de la evaluación de este método, modifica la matriz receptora eliminando la columna ubicada en *posición*.

Ejercicio 4

Se desea modelar en Smalltalk un juego de cartas. En este juego se utiliza un mazo de cartas españolas de 50 cartas. En el mismo puede participar un número arbitrario de jugadores, a cada uno de los cuales se la asignan *n* cartas del mazo al comienzo del juego. En particular se han identificado las siguientes clases y sus comportamientos:

Mazo

crear

Este método retorna una instancia que contiene todas las cartas del mazo

tomarPrimera

Retorna la primera carta del mazo y modifica el estado del mismo para reflejar que esa carta ya no se encuentra en el mazo.

Carta

crear: n palo: l

Este método retorna una instancia que representa la carta de valor **n** y palo **l**. No se puede crear una carta ya instanciada. Por ejemplo: al enviar el mensaje **crear:** *2* **palo:** *#oro* a Carta se crea la carta 2 de oro.

mono

Retorna verdadero si el receptor es una instancia de carta que representa a un mono (comodín) y falso en otro caso.

valor

Retorna el valor correspondiente a la carta (1, 2, ..., 12). Produce un error si se envía a una carta que representa un mono.

palo

Retorna el palo correspondiente a la carta receptor (oro, basto, espada o copa). Produce un error si se envía a una carta que representa un mono.

Jugador

crear. Este método retorna una instancia de jugador que no tiene cartas en su poder.

recibirCarta: unaCarta.

Agrega la carta unaCarta a las que posee el jugador.

Juego

Aquí debe colocar los métodos que considere adecuados.

Se le solicita lo siguiente:

1. Codificar todos los métodos de mazo y carta. En especial, defina la clase Carta de forma tal que el código de los métodos de instancia no utilice mensajes tipo ifTrue:ifFalse:.
2. Proponga un método **repartir:** *n* en la clase adecuada que reparta, es decir: asigne *n* cartas a los jugadores en la forma habitual en que se hace en un juego de cartas, entregando alternativamente cartas a los jugadores tomando siempre la primera carta del mazo.
3. Proponer un método para mezclar el mazo.

Ejercicio 5

Una Empresa dedicada a la venta de Jugos, desea registrar la informacion de sus vendedores, respecto a: *dni*, *nombre*, *zonaDeVenta*, *vtasMensuales*, *totalAnual* y *comisión*.

Donde *vtasMensuales* representa las ventas realizadas por el vendedor en cada uno de los doce meses del año, *totalAnual* representa la suma de las ventas en un año y *comision* un porcentaje que depende del *totalAnual*.

Además la variable *nombre* está asociada con una clase Nombre cuyas instancias representan el *apellido*, el *primerNombre* y el *segundoNombre* de cada vendedor.

a) Definir la clase Vendedor, y la clase Nombre

b) Definir en la clase que corresponda los métodos que permitan los siguientes requerimientos:

b.1) Crear una instancia de la clase Vendedor y solicitar al usuario los datos para inicializar sus variables, excepto las variables *vtasMensuales*, *totalAnual* y *comisión* que se inicializan con ceros.

b.2) Ingresar las ventas de un determinado mes.

b.3) Determinar la comisión del vendedor, teniendo en cuenta:

si *totalAnual* < $ 50.000 , la *comisión* es cero.

Si $50.000 <= *totalAnual* < $75.000 , la *comision* es del 15%.

Si $75.000 <= *totalAnual* < $100.000 , la *comision* es del 20%.

Si $100.000 <= *totalAnual,* la *comision* es del 30 %.

Ejercicio 6

Considere la situación representada por un tablero cuadrado con nxn celdas y una pieza peón, que puede realizar los siguientes movimientos:

peón(x+1,y)	peón(x-1,y)	peón(x,y+1)	peón(x,y-1)
derecha	izquierda	avanzar	retroceder

Se le solicita que defina las clases tablero, celda y peón (con los atributos y métodos que considere necesario).

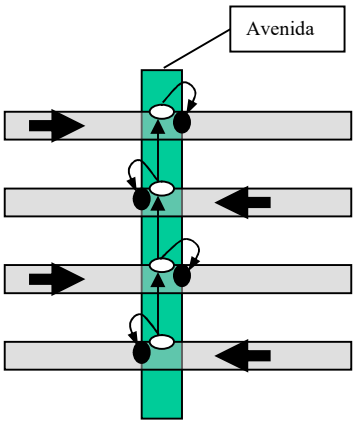
Una instancia de tablero deberá poder crearse de cualquier dimensión n.

