RETO 1 – PROGRAMACIÓN BÁSICA

VARIANTE 3

El Valle de Aburrá en su proceso de convertirse en el Valle del software ha decidido mejorar en cuanto a tecnología los autobuses de la ciudad, aportando a la modernización de la ciudad.

La ciudad dotará a los autobuses de aire acondicionado y Wifi para sus pasajeros, además de que aumentará la seguridad de los pasajeros en cuanto a los siguientes aspectos:

- El conductor no podrá recoger ni dejar bajar un pasajero si el autobús está en marcha.
- El autobús no podrá avanzar hasta que el conductor cierre la puerta después de recoger un pasajero.

También se agregará una tecnología que regule el número de pasajeros (Dependiendo si hay o no medidas restrictivas de movilidad por el Covid – 19), por lo que, si el autobús está completamente lleno, el conductor solo podrá abrir la puerta para dejar bajar uno o más pasajeros, y además de que el sistema cobrará el precio del pasaje dependiendo del estrato socioeconómico del pasajero.

Esta tecnología de administración se desarrollará en Java.

Usted ha sido contratado como Java Expert Developer, porque ha logrado demostrar habilidades de desarrollo en este lenguaje de programación y se le ha concedido implementar la clase correspondiente al autobús.

La empresa a cargo de esta implementación le da las siguientes observaciones sobre el funcionamiento del autobús y el sistema de seguridad:

1. El Wifi y el aire acondicionado estarán apagados siempre que el motor esté apagado.
2. La puerta del autobús debe permanecer cerrada siempre que esté en marcha.
3. Para poder que una persona se baje, la puerta debe estar abierta, y en consecuencia el autobús no puede estar en marcha.

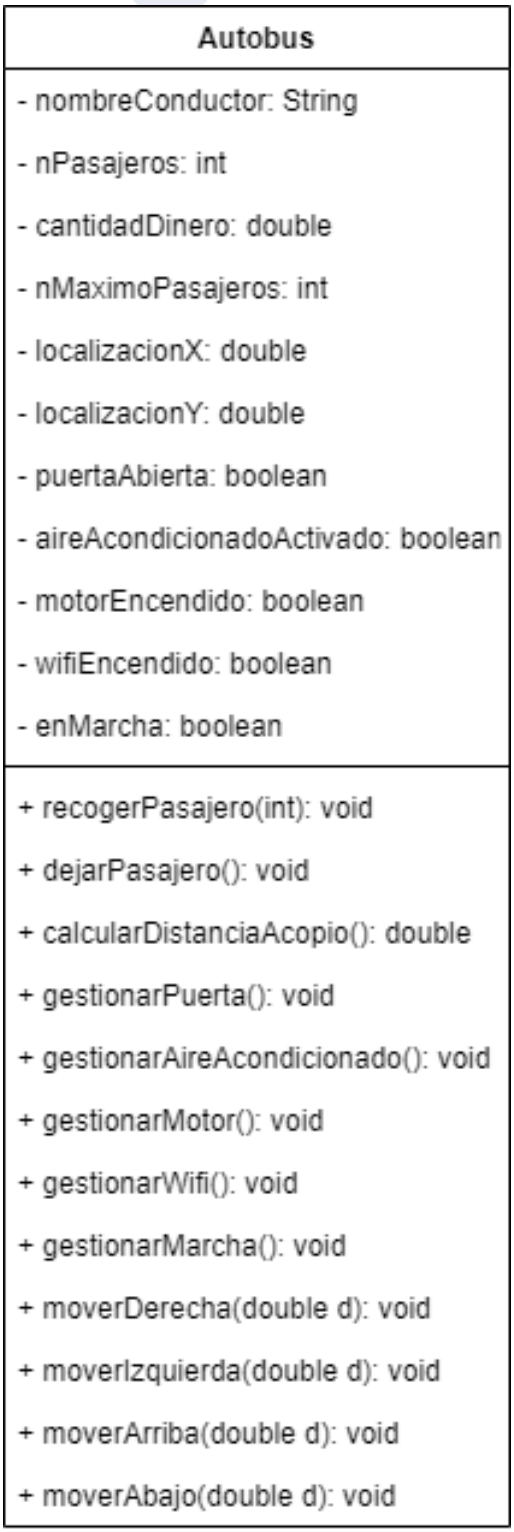
4. Para poder que una persona se suba, la puerta debe estar abierta, y en consecuencia el autobús no puede estar en marcha.
5. Para poder que el autobús esté en marcha, el motor deberá estar encendido (Pero si el motor está encendido no necesariamente debe estar en marcha).
6. Las tarifas son las siguientes:
    a. Para estratos 0, 1 y 2, el pasaje se cobra a 1500 pesos colombianos.
    b. Para estratos 3 y 4, el pasaje se cobra a 2600 pesos.
    c. Para estratos 5 y 6, el pasaje se cobra a 3000 pesos.
7. El conductor podrá saber a qué distancia (En kilómetros) se encuentra del acopio, gracias a este gestor.

