Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información

CI-2692 - Laboratorio de Algoritmos y Estructuras II
Septiembre-Diciembre 2013

Proyecto 2 (v2.1)

1. Problema

Un grupo de físicos del CERN ha decidido retirarse de la investigación y mercadear un descubrimiento que piensa los hará increíblemente exitosos. Este grupo de físicos ha encontrado un fenómeno cuántico de portales que les permite apropiarse de "espacio volumétrico" construido en universos paralelos (multiverso), y anclarlos secuencialmente a nuestro universo mediante portales dimensionales. Dichos espacios son paralelepípedos (*"tubos rectangulares"*) con anchura y altura de 3 metros y longitud aleatoria de entre 5 y 25 metros, y a cada uno se le asigna una etiqueta generada secuencialmente para su identificación.

Los físicos determinan que el único negocio que podría aprovechar esta propiedad del espacio-tiempo es uno de almacenamiento de vehículos para espectáculos, por lo que fundan la compañía Parqueadero Aleph-0, con el eslogan "Siempre tenemos puesto".

Ahora, cada "tubo" desafortunadamente tiene una sola abertura o "portal", por lo que los vehículos se estacionan desde el "fondo" hacia el "portal" y se recuperan en sentido inverso. Dado que la capacidad de cada *tubo* es limitada, sólo se puede estacionar un vehículo si existe espacio suficiente para el mismo. De no haber espacio suficiente para introducir un vehículo, se cierra el *tubo* y se fabrica uno nuevo, enviando el *tubo* cerrado al final de la secuencia de *tubos*.

Es de notar que la operación de *retiro* de un vehículo de un *tubo* específico desestabiliza la matriz taquiónica del mismo, y requiere que se "apropie" un nuevo *tubo* del multiverso, transferir todos los vehículos del tubo original (excepto el buscado, si existe) a este nuevo *tubo*, y que el nuevo *tubo* se coloque al final de la lista de *tubos* "apropiados". Este nuevo *tubo* hereda la etiqueta del *tubo* original, para evitar que se pierda la pista de los vehículos, y se crea del mismo tamaño. Es de notar que el mero recorrido de la secuencia de tubos ***sin entrar*** por los portales para retirar vehículos **NO** destruye el *tubo*.

~~Los físicos-empresarios han también determinado que existen peligros que pueden amenazar la integridad de un *tubo* lleno. Si en el universo paralelo donde se ubica el *tubo* estalla una supernova, el flujo de neutrinos procedentes de la explosión hará estallar el *tubo*, en cuyo caso habrá que recuperar TODOS los vehículos de ese *tubo* y repartirlos secuencialmente entre los otros *tubos* que le siguen, y si se agota la capacidad de los *tubos* existentes se irán creando nuevos *tubos* hasta que todos los vehículos estén ubicados.~~ ***NO APLICABLE***

1. Obligaciones

El énfasis en este proyecto es **reuso** reiterado de los métodos y clases, en una jerarquía adecuada. See deberán implementar los TDAs primitivos *Pila* y *Cola* en forma genérica. Deberá crear una

jerarquía de clases abstractas y concretas, y heredar apropiadamente las clases del problema. Usar métodos *selectores* (getXXX: void => valor) y *mutadores* (setXXX: valor) => void) para poder solicitar o cambiar el valor da cada atributo. Sólo se puede acceder individualmente a los diversos objetos usando las operaciones elementales de *Pila* y *Cola*.

a) Deberá crear la clase ***Vehículo***, con atributos de *longitud* en metros, *placa* alfanumérica, *modelo* alfanumérico, *año (anyo)* numérico y *color* (rojo, dorado, plateado, gris, azul, negro, verde, blanco), y *etiqueta* numérica que se genera en forma única y progresiva en cada creación de un objeto ***Vehículo.***

b) Deberá crear la clase ***Tubo***, con atributos de *capacidad* y *ocupación* en metros y *etiqueta* numérica que se genera en forma única y progresiva en cada creación de un objeto Tubo.

Dicha clase ***Tubo*** tendrá a su vez como mínimo los métodos públicos:

Estacionar: *Vehículo => void*
Estaciona un vehículo en el tubo, desde el fondo al portal, ajustando la *ocupación*.

Retirar: *void => void*
Retira un vehículo (si existe) del portal del tubo. Si el tubo queda vacío, se destruye.

Cercano: *void => Vehículo*
Devuelve el vehículo (si existe) más cercano al portal del tubo.

Cabe: *Vehículo => boolean.*
Determina si existe espacio suficiente para almacenar el vehículo.

Existe: *atributo* x *valor => boolean.*
Devuelve si existe al menos un vehículo que cumpla la condición "*atributo*==*valor*", donde *atributo* es un atributo del objeto y *valor* lo que se busca.

c) Deberá crear la clase ***Estacionamiento***, con *etiqueta* numérica que se genera en forma única y progresiva en cada creación de un objeto de ese tipo.

