



RETO 2 – PROGRAMACIÓN BÁSICA

VARIANTE 3

Medellín en aras de mantener el título de la ciudad más innovadora del mundo ha decidido mejorar la tecnología del transporte público, aportando a su modernización.

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

1. Para autobuses:
 - a. El conductor no podrá recoger ni dejar bajar un pasajero si el autobús está en marcha.
 - b. El autobús no podrá avanzar hasta que el conductor cierre la puerta después de recoger un pasajero.
2. Para taxis:
 - a. El taxi no podrá avanzar mientras el seguro de las puertas no esté activado (Cuando hay pasajeros a bordo).
 - b. Habrá un botón de pánico para el pasajero, este botón detendrá la marcha del taxi, y desbloqueará los seguros.

También se agregará una tecnología que regule el número de pasajeros (Dependiendo si hay o no medidas restrictivas de movilidad por el Covid - 19 y solo aplica para autobuses), 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 las clases correspondientes a vehículo, autobús y taxi.

La empresa a cargo de esta implementación le da las siguientes observaciones sobre el funcionamiento de un vehículo:





1. La marcha estará detenida, el Wifi y el aire acondicionado estarán apagados siempre que el motor esté apagado.
2. Para poder que el vehículo esté en marcha, el motor deberá estar encendido (Pero si el motor está encendido no necesariamente debe estar en marcha).

Funcionamiento de un autobús:

1. La puerta del autobús debe permanecer cerrada siempre que esté en marcha (No podrá moverse mientras esta esté abierta).
2. Para poder que una persona se baje, la puerta debe estar abierta, y en consecuencia el autobús no puede estar en marcha.
3. Para poder que una persona se suba, la puerta debe estar abierta, y en consecuencia el autobús no puede estar en marcha.
4. 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.

Funcionamiento de un taxi:

1. Si hay pasajeros dentro del taxi, es necesario que los seguros de las puertas estén activados para que el taxi pueda estar en marcha.
2. Cuando un pasajero presiona el botón de pánico, detiene la marcha del taxi, desactiva los seguros de las puertas, deja el pasajero (Sin cobro alguno).
3. El cobro de una carrera de taxi se hará del siguiente modo:
 - a. 3000 pesos de base (Cuando se enciende el taxímetro comienza el conteo en 3000 pesos).
 - b. 2300 por cada km recorrido.

Es decir, la fórmula a seguir es: $3000 + 2300 \times \text{distanciaRecorrida}$

Para facilitar la implementación de las clases Vehículo, Autobus y Taxi, el equipo de Ingeniería de software le hace entrega del diagrama de clases del gestor de funcionalidades, recuerde que los métodos relacionados a los *setters* y *getters* 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`).