Para facilitar la implementación de la clase Autobus, el equipo de Ingeniería de software le hace entrega del diagrama de clases del gestor de funcionalidades de un autobús, recuerde que los métodos relacionados a los getters y setters son obviados en el diagrama de clases, pero deberán ser incluidos en el código (Estos métodos deberán ser creados con el estándar camel case: Por ejemplo, si el atributo se llama nombreConductor, sus métodos correspondientes a get y set serían getNombreConductor y setNombreConductor).

Los getters de las variables booleanas son especiales, por ejemplo, si el atributo que contiene un dato de tipo booleano se llama `puertaAbierta`, su "getter" correspondiente es `isPuertaAbierta()`.

Hay 2 getters y setters que romperán este estándar:

- Para la variable `nPasajeros`:
    - `getnPasajeros()`
    - `setnPasajeros(int nPasajeros)`
- Para la variable `nMaximoPasajeros`:
    - `getnMaximoPasajeros()`
    - `setnMaximoPasajeros(int nMaximoPasajeros)`



Además del diagrama, el equipo de Ingeniería entrega esta documentación para comprender mejor los elementos del diagrama:

Clase Autobus

Atributos

NOMBRE	TIPO DATO	CONCEPTO	INICIALIZACIÓN
`nombreConductor`	`String`	Nombre del conductor	En el método constructor
`nPasajeros`	`int`	Número de pasajeros que hay dentro del bus	Debe estar inicializado en 0
`cantidadDinero`	`double`	Cantidad de dinero recogida por el cobro del pasaje	Debe estar inicializado en 0
`nMaximoPasajeros`	`int`	Cantidad máxima de pasajeros que pueden estar dentro del autobús	En el método constructor
`localizacionX`	`double`	Coordenada x del autobús	Debe estar inicializada en 0
`localizacionY`	`double`	Coordenada y del autobús	Debe estar inicializada en 0
`puertaAbierta`	`boolean`	Cantidad de vida que tiene el personaje durante la partida	En el método constructor
`aireAcondicionadoActivado`	`boolean`	`true` si el aire acondicionado está encendido y `false` en caso de que no lo esté	Debe estar inicializado en `false`
`motorEncendido`	`boolean`	`true` si el motor está encendido y `false` en caso de que no lo esté	Debe estar inicializado en `false`
`wifiEncendido`	`boolean`	`true` si el wifi está encendido y `false` en caso de que no lo esté	Debe estar inicializado en `false`

`enMarcha`	`boolean`	`true` si el autobús está en marcha y `false` en caso de que no lo esté	Debe estar inicializado en `false`

