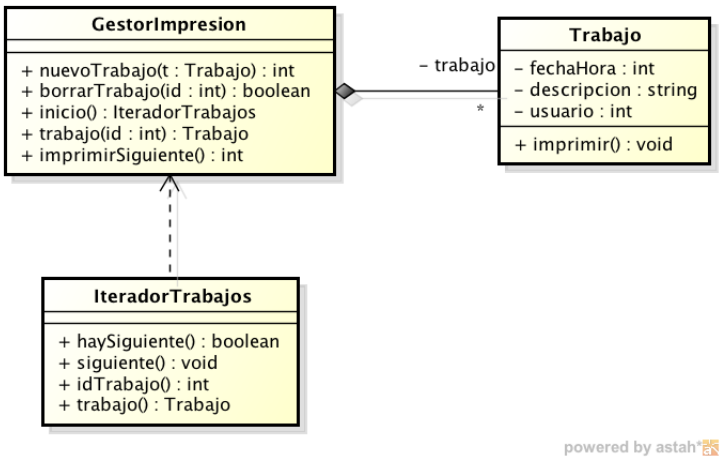
Estructuras de Datos

Relación de ejercicios 2: implementación en C++ de diagramas de clases seleccionando las estructuras de datos adecuadas

1. Las siguientes clases forman parte de un sistema de impresión de un sistema operativo. La clase *Trabajo* representa los trabajos de impresión enviados por los usuarios y el gestor de impresión el organizador de la cola de impresión de trabajos. Las operaciones de la clase impresión son:

   - *nuevoTrabajo()*: inserta un trabajo en la cola y devuelve un identificador único para cada trabajo.
   - *borraTrabajo():* elimina un trabajo de la cola indicando su identificador.
   - *trabajo()*: Devuelve el trabajo asociado a un identificador.
   - *inicio()*: Devuelve un iterador apuntando al primer trabajo.
   - *imprimir()*: imprime el siguiente trabajo de la cola y lo elimina. La política del gestor de impresión es el de una cola con prioridad. Si existe algún trabajo en la cola enviado por el *root* (id de usuario 0), este tendrá preferencia sobre todos los demas. El resto sigue un criterio FIFO.

   Implementar las clases del diagrama. No pueden usarse las clases STL *queue* o *priorityqueue*. Aunque no aparecen en el diagrama, deben implementarse también sus constructores.

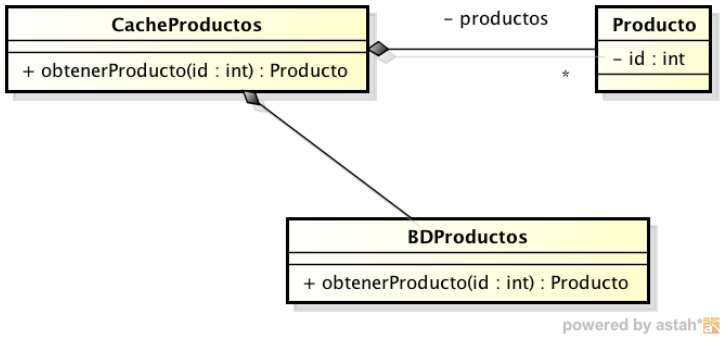


2. Disponemos de una base de datos de productos con un número muy grande de entradas que debe ser consultada con una frecuencia muy alta para

localizar productos a partir de su identificador. Para acelerar este acceso disponemos de una caché que guarda los últimos 1000 productos solicitados. El funcionamiento es el siguiente:

- El producto es solicitado a través de la operación obtenerProducto de CacheProductos.
- Si existe en la caché, se devuelve inmediatamente.
- Si no existe en la caché, se solicita a la base de datos y se guarda en la caché por si vuelve a ser solicitado. Si la caché tiene 1000 elementos, se elimina el último que fue solicitado y se sustituye por éste. Finalmente se devuelve el producto.

Implementar la clase CacheProductos. La relación "productos no tiene por qué traducirse en una única estructura de datos, y pueden incluirse los atributos adicionales que se consideren necesarios.