Dicha clase ***Estacionamiento*** tendrá a su vez como mínimo los métodos públicos:

Estacionar: *Vehículo => ticket*
Estaciona un vehículo en el estacionamiento, devolviendo como *ticket* la *etiqueta* del *tubo* en que fue estacionado.

Retirar: *placa* x *ticket => Vehículo*
Retira el vehículo (si existe) ~~más cercano al portal del tubo~~.

Existe: *placa* x *ticket => boolean*
Devuelve si existe el vehículo que se busca

Generar: *void => void*
Devuelve un nuevo tubo vacío y lo coloca de último en la secuencia de tubos

Destruir: *void => void*
Destruye el primer tubo.

Vaciar: *etiqueta => void.*
Vacía (si existe) el *tubo* con *etiqueta*, repartiendo secuencialmente los *vehículos* del *tubo* en los otros *tubos*, generando nuevos *tubos* si es necesario.

d) En la clase *Estacionamiento* proveerá un método **ProcesarLlegadas :** *nombreArchivo => void,* que leerá líneas de *eventos* de **nombreArchivo,** que deberá ser procesado secuencialmente. Cada línea a su vez crea un objeto de la clase **Evento** cuyo método principal será **Procesar** : *void => resultado.* El resultado de cada operación deberá escribirse en una línea en un archivo de traza, cuyo formato se proveerá en el Aula Virtual.

Un evento consta de: *Código* \t *valor1* \t *valor2* \t ... \t *valorN* \r según contexto. El código puede ser

C \t *nombre* \r :
Crear Estacionamiento (primera instrucción siempre) y guardar la traza en el archivo *nombre.*

P \t *placa* \t *longitud* \t *modelo* \t *anyo* \t *color* \r :
Estacionar vehículo (crear vehículo y devolver *ticket*)

R \t *placa* \t *ticket*:
Retirar vehículo (devolver *vehículo*)

B \t *selector* \t *valor*:
Buscar vehículos en el estacionamiento que se correspondan a *selector* = *valor*. Devolver secuencia de *vehículos* con *etiquetas* de los tubos.

E \t *placa* \t *ticket*:
Existe vehículo (devolver *sí_existe* o *no_existe*)

~~*X* \t *etiqueta*:~~ ~~El Tubo con *etiqueta* sufre la explosión de una supernova.~~

K \r :
Destruir Estacionamiento, cerrar el archivo de traza y finalizar

Se le proveerá de casos de prueba como diferentes archivos de eventos y los resultados esperados.

ACLARACIONES:

Para asegurar replicabilidad, debe usar el modulo **random**

```
from random import *
```

En el constructor de Estacionamiento, debe llamar al procedimiento *seed* con el valor provisto

```
seed(0.101101)  #inicializa secuencia aleatoria a una semilla conocida
```

El constructor de Tubo debe tener un parámetro *tamanyo*

```
def __init__ (self, tamanyo):
        self.tamanyo = tamanyo
        self.etiqueta = ...
        ...
```

De manera que se pueda pasar como parámetro el tamaño, ya sea aleatorio (cuando se crea normal, usando *randint*) o cuando se copia por destrucción inminente de otro. Sólo se debe crear un nuevo tubo cuando se necesite, dado que por construcción no puede haber tubos vacíos.

```
E = Tubo( randint(A, B))    # randint(A,B) devuelve un entero en el intervalo [A..B] ([5..25]
F = Tubo (E.tamanyo)
```

Entrega:

1) Cada integrante de un Equipo deberá imprimir, firmar y entregar a su profesor la *"Declaración de Autenticidad para Entregas"*, cuya plantilla se encuentra en el Aula Virtual del curso.

2) Deposite una ***carpeta*** que contenga los archivos de código fuente, así como otros que sean necesarios para la ejecución de la aplicación. Todos sus archivos de código deben tener como comentario un encabezado con los nombres y números de carnet de ***ambos*** miembros del equipo.

3) La entrega del proyecto será directamente en el espacio de su grupo en el Aula Virtual **ANTES** de la 1:30 pm del Jueves 5 de Diciembre

4) La corrección de este proyecto se hará mediante un script diseñado a tal fin.

El no cumplimento de los requerimientos podrá resultar en el rechazo de su entrega.

Para la implementación de las etiquetas se recomienda leer sobre variables de clases y variables de instancia. Algunas lecturas por las cuales iniciar:

4a) Diferencias: http://rtomaszewski.blogspot.com/2012/08/difference-between-class-and-instance.html
4b) Asignacion y Busqueda: http://www.python.org/doc/essays/ppt/acm-ws/sld052.htm
4c) Diferencias para implementacion de "constantes": http://stackoverflow.com/questions/12657867/class-variable-vs-instance-variable-python