Se pretende que una instancia de tablero dada posea una instancia de peón, el cual debería poder moverse con las operaciones indicada en la tabla. El peón debería poseer una referencia a la instancia de celda en la cual se encuentra.

Además se solicita que la pieza indique que no puede ejecutar una dada operación si la misma implica que salga de los límites del tablero.

Ejercicio 7

Se le solicita que genere las clases necesarias en SmallTalk para representar el comportamiento de su sistema de semáforos, como el esquematizado en la figura. Las elipses representan los semáforos:



Un semáforo puede estar en Rojo, Verde o Amarillo. La secuencia de transición posible es: Verde, Amarillo, Rojo, Verde.

Los semáforos ubicados sobre la avenida deben coordinar una onda verde, es decir que una vez que un semáforo pasó a encender su luz verde, el semáforo de la intersección siguiente deberá pasar a verde luego de 10" (este valor debería poder ser modificado por el sistema). Cuando un semáforo sobre la avenida se desea pasar a verde, deberá notificar al semáforo que controla el tráfico sobre la correspondiente transversal para que el mismo comience su transición a rojo.

En el sistema que se está diseñando los objetos avenida deben conocer cuales son todos los semáforos que poseen. Es importante notar que solamente los semáforos que están sobre una avenida tienen semáforos sucesores y asociados a su calle transversal. Todos los días el sistemas de semáforo de las avenidas comienza a funcionar a las 6 de la mañana y a partir de las 23 horas todos los semáforos de la avenida y sus transversales permanecen en Amarillo. Considere que en el sistema de objetos existe un objeto **reloj**, al cual se le puede enviar el mensaje *hora*, el cual retorna la hora actual.

Se le solicita que defina las clases: **Avenida** y **Semáforo,** para lo cual debe definir por ejemplo los siguientes métodos. Estos no son los únicos métodos de las mencionadas clases, agregue todos aquellos que considere necesarios.

Clase Avenida

agregarSemáforo
Este método es el encargado de incorporar un nuevo semáforo, debe también crear el semáforo de la calle transversal.

retardo: retardoEntreSemáforos
Modifica el tiempo de retardo entre todos los semáforos de una avenida

Clase Semáforo

asociarSucesor: semáforo

asociarTransversal: semáforo

estado
Retorna el estado del semáforo (Rojo, Amarillo o Verde)

habilitar
Este método espera el retardo establecido, y luego se coloca en verde. Suponga que tiene disponible el método, *suspender:* segundos, el cual suspende la ejecución del objeto durante el intervalo indicado por segundos. Suspender: está disponible en el sistema de objetos, usted no debe definirlo.

Ejercicio 8

Se desea construir la estructura básica de un editor de texto tomando como clases básicas: documento, página, línea, palabra y caracter. En esta definición elemental, una página tendrá un número máximo de líneas y una línea tendrá un número máximo de caracteres. Una palabra tendrá un dado número de caracteres.

Clase Documento
new
Crea un objeto documento vacío.

insertarPágina
Crea una página vacía al final del documento.

Clase Página
new
Crea un objeto página, el cual no contiene ninguna línea.

adicionarLínea: *línea* **posición**: *n*
Incorpora una línea en una posición dada de la página. Si como consecuencia de la incorporación de la línea se excede el número máximo de líneas de la página, la última línea deberá ser enviada a la próxima página, y así sucesivamente. La línea incorporada puede ser vacía.

Clase Línea
new

palabrasAMover: *palabra*
Retorna una OrderedCollection con las palabras que quedarían fuera de la línea como consecuencia de incorporar *palabra* al comienzo de la misma. Si la incorporación no produce que se exceda el número de caracteres máximo de la línea, el método retorna una colección vacía.

adicionarPalabra: *palabra* **posición:** *n*

Incopora un objeto palabra en la línea, como la palabra enésima. Si la palabra, en esa posición excede la longitud máxima de la línea, el método anuncia la imposibilidad de incorporarla en esa posición. Si la puede incorporar, pero algunas de las palabras posteriores ya no tienen más espacio, deberán ser detectadas y enviadas al comienzo de la línea siguiente.