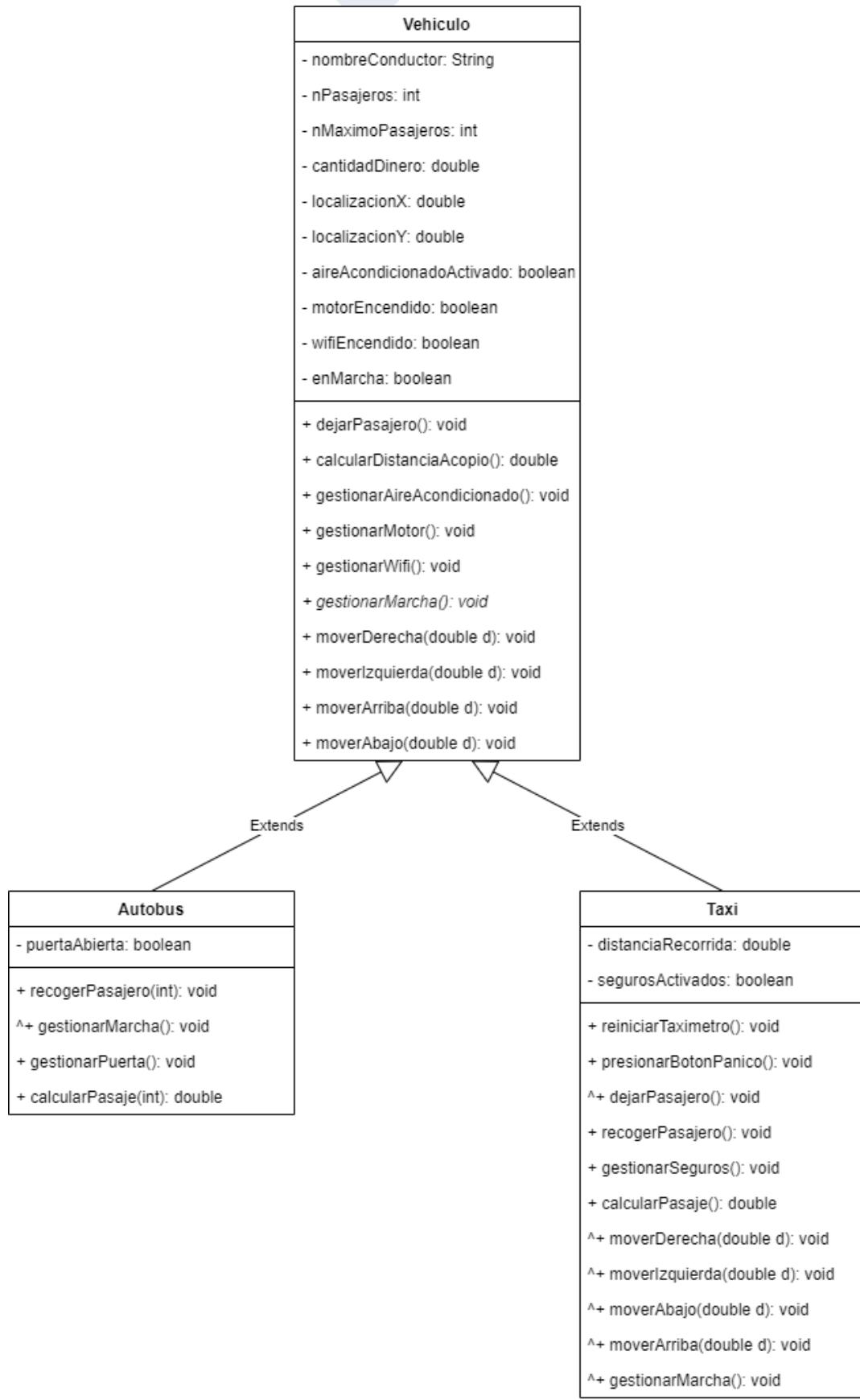


Los *getters* de las variables booleanas son especiales, por ejemplo, si el atributo que contiene un dato de tipo booleano se llama `puertaAbierta`, su “getter” correspondiente es `isPuertaAbierta()`.

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 Vehiculo

Atributos

| NOMBRE | TIPO DATO | CONCEPTO | INICIALIZACIÓN |
|---------------------------|-----------|--|----------------------------------|
| nombreConductor | String | Nombre del conductor | En el método constructor |
| nPasajeros | int | Número de pasajeros que hay dentro del vehículo | Debe estar inicializado en 0 |
| cantidadDinero | double | Cantidad de dinero recogida por el cobro de pasajes | Debe estar inicializado en 0 |
| nMaximoPasajeros | int | Cantidad máxima de pasajeros que pueden estar dentro del vehículo | En el método constructor |
| localizacionX | double | Coordinada x del vehículo | Debe estar inicializada en 0 |
| localizacionY | double | Coordinada y del vehículo | Debe estar inicializada en 0 |
| 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 vehículo está en marcha y false en caso de que no lo esté | Debe estar inicializado en false |
|----------|---------|--|----------------------------------|

Métodos

| NOMBRE | TIPO RETORNO | PARÁMETROS | CONCEPTO |
|----------------------------|--------------|------------|--|
| dejarPasajero | void | No recibe | Resta 1 a nPasajeros |
| calcularDistanciaA copio | double | No recibe | Retorna la distancia entre el origen de coordenadas y el punto en el que se encuentra el vehículo. |
| gestionarAireAcondicionado | void | No recibe | Si el motor está encendido y el aire acondicionado está apagado, lo enciende. Para cualquier otro caso lo apaga. |
| gestionarMotor | void | No recibe | Si el motor está apagado, lo enciende. Para cualquier otro caso lo apaga (Y además se debe apagar el aire acondicionado, el Wifi, y detener la marcha). |
| gestionarWifi | void | No recibe | Si el motor está encendido y el Wifi está apagado, lo enciende. Para cualquier otro caso lo apaga. |
| gestionarMarcha | void | No recibe | Método abstracto |