Métodos

NOMBRE	TIPO RETORNO	PARÁMETROS	CONCEPTO
`recogerPasajero`	`void`	`int estrato`: Estrato de la persona que recogió	Suma el costo del pasaje pagado por el pasajero a `cantidadDinero`
`dejarPasajero`	`void`	No recibe	Resta 1 a `nPasajeros`
`calcularDistanciaAcopio`	`double`	No recibe	Retorna la distancia entre el origen de coordenadas y el punto en el que se encuentra el autobús.
`gestionarPuerta`	`void`	No recibe	Abre la puerta si está cerrada al invocar este método y viceversa (El abrir o cerrar la puerta no se ve afectado por el estado encendido del motor)
`gestionarAireAcondicionado`	`void`	No recibe	Enciende el aire acondicionado si está apagado al invocar este método y viceversa
`gestionarMotor`	`void`	No recibe	Enciende el motor si está apagado al

			invocar este método y viceversa
`gestionarWifi`	`void`	No recibe	Enciende el wifi si está apagado al invocar este método y viceversa
`gestionarMarcha`	`void`	No recibe	Activa la marcha del autobús si está parado al invocar este método y viceversa
`moverDerecha`	`void`	`double d`: Cantidad de km a mover el autobús a la derecha.	Suma d a `localizacionX`
`moverIzquierda`	`void`	`double d`: Cantidad de km a mover el autobús a la izquierda.	Resta d a `localizacionX`
`moverArriba`	`void`	`double d`: Cantidad de km a mover el autobús hacia arriba.	Suma d a `localizacionY`
`moverAbajo`	`void`	`double d`: Cantidad de km a mover el autobús hacia abajo.	Resta d a `localizacionY`

PRECISIONES

1. No hay métodos estáticos.
2. El método constructor debe asignar **SOLAMENTE** los atributos `nombreConductor`, `puertaAbierta` y `nMaximoPasajeros`.
3. Los atributos `nPasajeros`, `cantidadDinero`, `localizacionX` y `localizacionY` deben ser inicializados en `0` y los atributos

`aireAcondicionadoActivado`, `motorEncendido`, `wifiEncendido` y `enMarcha` deben ser inicializados en `false`.

4. El autobús no podrá cambiar sus coordenadas geográficas mientras el autobús no se encuentre en marcha.

TAREAS

- En el archivo preconstruido en la plataforma Moodle, implementar la clase especificada en el diagrama de clases, teniendo en cuenta las precisiones dadas por el equipo de Ingeniería de software.
- Los nombres de los métodos y atributos **DEBEN** ser nombrados tal y como aparecen en el diagrama de clases.
- Usted **NO** debe solicitar datos por teclado, ni programar un método `main`, tampoco use `Java Package`, usted está solamente encargado de la construcción de la clase.

EJEMPLO

El calificador automático hará las veces de jugador, y será quien evalúe la experiencia de usuario que tuvo al jugar con su personaje:

1. El calificador recibe un autobús

```
Autobus camion1 = new Autobus("Pepe", 30, false);
```

NOMBRE	CONTENIDO
nombreConductor	"Pepe"
nPasajeros	0
cantidadDinero	0.0
nMaximoPasajeros	30
localizacionX	0.0
localizacionY	0.0
puertaAbierta	false
aireAcondicionadoActivado	false
motorEncendido	false
wifiEncendido	false

enMarcha	false

2. El calificador intenta mover el autobús 5 km a la derecha

```
camion1.moverDerecha(5);
```

NOMBRE	CONTENIDO
nombreConductor	"Pepe"
nPasajeros	0
cantidadDinero	0.0
nMaximoPasajeros	30
localizacionX	0.0
localizacionY	0.0
puertaAbierta	false
aireAcondicionadoActivado	false
motorEncendido	false
wifiEncendido	false
enMarcha	false

Note que el calificador **NO** pudo hacer el movimiento del autobús porque el autobús **NO** se encuentra en marcha.

3. El calificador intenta encender el Wifi, el aire acondicionado, y poner el autobús en marcha.

```
camion1.gestionarWifi();
camion1.gestionarAireAcondicionado();
camion1.gestionarMarcha();
```

NOMBRE	CONTENIDO
nombreConductor	"Pepe"

nPasajeros	0
cantidadDinero	0.0
nMaximoPasajeros	30
localizacionX	0.0
localizacionY	0.0
puertaAbierta	false
aireAcondicionadoActivado	false
motorEncendido	false
wifiEncendido	false
enMarcha	false

Note que mientras el motor está desactivado, el calificador **NO** podrá encender nada de lo mencionado.