Clase Palabra

longitud

retorna la longitud de la palabra

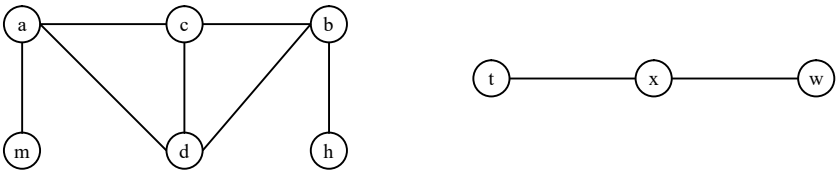
Se le solicita que defina las clases y métodos explicitados (y todos aquellos que considere convenientes), teniendo especial atención en identificar las relaciones entre las distintas clases.

Ejercicio 9

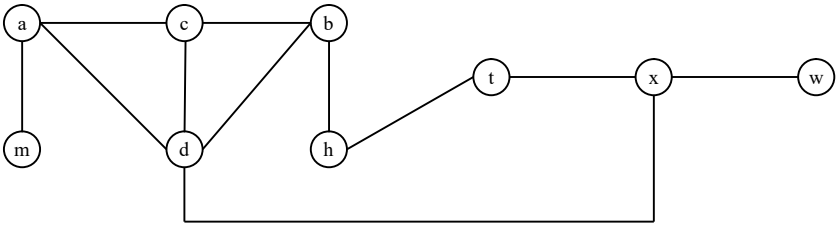
Un grafo G = {V, A} es conexo si para todo par vértices $v_i, v_j \in V \ / \ v_i \neq v_j$ existe un camino (secuencia de aristas $\in$ A) que los une.

Ejemplos

Grafo no conexo



Grafo conexo



Definir las clases **Grafo**, **Nodo** y **Arco** para representar el grafo

Clase Grafo

adicionarNodo: *nodo*
conectar: *nodo1* **con:** *nodo2*
eliminar: *nodo*
conexo?

El método conexo? retorna ***true*** si el Grafo receptor satisface la propiedad indicada.

Defina claramente la estructura de cada clase, los métodos necesarios además de los requeridos en **Grafo** y las restantes clases. Documente de que forma se relacionan las instancias de las distintas clases. Recuerde definir los métodos de creación de instancia.

Ejercicio 10

Proponer una jerarquía de clases que utilice como clase base una clase llamada LAN (red de área local). Las subclases derivadas deben representar diferentes topologías de red, como ser estrella, anillo y bus. Un objeto red debe poseer propiedades tales como: soporte de transmisión, control de acceso, formato de datos, velocidad de transmisión. Se desea simular las actividades de los nodos de tal LAN.

La red consta de nodos, que pueden ser dispositivos tales como: computadoras personales, máquinas fax, impresoras, etc.

Una tarea de la LAN es soportar comunicación de datos entre sus nodos. Por lo tanto, para emplear el sistema de objetos en la simulación de la red, se debe poder como mínimo:

- Listar todos los nodos de la red
- Añadir un nuevo nodo a la red
- Quitar un nodo de la red
- Configurar una topología de estrella, en la cual hay un nodo central que se ocupa de recibir y distribuir los envíos de paquetes.
- Especificar el tamaño del paquete, que es el tamaño en bytes del mensaje que va de un nodo a otro
- Enviar un paquete de un nodo especificado a otro
- Difundir un paquete desde un nodo a todos los demás nodos de la red

Se le solicita que modele la descripción anterior en SmallTalk

Ejercicio 11

Se le solicita que genere en SmallTalk las clases necesarias para representar una porción del manejo de pedidos de una empresa manufacturera. El objetivo inicial es determinar si para una orden de compra dada, es posible cumplir con la fecha de entrega, y qué partes es necesario adquirir para ello. Para poder implementar esto es necesario conocer como se puede implementar una estructura de un producto. La estructura de producto es el sistema de información que contiene los datos vinculados con qué productos y en qué cantidad son necesarios, para producir las distintas partes empleadas en la empresa. Cualquier producto Final de los comercializados por la empresa requiere como insumos, partes fabricados por la misma o partes compradas a terceros. Las partes fabricadas, a su vez requieren de otras partes fabricadas o compradas. Es decir que un producto final esta compuesto por partes fabricadas y/o partes compradas, y esto se continua recursivamente para las partes fabricadas. Todas las partes tienen asociadas un tiempo de fabricación o de entrega por el proveedor (TiempoD), el cual deberá ser utilizado cuando se desee saber si se puede satisfacer la fecha de entrega especificada en la OrdenCompra.