| | | | |
|----------------|------|--|---|
| moverDerecha | void | double d: Cantidad de km a mover el vehículo a la derecha. | Si el vehículo está en marcha, suma <i>d</i> a localizacionX |
| moverIzquierda | void | double d: Cantidad de km a mover el vehículo a la izquierda. | Si el vehículo está en marcha, resta <i>d</i> a localizacionX |
| moverArriba | void | double d: Cantidad de km a mover el vehículo hacia arriba. | Si el vehículo está en marcha, suma <i>d</i> a localizacionY |
| moverAbajo | void | double d: Cantidad de km a mover el vehículo hacia abajo. | Si el vehículo está en marcha, resta <i>d</i> a localizacionY |





Clase Autobus (Hija / Hereda de Vehiculo)

Atributos

| NOMBRE | TIPO DATO | CONCEPTO | INICIALIZACIÓN |
|---------------|-----------|--|----------------|
| puertaAbierta | boolean | <p>true si la puerta del vehículo está abierta y false en caso de que no lo esté</p> <p>(La puerta no se puede abrir mientras el autobús se encuentre en marcha)</p> | false |

Métodos

| NOMBRE | TIPO RETORNO | PARÁMETROS | CONCEPTO |
|-----------------|--------------|--|--|
| recogerPasajero | void | int estrato: Estrato de la persona que recogió | Suma this.calcularPasaje (estrato) a cantidadDinero y suma 1 a nPasajeros (nPasajeros no puede exceder la cantidad de nMaximoPasajeros) |
| gestionarPuerta | void | No recibe | <p>Si la puerta está cerrada y el autobús NO está en marcha, la abre.</p> <p>Para cualquier otro caso se cierra la puerta.</p> <p>(El funcionamiento de la puerta no se ve afectado por el estado del motor)</p> |
| gestionarMarcha | void | No recibe | Si el autobús está detenido y con la puerta |





| | | | |
|----------------|--------|--|--|
| | | | <p>cerrada, activa la marcha.</p> <p>Para cualquier otro caso se detiene el autobús.</p> |
| calcularPasaje | double | int estrato: Estrato del pasajero que recogió. | Retorna el valor del pasaje del pasajero. |

Clase Taxi (Hija / Hereda de Vehiculo)

Atributos

| NOMBRE | TIPO DATO | CONCEPTO | INICIALIZACIÓN |
|--------------------|-----------|---|----------------------------------|
| distanciaRecorrida | double | Distancia en kilómetros que ha recorrido el taxi con el pasajero que lleva en ese instante | Debe estar inicializado en 0 |
| segurosActivados | boolean | true si el taxi tiene los seguros de las puertas activados y false en caso de que no lo estén | Debe estar inicializado en false |

Métodos

| NOMBRE | TIPO RETORNO | PARÁMETROS | CONCEPTO |
|---------------------|--------------|------------|--|
| reiniciarTaximetro | void | No recibe | Asigna 0 a la variable distanciaRecorrida |
| presionarBotonPanic | void | No recibe | Detiene la marcha del taxi, y asigna false a segurosActivados y deja al pasajero (Sin cobro) |
| dejarPasajero | void | No recibe | Resta 1 a nPasajeros, reinicia el taxímetro y |





| | | | |
|-----------------|------|---|---|
| | | | suma this.calcularPasaje() () a cantidadDinero |
| recogerPasajero | void | No recibe | Suma 1 a nPasajeros |
| gestionarMarcha | void | No recibe | Si el taxi está detenido y con los seguros activados, activa la marcha. Detiene la marcha si se presiona el botón de pánico. Para cualquier otro caso se detiene el taxi. |
| moverDerecha | void | double d: Cantidad de km a mover el autobús a la derecha. | Si el vehículo está en marcha, suma d a localizacionX y si hay pasajeros, suma d a distanciaRecorrida |
| moverIzquierda | void | double d: Cantidad de km a mover el autobús a la izquierda. | Si el vehículo está en marcha, resta d a localizacionX y si hay pasajeros, suma d a distanciaRecorrida |
| moverArriba | void | double d: Cantidad de km a mover el autobús hacia arriba. | Si el vehículo está en marcha, suma d a localizacionY y si hay pasajeros, suma d a distanciaRecorrida |
| moverAbajo | void | double d: Cantidad de km a mover el autobús hacia abajo. | Si el vehículo está en marcha, resta d a localizacionY y si hay pasajeros, suma d a distanciaRecorrida |





| | | | |
|------------------|--------|-----------|--|
| calcularPasaje | double | No recibe | Retorna el valor del pasaje del pasajero. |
| gestionarSeguros | void | No recibe | Si el taxi está detenido y con los seguros activados, estos se desactivan Si el pasajero presionó el botón de pánico, estos se desactivan Para cualquier otro caso se activan. |