4. El calificador enciende el motor, pone el autobús en marcha, y mueve el autobús 5 km a la derecha

```
camion1.gestionarMotor();
camion1.gestionarMarcha();
camion1.moverDerecha(5);
```

NOMBRE	CONTENIDO
nombreConductor	"Pepe"
nPasajeros	0
cantidadDinero	0.0
nMaximoPasajeros	30
localizacionX	5.0
localizacionY	0.0
puertaAbierta	false

aireAcondicionadoActivado	false
motorEncendido	true
wifiEncendido	false
enMarcha	true

5. El calificador intenta recoger un pasajero de estrato 2, intenta abrir la puerta, mueve el autobús 10 km hacia arriba

```
camion1.recogerPasajero(2);
camion1.gestionarPuerta();
camion1.moverArriba(10);
```

NOMBRE	CONTENIDO
nombreConductor	"Pepe"
nPasajeros	0
cantidadDinero	0.0
nMaximoPasajeros	30
localizacionX	5.0
localizacionY	10.0
puertaAbierta	false
aireAcondicionadoActivado	false
motorEncendido	true
wifiEncendido	false
enMarcha	true

Note que el calificador no puede recoger al pasajero, ni abrir la puerta porque el autobús continúa en marcha.

6. El calificador frena el autobús, abre la puerta y recoge 3 pasajeros de estratos 1, 4 y 6 respectivamente.

```
camion1.gestionarMarcha();
camion1.gestionarPuerta();
camion1.recogerPasajero(1);
camion1.recogerPasajero(4);
camion1.recogerPasajero(6);
```

NOMBRE	CONTENIDO
nombreConductor	"Pepe"
nPasajeros	3
cantidadDinero	7100.0
nMaximoPasajeros	30
localizacionX	5.0
localizacionY	10.0
puertaAbierta	true
aireAcondicionadoActivado	false
motorEncendido	true
wifiEncendido	false
enMarcha	false

7. El calificador enciende el wifi, el aire acondicionado, intenta poner en marcha el autobús, y lo intenta mover 2 km a la izquierda

```
camion1.gestionarWifi();
camion1.gestionarAireAcondicionado();
camion1.gestionarMarcha();
camion1.moverIzquierda(2);
```

NOMBRE	CONTENIDO
nombreConductor	"Pepe"
nPasajeros	3
cantidadDinero	7100.0

nMaximoPasajeros	30
localizacionX	5.0
localizacionY	10.0
puertaAbierta	true
aireAcondicionadoActivado	true
motorEncendido	true
wifiEncendido	true
enMarcha	false

8. El calificador deja un pasajero, apaga el motor y calcula qué tan lejos está del acopio

```
camion1.dejarPasajero();
camion1.gestionarMotor();
camion1.calcularDistanciaAcopio();
```

NOMBRE	CONTENIDO
nombreConductor	"Pepe"
nPasajeros	2
cantidadDinero	7100.0
nMaximoPasajeros	30
localizacionX	5.0
localizacionY	10.0
puertaAbierta	true
aireAcondicionadoActivado	false
motorEncendido	false
wifiEncendido	false
enMarcha	false

La salida esperada es `11.180339887498949` (Se considerará un 1% de error en el cálculo de este valor, debido a aproximaciones que usted pueda realizar).

La distancia con respecto al origen de coordenadas se calcula de la siguiente forma:

$$\sqrt{localizacionX^2 + localizacionY^2}$$

Para este caso en particular: $\sqrt{(5)^2 + (10)^2} = 11.180339887498949$

Nota: En caso tal de que el autobús estuviera en marcha, también se debía frenar, un autobús no va en marcha si el motor está apagado (Para este caso).

El calificador realizará varios viajes antes de calificar su código, y mostrará en la consola si sus resultados finales coinciden con los esperados.