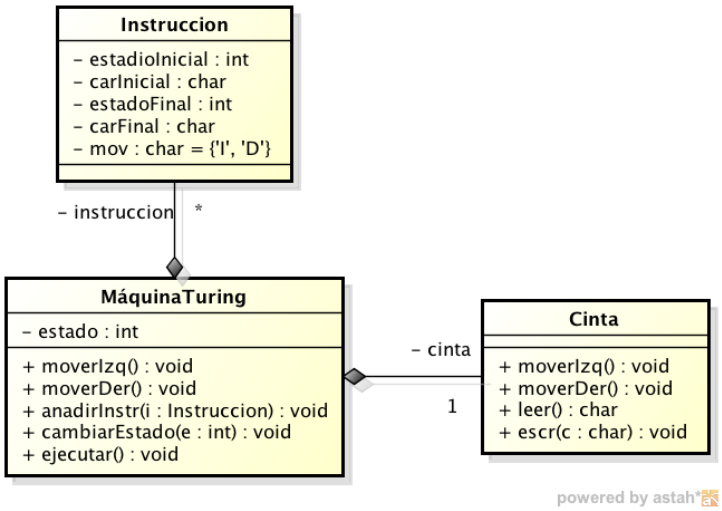
3. Implementar una Máquina de Turing utilizando la descripción proporcionada por el siguiente diagrama UML. Una máquina de Turing es un modelo simplificado de una computadora, formado por los siguientes elemenetos:

- Una cinta de caracteres, idealmente infinita, que hace las veces de memoria. La cinta se desplaza a izquierda o derecha y permite leer el caracter situado bajo la cabeza de la máquina o modificar este caracter.
- El estado actual representado por un entero positivo.
- Un conjunto de instrucciones que definen el comportamiento de la máquina. Una instrucción está constituida por 5 elementos:
  - El estado inicial de la máquina
  - El carácter que debe haber en la cinta
  - El estado después de ejecutar la instrucción
  - El carácter a escribir
  - La dirección donde debe moverse la cabeza.

Una instrucción se activa si el estado inicial y el carácter que hay en la cinta son los indicados (precondición), y su efecto es pasar al estado final, escribir el carácter indicado y mover la cabeza en la dirección indicada.

Las operaciones moverIzq(), moverDer() y escr() de la clase MaquinaTuring tienen como objeto escribir el contenido inicial de la cinta. De modo similar añadirInstr() permite especificar las instrucciones de la máquina y cambiarEstado() establecer el estado inicial. Todas esta configuración se realiza antes de que la máquina empiece a funcionar.

La operación ejecutar() es la que pone la máquina en funcionamiento. En cada paso de la ejecución se debe comprobar si se cumple la precondición de alguna instrucción. En caso afirmativo se realizan las acciones indicadas por ésta. El proceso se repite hasta que no se cumpla la precondición de ninguna instrucción. En este caso la máquina para y la operación ejecutar() termina.



4. El siguiente diagrama contiene las clases necesarias para la simulación de un proceso de evolución genética. La clase *Cromosoma* contiene la información genética en forma de un vector de 128 genes (de tipo char). Las operaciones que incluye son:

- *init:* Inicializar los genes aleatoriamente.
- *copiar:* Copiar los genes de otro cromosoma
- *mutar:* Cambiar de forma aleatoria los genes con probabilidad indicada por el índice de mutación (real entre 0 y 1, para cada gen 0->nunca mutar, 1->mutación segura).
- *cruzar:* Generar un nuevo cromosoma con los 64 primeros genes del cromosoma y las 64 últimos del otro cromosoma *b*.

La clase *Genoma* contiene 64 cromosomas, y controla el proceso de evolución. Sus operaciones son:

- *init:* Inicializar todos los cromosomas aleatoriamente.
- *generación:* Construye la siguiente generación siguiendo el procedimiento que se explica a continuación. Primero se escogen 32 cromosomas aleatoriamente, después se obtiene una nuevo conjunto de 64 como resultado del cruzamiento en ambos sentidos con los 32 cromosomas restantes. Finalmente se mutan los cromosomas obtenidos, que pasan a constituir la nueva información del genoma.

Realizar la implementación completa de las clases anteriores. Incluir los constructores, operadores, etc., que se consideren necesarios. Puede utilizarse STL.