PRECISIONES

1. Vehiculo es una clase abstracta (Existe un método abstracto, ya viene cargado en el VPL, no lo modifique).
2. Solo calcularPasaje de la clase hija Autobus es un método estático.
3. Los taxis pueden llevar hasta 5 personas, pero tenga en cuenta que en la realidad todas son conocidas (Un grupo familiar, un grupo de amigos, etc.) por lo que se debe considerar el número de personas que hay dentro de un taxi como una sola persona, y el número máximo de personas que puede llevar un taxi es 1, esto significa que, si vas en un taxi con tus padres y tu hermano, solo representan una persona.
4. Ni el autobús ni el taxi podrán cambiar sus coordenadas geográficas mientras no se encuentren 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 conductor, y será quien evalúe la modernización realizada:

Evaluación de un autobús:

1. Se le hace entrega al calificador de un autobús:

```
Autobus a = new Autobus("Pepe", 30);
```

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

2. El calificador intenta recoger un pasajero estrato 2:

```
a.recogerPasajero(2);
```

| NOMBRE | CONTENIDO |
|------------------|-----------|
| nombreConductor | "Pepe" |
| nPasajeros | 0 |
| cantidadDinero | 0 |
| nMaximoPasajeros | 30 |





| | |
|---------------------------|-------|
| localizacionX | 0 |
| localizacionY | 0 |
| puertaAbierta | false |
| aireAcondicionadoActivado | false |
| motorEncendido | False |
| wifiEncendido | false |
| enMarcha | false |

Pero como no está la puerta abierta, no lo puede recoger.

3. El calificador abre la puerta e intenta poner el autobús en marcha:

```
a.gestionarPuerta();  
a.gestionarMarcha();  
...
```

| NOMBRE | CONTENIDO |
|---------------------------|-----------|
| nombreConductor | "Pepe" |
| nPasajeros | 0 |
| cantidadDinero | 0 |
| nMaximoPasajeros | 30 |
| localizacionX | 0 |
| localizacionY | 0 |
| puertaAbierta | true |
| aireAcondicionadoActivado | false |
| motorEncendido | False |
| wifiEncendido | false |
| enMarcha | false |

Como el autobús está con la puerta abierta, no se puede poner en marcha.





4. El calificador recoge un pasajero estrato 2, intenta poner el autobús en marcha y cierra la puerta:

```
a.recogerPasajero(2);  
a.gestionarMarcha();  
a.gestionarPuerta();
```

| NOMBRE | CONTENIDO |
|---------------------------|-----------|
| nombreConductor | "Pepe" |
| nPasajeros | 1 |
| cantidadDinero | 1500 |
| nMaximoPasajeros | 30 |
| localizacionX | 0 |
| localizacionY | 0 |
| puertaAbierta | false |
| aireAcondicionadoActivado | false |
| motorEncendido | false |
| wifiEncendido | false |
| enMarcha | false |

Como el autobús estaba con la puerta abierta, no se puede poner en marcha. (Primero debe cerrar la puerta antes de ponerse en marcha).

5. El calificador intenta mover el autobús 2 kilómetros hacia arriba, 5 a la derecha, enciende el Wifi, el aire acondicionado:

```
a.moverArriba(2);  
a.moverDerecha(5);  
a.gestionarWifi();  
a.gestionarAireAcondicionado();
```

| NOMBRE | CONTENIDO |
|-----------------|-----------|
| nombreConductor | "Pepe" |





| | |
|---------------------------|-------|
| nPasajeros | 1 |
| cantidadDinero | 1500 |
| nMaximoPasajeros | 30 |
| localizacionX | 0 |
| localizacionY | 0 |
| puertaAbierta | false |
| aireAcondicionadoActivado | false |
| motorEncendido | false |
| wifiEncendido | false |
| enMarcha | false |

Todo sigue igual, pues el autobús estaba con el motor apagado.