Es importante tener registrada en la estructura de producto que cantidad de las partes intervinientes es necesaria para la fabricación del mismo.

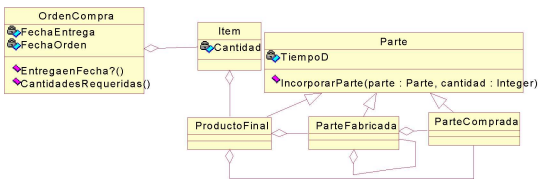
Por otra parte, la OrdenCompra indica que productos se debe proveer, en qué cantidades y en qué fecha.

Se le solicita que defina las clases que se presentan a continuación (si requiere otras incorpórelas) de forma tal de poder definir una estructura de producto, y poder implementar los métodos de OrdenCompra:

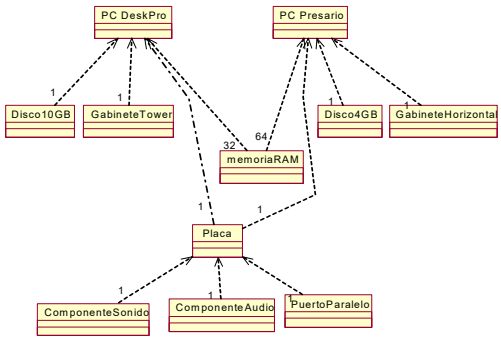
entregaFecha? :Retorna True si se puede satisfacer la fecha de entrega. Este deberá encontrar la trayectoria de máxima duración en la estructura de producto del Producto Final especificado en la Orden de compra.

cantidadesRequeridas : Cantidad de partes que deben ser compradas

Diagrama de Clases



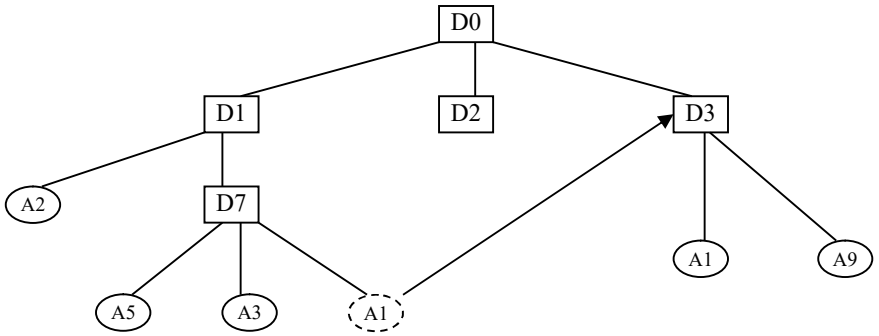
Ejemplo de Estructura de Producto



Ejercicio 12

El árbol que se presenta en la figura se utiliza para llevar el registro de un sistema de archivo, en él, cada nodo puede ser:

- un archivo (hoja)
- un directorio raíz (raíz)
- un directorio (nodo interno / nodo hoja)
- un enlace (hoja)



La diferencia que existe entre un enlace y un archivo es que el enlace se utiliza para indicar los archivos compartidos, por ejemplo, en la figura presentada el archivo A1 se encuentra en el directorio D3 (archivo físico) y tiene un enlace en el directorio D7. La información que provee el enlace es la cadena de directorios para llegar al archivo físico (para el ejemplo sería (D0 D3)).
Se le solicita que defina en Smalltalk las clases Archivo, Directorio, Directorio Raíz y Enlace, con sus respectivas relaciones y atributos (los mismos deberán ser significativos).
Para cada clase codifique el método para la creación de instancias.

