Primer Parcial

Tema 1

14 de Septiembre 2021

Estacionamiento
<<Atributos de clase>> capacidad_maxima = 10 *<<Atributos de Instancia>>* cantidad_autos:entero tarifa_por_hora:real
<<Constructor>> Estacionamiento(tarifa_hr:real) *<<Comandos>>* ocupar():boolean desocupar(horas:real):real *<<Consultas>>* hayDisponible():boolean obtenerCantidadAutos():entero obtenerTarifaPorHora():real equals(Estacionamiento e):boolean toString():String

Restaurante
<<Atributos de clase>> aforo = 60 *<<Atributos de Instancia>>* cantidad_comensales:entero estacionamiento:Estacionamiento precio_por_comensal:real
<<Constructor>> Restaurante(e:Estacionamiento, precio:real) *<<Comandos>>* ocuparMesa(cant_comensales:entero, enAuto:boolean):boolean desocuparMesa(cant_comensales:entero, enAuto:boolean, horas:real):real *<<Consultas>>* obtenerEstacionamiento():Estacionamiento obtenerPrecioPorComensal():real obtenerCantidadComensales():entero equals(Restaurante r):boolean toString():String

La clase Estacionamiento modela los atributos y servicios de una clase que permite mantener el estado actual de un dado estacionamiento. Cada instancia de Estacionamiento mantiene en un momento dado, la cantidad de autos que ocupan el estacionamiento en ese instante y la tarifa por hora. Cuando se crea un estacionamiento la cantidad de autos es 0.

- El comando **`ocupar()`**, si hay lugar disponible en el estacionamiento, incrementa en uno la cantidad de autos estacionados. El método devuelve true si logró estacionar exitosamente un nuevo auto.
- El comando **`desocupar(horas:real)`**, decrementa en uno la cantidad de autos estacionados y devuelve el costo total del estacionamiento. El costo se calcula a partir del precio por hora y la cantidad de horas. Requiere que haya al menos un auto en el estacionamiento.
- La consulta **`hayDisponible()`** devuelve true si hay al menos un espacio disponible en el estacionamiento.

La clase Restaurante modela los atributos y servicios de una clase que permite mantener el estado actual de un dado restaurante. Cada instancia de clase Restaurante se caracteriza por la cantidad de comensales dentro del recinto, el precio fijado por persona y un estacionamiento asociado. La cantidad de comensales en el momento que se crea un Restaurante es 0. El constructor de Restaurante requiere e ligado.

- El comando **`ocuparMesa(cant_comensales:entero, enAuto:boolean)`**, actualiza la cantidad de comensales en el restaurante. Si fueron en auto actualiza además los lugares disponibles en el estacionamiento. El éxito del comando depende de si hay lugar en el restaurante y en caso de que corresponda, si pudo ocupar un lugar en el estacionamiento. Si el comando puede ejecutar exitosamente devuelve true.
- El comando **`desocuparMesa(cant_comensales:entero, enAuto:boolean)`**, actualiza la cantidad de comensales en el recinto. Si fueron en auto actualiza además los lugares disponibles en el estacionamiento. Requiere que haya al menos cant_comensales en el restaurante y, si corresponde, que haya al menos un auto en el estacionamiento. El método devuelve el costo total del servicio, esto es, la suma del costo del estacionamiento (si corresponde) y de la comida en el restaurante.
- La consulta **`equals(Restaurante r)`** es implementada en profundidad. Requiere r ligado.

Dada la descripción de las clases Estacionamiento y Restaurante:

1. Implementar la clase Estacionamiento. Los métodos no descritos son los triviales (`obtenerTarifaPorHora()`, `obtenerCantidadAutos()`, etc) y también deben ser implementados.
2. Implementar el método main en una clase tester para verificar algunos servicios de la clase Estacionamiento. El tester crea dos objetos de clase Estacionamiento con tarifa 15.0 y luego muestra el valor que resulta al enviar el mensaje equals a un estacionamiento con el otro como parámetro. A continuación, envía el mensaje ocupar dos veces a un estacionamiento y luego vuelve a mostrar el valor que resulta al enviar el mensaje equals. Por último muestra en consola el estado interno de cada estacionamiento usando **`toString()`**.
3. Implementar la clase Restaurante. Los métodos no descritos son los triviales (`obtenerEstacionamiento()`, `obtenerPrecioPorComensal()`, etc.) y también deben ser implementados.
4. Comprimir todo el proyecto de BlueJ (las dos clases y el tester) y subir a la plataforma moodle.