6. El calificador enciende el motor, e intenta moverse a la izquierda:

```
a.gestionarMotor();  
a.moverIzquierda(2);
```

| NOMBRE | CONTENIDO |
|---------------------------|-----------|
| nombreConductor | "Pepe" |
| nPasajeros | 1 |
| cantidadDinero | 1500 |
| nMaximoPasajeros | 30 |
| localizacionX | 0 |
| localizacionY | 0 |
| puertaAbierta | false |
| aireAcondicionadoActivado | false |
| motorEncendido | true |





| | |
|---------------|-------|
| wifiEncendido | false |
| enMarcha | false |

Solo se enciende el motor, porque como no está en marcha, no puede moverse.

Evaluación de un taxi:

1. Se le hace entrega al calificador de un taxi:

```
Taxi b = new Taxi("Pepe");
```

| NOMBRE | CONTENIDO |
|---------------------------|-----------|
| nombreConductor | "Pepito" |
| nPasajeros | 0 |
| cantidadDinero | 0 |
| nMaximoPasajeros | 1 |
| localizacionX | 0 |
| localizacionY | 0 |
| aireAcondicionadoActivado | false |
| motorEncendido | false |
| wifiEncendido | false |
| enMarcha | false |
| distanciaRecorrida | 0 |
| segurosActivados | false |

2. El calificador enciende el motor, intenta poner en marcha el taxi, y recoge un pasajero:

```
b.gestionarMotor();  
b.gestionarMarcha();  
b.recogerPasajero();
```





| NOMBRE | CONTENIDO |
|---------------------------|-----------|
| nombreConductor | "Pepito" |
| nPasajeros | 1 |
| cantidadDinero | 0 |
| nMaximoPasajeros | 1 |
| localizacionX | 0 |
| localizacionY | 0 |
| aireAcondicionadoActivado | false |
| motorEncendido | true |
| wifiEncendido | false |
| enMarcha | false |
| distanciaRecorrida | 0 |
| segurosActivados | false |

El taxi no se puso en marcha, ya que los seguros estaban desactivados pero pudo recoger el pasajero porque los seguros estaban desactivados y no estaba en marcha.

3. El calificador intenta poner en marcha el taxi, intenta recoger un nuevo pasajero, intenta mover el taxi, activa los seguros, e intenta mover nuevamente el taxi:

```
b.gestionarMarcha();  
b.recogerPasajero();  
b.moverArriba(10);  
b.moverDerecha(12);  
b.gestionarSeguros();  
b.moverAbajo(10);  
b.moverIzquierda(13);
```

| NOMBRE | CONTENIDO |
|-----------------|-----------|
| nombreConductor | "Pepito" |
| nPasajeros | 1 |





| | |
|---------------------------|-------|
| cantidadDinero | 0 |
| nMaximoPasajeros | 1 |
| localizacionX | 0 |
| localizacionY | 0 |
| aireAcondicionadoActivado | false |
| motorEncendido | true |
| wifiEncendido | false |
| enMarcha | false |
| distanciaRecorrida | 0 |
| segurosActivados | true |

El taxi no se pudo poner en marcha porque tenía el seguro desactivado, no puede recoger más pasajeros porque ya hay uno a bordo, no puede mover el taxi porque la marcha está detenida, y activa los seguros.

4. El calificador enciende el aire acondicionado, el wifi, apaga el motor, el pasajero presiona el botón de pánico y se baja (Recuerde que cuando se usa el botón de pánico, el pasajero no paga).

```
b.gestionarAireAcondicionado();  
b.gestionarWifi();  
b.gestionarMotor();  
b.presionarBotonPanico();
```

| NOMBRE | CONTENIDO |
|---------------------------|-----------|
| nombreConductor | "Pepito" |
| nPasajeros | 0 |
| cantidadDinero | 0 |
| nMaximoPasajeros | 1 |
| localizacionX | 0 |
| localizacionY | 0 |
| aireAcondicionadoActivado | false |





| | |
|--------------------|-------|
| motorEncendido | false |
| wifiEncendido | false |
| enMarcha | false |
| distanciaRecorrida | 0 |
| segurosActivados | false |

El wifi y aire acondicionado se apagan cuando el motor se apaga, el pasajero se baja sin pagar cuando presiona el botón de pánico.