Para *"Directorio Raíz"* codificar:

buscarArchivo: *camino*
Retorna la instancia de archivo que especifica camino. Tener en cuenta que la ruta suministrada (*camino*) puede llevar a un archivo enlace, en caso de ser así, se deberá tomar la información de este, para luego buscar el archivo físico correspondiente. Considerar que *camino*, contendrá los nombres de los directorios (**no** los objetos directorios) y el nombre del archivo o enlace al final.

agregarDirectorio: *camino* **nuevoDir:** *directorio*

Agrega el directorio "directorio" en el directorio que especifica la ruta "camino"

borrarDirectorio: *camino*

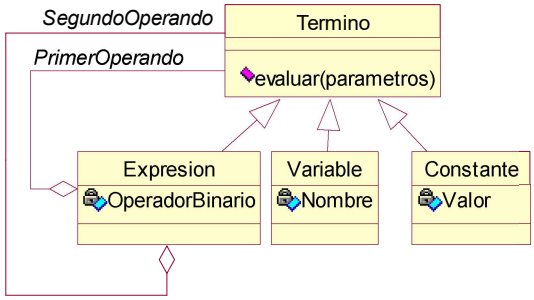
Elimina el directorio que se encuentra al final de *camino*. Eliminar un directorio tiene como consecuencia eliminar también su contenido.

Nota:

Obviamente deberán definirse métodos auxiliares para poder implementar los métodos solicitados. Especifique claramente el significado de cada uno y sus argumentos.

Ejercicio 13

A continuación se presentan las clases que representan un sistema de expresiones algebraicas binarias:



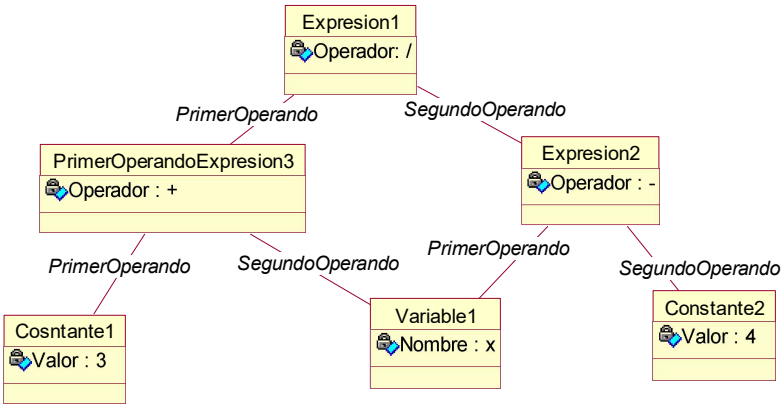
Este diagrama de clases pretende representar la siguiente gramática:

<Termino> :=: <Expresion> | <Variable> | <Constante>
<Expresion> :=: <PrimerOperando> <Operador> <SegundoOperando>
<PrimerOperando> :=: <Termino>
<SegundoOperando> :=: <Termino>
<Operador> :=: + | - | * | /

Se le solicita que defina en SmallTalk las clase indicadas, con los correspondientes métodos. Una constante debe ser un entero. El método *evaluar(*parámetro*)*, deberá ser redefinido en cada subclase de Termino. El argumento **parámetro** deberá ser una instancia de Dictionary, en donde la clave corresponde a una variable y el valor asociado será el entero por el cual se reemplazarán las ocurrencias de la variable en la expresión.

Expresión	**Variable**	**Constante**
new *addPrimerOperando* *addSegundoOperando* *variables?* **; retorna la lista de objetos variable que participan de la expresión.**	*new* *nombre* *nombre:* nombre	*new* *valor* *valor:* valor

A continuación se muestra la estructura de instancias correspondiente a la expresión:
(3 + x) / (x + 4) ; que estaría representada por el objeto Expresión 1



Ejercicio 14

Se le solicita que defina la clase GrafoDirigido, los objetos instancias de esta clase representarán grafos dirigidos.

Definir los siguientes métodos:

De Clase

new : crea una instancia de grafo dirigido sin ningún nodo.

De instancia.

agregarNodo: *nodo*
Agrega el objeto nodo al grafo.

conectarNodo1: *nodo1* **con:** *nodo2*
Conecta el nodo1 con el nodo2.

existeCiclo
Retorna *true* en caso que el grafo tenga al menos un ciclo.

conexo
Devuelve una nueva instancia de grafo, pero con la característica de ser conexo. Se tendrá que determinar si la instancia a la que se le envía el mensaje satisface dicha propiedad, de no ser así, se deberán implementar los mecanismos adecuados para transformar a dicha instancia en un grafo conexo.

