**Universidad CENFOTEC**



**Estructura de Datos 1.**

**Primer Proyecto Programado**

**Profesor:** Christhian Sibaja.

**Estudiantes:**

Denis Mauricio Granados Ly

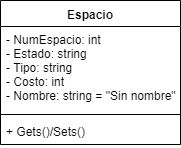
Rafael Alberto Briceño Aguilar.

**Fecha de entrega:** 01/04/19

Cuatrimestre No.1 2019

# TAD Espacio

**Representación abstracta**



**Definición de invariante**

* El objeto Espacio pasa a ser instanciado al momento de crear la lista.
* Los valores de los atributos del objeto varían según el tipo de lista al que sea asignado el objeto.
* Al instanciar el objeto, no se toma en cuenta el atributo Nombre, hasta el momento de realizarse una acción que cambie los valores de los otros atributos.

**Definición de operaciones**

* Espacio: (void) -> Espacio // Constructor por defecto del objeto (Constructora).
* Espacio: (int, string, string, int) -> void // Se define el objeto Espacio (Constructora).
* MostrarEspacio: (Espacio) -> void // Se muestra el contenido de los atributos del objeto (Salida en pantalla).

**Manejo del error**

* Los valores asignados a los atributos dependerán de la lista al que sea asignado el objeto.

# TAD ListaEspacios

**Representación abstracta**

<Espacio1, Espacio2, …, Espacion>, donde Espacio != NULL.

**Definición de la invariante**

* Los elementos de la lista son del tipo Espacio.
* Los elementos de las listas se ordenarán de acuerdo con lo que se le solicite.
* Se manejan contadores para gestionar la cantidad de pagos y montos realizados en la lista.

**Definición de operaciones**

* ListaEspacios: (void) -> ListaEspacios // Constructor por defecto, devuelve una lista vacía. (Constructora)
* ReservarEspacioVIP: (Espacio) -> void // Recibe como parámetro el objeto Espacio para insertarlo de forma ordenada de acuerdo con el número del espacio. El atributo “Estado” del objeto “Espacio” ingresa como “Reservado”. (Modificadora)
* ReservarEspacioGeneral: (Espacio) -> void // Recibe como parámetro el objeto Espacio insertando al inicio. El atributo “Estado” del objeto “Espacio” ingresa como “Reservado”. (Modificadora)
* PagarEspacioReservado: (int) -> void // Recibe como parámetro el número del espacio en el que se hizo la reservación, Cambiando el “Estado” del objeto “Espacio” de “Reservado” a “Pagado”, aumentado el contador de “pagados” de la lista. (Modificadora)
* LiberarReservas: (void) -> void // Modifica todos los espacios con el estado de “Reservado” a “Libre”. (Modificadora)
* EsDisponible: (int) -> bool // Verifica si el numero de espacio recibido existe en la lista. (Analizadora)
* IngresarVIP: (int, string) -> bool // Recibe como parámetros el espacio al que se desea ingresar y el nombre del cliente atendido de la cola para ingresar a la lista como “Pagado”. Retorna un booleano que indica si se realizó el pago. (Modificadora)
* IngresarGeneral: (string) -> Recibe como parámetro el nombre del cliente atendido de la cola para ingresar a la lista como “Pagado”. Retorna un booleano que indica si se realizó el pago. (Modificadora)

**Manejo del error**

* Se validará la longitud de la lista para determinar su límite, de acuerdo con esto se reportará al usuario
* Es responsabilidad del usuario liberar las reservas 30 minutos antes de iniciar la función.
* Al ingresar información para pagar o reservar, que no corresponde a la almacenada en las listas el sistema reportará al usuario.

# TAD Pila

**Representación abstracta**

Espacion, Espacion-1, …, Espacio1 donde Espacio != NULL.

**Definición de la invariante**

* La longitud máxima de las pilas va a ser de 5 espacios.
* Los espacios que se ingresan a la pila se ingresan como reservados.

**Definición de operaciones**

* Pila: (void) -> PilaEspacios // Constructor por defecto, devuelve una pila vacía. (Constructora)
* sumarLongitud: (void) -> void // Suma números enteros a la longitud de la pila. (Modificadora)
* restarLongitud: (void) -> void // Resta números enteros a la longitud de la pila. (Modificadora)
* insertarElem: (Espacio) -> void // Añade elementos de tipo Espacio a la pila. El atributo “Estado” del objeto ingresa como “Reservado”.
* RetirarElemTope: (void) -> Espacio // Retira el primer elemento de la pila y lo retorna. (Modificadora)
* topePila: (void) -> Espacio // Retorna el tope de la pila. (Analizadora)
* esVaciaPila: (void) -> bool // Determina si la pila es vacía. (Analizadora)
* mostrarPila: (void) -> void // Muestra los elementos que se encuentran en la pila. (Impresión en pantalla)
* LiberarReservas: (void) -> void // Retira los elementos de la pila. (Modificadora)

**Manejo del error**

* Al no haber más espacios disponibles en una hilera (Pila), se procede a llenar la siguiente hilera disponible. El sistema se encarga de cambiar de hilera si esto ocurre.
* Al reservar un espacio y todas las hileras están llenas, el sistema reportará al usuario y no lo dejará ingresar más reservas.
* Es responsabilidad del usuario liberar las reservas 30 minutos antes de iniciar la función.
* Al ingresar información para pagar o reservar, que no corresponde a la almacenada en las pilas el sistema reportará al usuario.

# TAD Cola

**Representación abstracta**

Cliente1, Cliente2, …, Clienten-1, Clienten

**Definición de la invariante**

* Cada cliente se va atendiendo según sea el orden de la cola

**Definición de operaciones**

* Cola: (void) -> Cola // Constructor por defecto, devuelve una cola vacía. (Constructora)
* EsVacia: (void) -> bool // Retorna si la cola es vacía. (Analizadora)
* InsertarCliente: (string) -> void // Inserta el nombre de un nuevo cliente a la cola. (Modificadora)
* Atender: (void) -> void // El primer cliente de la cola pasa a comprar un espacio en el teatro. (Modificadora)

**Manejo del error**

* Se reporta al usuario cuando se trate de atender a un cliente y la cola se encuentra vacía.

# Diagrama de clases

