Laboratorio de Computadoras Electrónicas

Examen de Unidad: POO **Tema I**

Fecha: _____/_____/_____ Cantidad de hojas: _____

Nombre y apellido:

Curso: Calificación: _____

Ejercicio

Contextualización

Se esta desarrollando un sistema de gestión para un estacionamiento. El desarrollo se lleva a cabo utilizando el paradigma de Programación Orientada a Objetos. Hasta el momento, la estructura del proyecto es la siguiente:

```
src
├── app
│   └── Main.java
├── entidades
│   ├── Espacio.java
│   ├── Estacionamiento.java
│   └── Vehiculo.java
└── enums
    └── TipoVehiculo.java
```

Todas las clases del proyecto ya se encuentran implementadas a excepción de la clase `Estacionamiento`.

Consigna

Implemente la clase `Estacionamiento` considerando las siguientes especificaciones:

Atributos

- `espacios`: un ArrayList de objetos de tipo `Espacio` que representa los espacios disponibles en el estacionamiento.
- `espaciosOcupados`: una variable entera que lleva la cuenta de la cantidad de espacios ocupados en el estacionamiento.
- `CAPACIDAD_MAXIMA`: una constante entera que representa la capacidad máxima del estacionamiento y se corresponde con la cantidad de elementos de `espacios`.
- `TARIFA_MOTO`: una constante entera que representa la tarifa por hora para motos.
- `TARIFA_ESTANDAR`: una constante entera que representa la tarifa por hora para autos y camionetas.

Métodos

- `Estacionamiento(ArrayList<Espacio> espacios, int tarifaMoto, int tarifaEstandar)`: constructor que inicializa los atributos `espacios`[1], `TARIFA_MOTO` y `TARIFA_ESTANDAR`. El atributo `CAPACIDAD_MAXIMA` se inicializa con la cantidad de elementos en `espacios`, y `espaciosOcupados` se inicializa en 0.
- `estacionarVehiculo(Vehiculo vehiculo)`: método que intenta estacionar un vehículo en algún espacio libre del estacionamiento. Si hay un espacio disponible, debe estacionar el vehículo en ese espacio, incrementar `espaciosOcupados` y devolver `true`. Si no hay espacios disponibles, debe devolver `false`.

[1] Considere que los números de espacios en la lista están completos y ordenados.

- `liberarEspacio(int numeroEspacio)`: método que debe liberar el espacio ocupado `numeroEspacio` y decrementar `espaciosOcupados`.
- `obtenerCostoAcumulado(int numeroEspacio)`: método que calcula y devuelve el costo acumulado por el tiempo que un vehículo ha estado estacionado en un espacio específico, basado en la tarifa correspondiente al tipo de vehículo.

Documentación

Clase enumerada TipoVehiculo

Un TipoVehiculo representa los diferentes tipos de vehículos que pueden estacionarse en el estacionamiento.

Resumen de constantes

Constante	Descripción
`MOTO`	Indica que el vehículo es una moto.
`VEHICULO_ESTANDAR`	Indica que el vehículo es un automóvil o camioneta.

Clase Espacio

La clase Espacio representa un espacio individual dentro del estacionamiento. Todos los espacios están numerados secuencialmente comenzando desde 1. El numero de espacio se asigna automáticamente al momento de crear un objeto Espacio y no puede ser modificado posteriormente.

Constructores

Constructor	Descripción
`Espacio()`	Crea un espacio vacío con su numero correspondiente.

Resumen de métodos públicos

Retorno	Método	Descripción
`boolean`	`estaOcupado()`	Devuelve `true` si el espacio está ocupado por un vehículo; de lo contrario, devuelve `false`.
`int`	`obtenerNumeroEspacio()`	Devuelve el número del espacio.
`boolean`	`estacionarVehiculo(Vehiculo vehiculo)`	Intenta estacionar un vehículo en el espacio. Devuelve `true` si se logra estacionar; de lo contrario, devuelve `false`.
`void`	`liberarEspacio()`	Elimina el vehículo del espacio, dejándolo libre.
`TipoVehiculo`	`obtenerTipoVehiculo()`	Devuelve el tipo de vehículo que está estacionado en el espacio.
`int`	`obtenerTiempoEstacionado()`	Devuelve el tiempo en horas que el vehículo ha estado estacionado en el espacio.