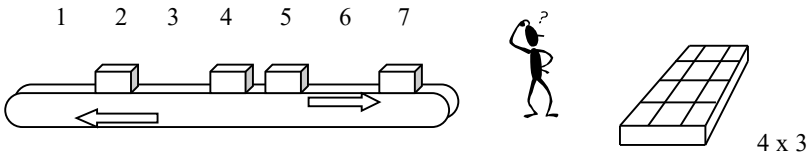
= *unGrafo*
Retorna *true* en caso que los grafos sean iguales. Dos grafos son iguales si tienen los mismos nodos y las mismas relaciones entre nodos (aristas).

Ejercicio 15

Para el diseño de un simulador de una planta de fabricación de piezas se desea desarrollar en Smalltalk las clases necesarias para representar la situación que se describe a continuación:

El trabajo de rutina de un operario en la planta consiste en acomodar piezas en contenedores. Las piezas son tomadas desde una cinta transportadora, la cual tiene una capacidad máxima de carga, dada por el número de piezas máximo que puede transportar en un determinado momento (todas las piezas tienen el mismo tamaño, el cual no debe considerarse en el modelo). Para evitar que las piezas caigan al suelo al realizar su trabajo, el operario hace avanzar manualmente la cinta y descarga las piezas a medida que estas están disponibles, utilizándolas para completar los contenedores. Cada contenedor posee un espacio también limitado, con celdas para las distintas piezas que se cargarán en el mismo. El operario debe llenar el contenedor con las piezas obtenidas de la cinta transportadora, para lo cual debe considerar el número de piezas que éste puede contener. Un contenedor contiene una grilla de celdas (n x m) cuyas dimensiones varían de un contenedor a otro. El operario no conoce de antemano el tamaño del contenedor.

La siguiente figura presenta un ejemplo de una cinta con capacidad máxima de 7 piezas y un contenedor (la dimensión de éste se define en el momento de crearlo) de dimensión 4 x 3. Nótese que no necesariamente todas las posiciones de la cinta contendrán una pieza.



En el estado representado por la figura, si la cinta avanza, la pieza en la posición 7 caerá al suelo, lo cual no debe ocurrir (el operario no debe avanzar la cinta si hay una pieza lista).

Se han identificado las clases Pieza, CintaTransportadora, Operario y Contenedor.
En ellas, se desean implementar los siguientes métodos:

Clase CintaTransportadora:

avanzar

Avanza la cinta 1 posición. Si existe una pieza lista para descargar cuando la cinta avanza, dicha pieza se pierde (cae al suelo). Al avanzar, cada pieza o espacio vacío avanza un lugar.

cargar: *unaPieza*

Agrega la pieza unaPieza a la cinta transportadora que recibe el mensaje. La nueva pieza ingresa a la cinta en la posición inicial (posición 1 en la figura). Si ya existe una pieza en la posición inicial se produce un error.

piezaLista

Devuelve un valor booleano indicando si hay una pieza disponible para descargar en la posición final de la cinta que recibe el mensaje (posición 7 en la figura).

descargar

Descarga la pieza que se encuentra en la posición final de la cinta transportadora que recibe el mensaje, retornando la misma. Produce un error cuando no hay una pieza disponible.

Clase Pieza:

ubicación

retorna el objeto en que se encuentra la pieza, unaCinta o unContenedor

Clase Operario:

transladarPiezasDesde: *unaCintaTransportadora* **hasta:** *unContenedor*

Realiza el trabajo de llenar el contenedor unContenedor (un objeto Contenedor) con piezas descargadas de unaCintaTransportadora (un objeto Cinta Transportadora). (*Nótese que el operario no es el encargado de cargar la cinta transportadora*).

Clase Contenedor:

dimension

Devuelve las dimensiones del contenedor receptor (alto x ancho).

colocar: *unaPieza* **en:** *unaPosicion*

Ubica la pieza unaPieza en la posición unaPosicion del receptor. Produce un error si unaPosicion ya contiene otra pieza o si la posición unaPosición no es válida.

Implemente los métodos anteriormente descriptos junto con los métodos de inicialización que correspondan. Puede añadir otros métodos, describiendo detalladamente el uso para el cual se destinan. Defina siempre claramente las variables de instancia o de clase que utiliza.

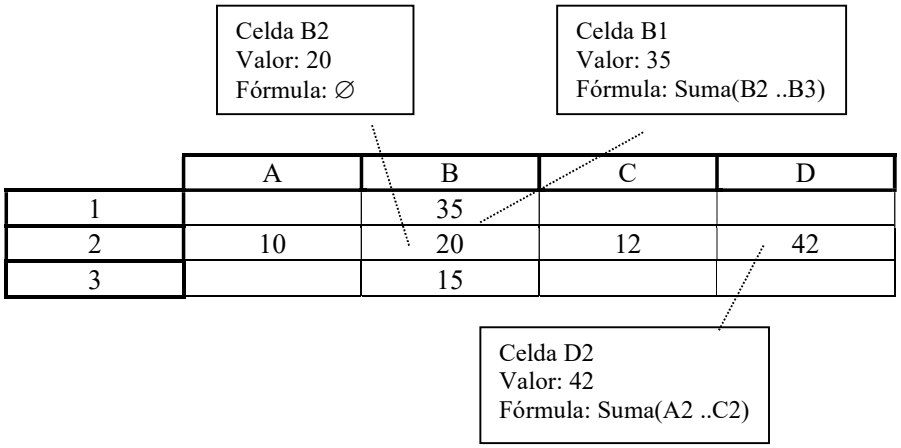
Defina además la clase **CintaTransportadoraSegura**. Una cinta transportadora segura es una cinta transportadora que no permite que sus piezas caigan al suelo.

Ejercicio 16

Desarrollar en SmallTalk una versión reducida de una planilla de cálculo. La planilla se representa con una estructura matricial compuesta por celdas, donde cada celda se identifica por una columna y una fila. Las columnas se identifican con una letra y las filas con un número.

Cada celda tiene un valor (un número) y puede o no tener una fórmula asociada. Una celda que posee una fórmula obtiene su valor a partir de la evaluación de la misma. Las fórmulas permiten que el valor de una celda dependa de los valores de otras celdas

De esta manera puede utilizarse la planilla para realizar numerosos cálculos. Por ejemplo:



Se desea que la modificación del valor de una celda produzca la actualización de los valores de aquellas celdas que utilizan ese valor en sus fórmulas. En el ejemplo, esto significa que si se modifica A2, B2 o C2, entonces debe actualizarse el valor de D2.

a) Definir la clase **PlanillaCalculo**. Indicar claramente la representación elegida. Puede definir las clases auxiliares que necesite. No limitar el tamaño de una planilla de cálculo a ningún número máximo de filas y columnas.

   Desarrollar los métodos necesarios para crear e inicializar instancias de PlanillaCalculo.

b) Definir los métodos de instancia siguientes:

valor: *unNumero* **en:** *unaPosicion*

Asigna el valor *unNumero* a la celda indicada por *unaPosicion*. Si la celda posee una fórmula la misma queda anulada. El objeto *unaPosicion* será una instancia de la clase **Array** compuesta por dos elementos: un carácter y un número, por ejemplo: #($A, 2). Tenga en cuenta que al asignar un nuevo valor a una celda esto puede modificar el valor de otras celdas.

valorEn: *unaPosicion*

Retorna el valor asociado a la celda indicada por *unaPosicion*. El objeto *unaPosicion* será una instancia de la clase **Array** compuesta por dos elementos: un carácter y un número, por ejemplo: #($A, 2).

sumarDesde: *unaPosicionInicial* **hasta:** *unaPosicionFinal* **en:** *unaPosicion*

Asigna una fórmula de suma a la celda indicada por *unaPosicion*. El nuevo valor de la celda será la suma de los valores del conjunto de las celdas ubicadas entre *unaPosicionInicial* y *unaPosicionFinal*, al igual que en el ejemplo. Considere que las posiciones inicial y final necesariamente se encuentran en la misma fila o columna y que *unaPosicion* no debe ser miembro del conjunto mencionado.

Tener en cuenta que, aunque no se le piden otras fórmulas además de la suma mencionada, **debe definir una implementación que le permita incorporar fácilmente otros tipos de fórmulas**.