Instrucciones Miniproyecto 2

Descripción

Debe construir una simulación del funcionamiento de vehículos usando orientación a objetos.

El trabajo a realizar

Se entregará un archivo MP2.zip, con dos archivos: `parametros.py` y `main.py`. Debe trabajar únicamente sobre el archivo `main.py`.

El código que se entrega contiene la función `seleccionar` y la clase `Rueda` que **ya están completamente implementadas**, además de todo el manejo del menú (últimas líneas del código).

ESTA CLASE ESTA LISTA NO HAY QUE MODIFICAR NADA

La clase `Rueda` tiene los siguientes atributos:

- `resistencia_actual`: Es la resistencia actual de la rueda. Se inicializa como un int random entre los números almacenados en la lista `RESISTENCIA` en el archivo `parametros.py`.
- `resistencia_total`: Corresponde a la resistencia total de la rueda. Se inicializa con el valor inicial del atributo `resistencia_actual`.
- `estado`: Es el estado en el que se encuentra la rueda. Se inicializa en un string que diga “Perfecto”.

Además tiene los siguientes métodos:

- `gastar(accion)`: Recibe el argumento `accion` que corresponde a un string que puede tener el valor “acelerar” o “frenar”. Dependiendo de la acción, el valor del atributo `resistencia_actual` de la rueda disminuye. No retorna nada.
- `actualizar_estado()`: No recibe argumentos. Actualiza el estado según la nueva `resistencial_actual` en comparación a la `resistencia_total` de la rueda. No retorna nada.

Solo modifica las partes del código indicadas en el archivo `main.py` según lo descrito en el enunciado.

Usando orientación a objetos debe realizar lo siguiente:

1. Definir una clase `Automovil` que contenga los siguientes atributos:
   - `kilometraje`: Kilometraje del vehículo. Se recibe como parámetro de inicialización. Debe ser un atributo privado. (Se mide en km)
   - `ano`: Año de fabricación. Se recibe como parámetro de inicialización.
   - `ruedas`: Lista de objetos de tipo `Rueda`. Se inicializa como una lista vacía.
   - `aceleracion`: Aceleración del vehículo. Se inicializa en 0. (Se mide en m/s^2)
   - `velocidad`: Velocidad del vehículo. Se inicializa en 0. (Se mide en km/h)

   También debe contener los siguientes métodos:
   - `avanzar(tiempo)`: Recibe el argumento `tiempo` de tipo int expresado en segundos. Incrementa el atributo `kilometraje` de acuerdo a la siguiente fórmula $kilometraje \ += \ velocidad \ * \ tiempo$. No retorna nada.
   - `acelerar(tiempo)`: Recibe el argumento `tiempo` de tipo int expresado en segundos. Primero agrega $tiempo \ * \ 0.5$ al atributo de `aceleracion`. Luego incrementa la `velocidad` del vehículo de acuerdo al nuevo atributo `aceleración`, con la fórmula $velocidad \ += \ aceleración \ * \ tiempo \ * \ 3.6$. Después llama al método `avanzar(tiempo)` entregandole como atributo el `tiempo` recibido anteriormente y finalmente devuelve el atributo `aceleracion` a 0. No retorna nada.
   - `frenar(tiempo)`: Recibe el argumento `tiempo` de tipo int expresado en segundos. Primero resta $tiempo \ * \ 0.5$ al atributo de `aceleracion`. Luego disminuye la `velocidad` del vehículo de acuerdo al nuevo atributo `aceleración`, con la fórmula $velocidad \ += \ aceleración \ * \ tiempo \ * \ 3.6$. Si la velocidad queda negativa, se tiene que dejar en 0. Después llama al método `avanzar(tiempo)` entregandole como atributo el `tiempo` recibido anteriormente y finalmente devuelve el atributo `aceleracion` a 0. No retorna nada.

- `obtener_kilometraje`: No recibe argumentos, solo retorna el valor del atributo privado `kilometraje`.

- `reemplazar_rueda`: No recibe argumentos. Debe buscar dentro de la lista `ruedas` la rueda con menor resistencia y eliminarla de la lista. Luego debe instanciar un nuevo objeto de la clase `Rueda` y añadir este a la lista `ruedas`. No retorna nada.
  En caso de que hayan dos o más ruedas con el valor mínimo queda a su criterio cuál reemplazar.

2. Definir la clase `Moto`, que **hereda** de `Automovil`. `Moto` posee un atributo adicional, a los que ya tiene de la clase `Automovil`:

   - `cilindrada`: Se recibe como parámetro inicializador.

   Su inicializador también inicializa el atributo `ruedas` como una lista de dos objetos de clase `Rueda`. Además, debe contener los siguientes métodos:

   - `acelerar(tiempo)`: Recibe el argumento `tiempo` de tipo int expresado en segundos. Ejecuta el método `acelerar` de la clase `Automovil` y luego ejecuta el método `gastar` de cada uno de los objetos guardados en la lista `ruedas` entregandole como argumento el string “acelerar”. No retorna nada.
   - `frenar(tiempo)`: Recibe el argumento `tiempo` de tipo int expresado en segundos. Ejecuta el método `frenar` de la clase `Automovil` y luego ejecuta el método `gastar` de cada uno de los objetos guardados en la lista `ruedas` entregandole como argumento el string “frenar”. No retorna nada.

3. Definir la clase `Camion`, que **hereda** de `Automovil`. Posee un atributo adicional, a los que ya tiene de la clase `Automovil`:

   - `carga`: Indica la cantidad de carga en toneladas que puede transportar. Este atributo debe recibirse como argumento en el inicializador. (Se mide en kg)

   Su inicializador también inicializa el atributo `ruedas` como una lista de seis objetos de clase `Rueda`. Además, debe contener los siguientes métodos:

   - `acelerar(tiempo)`: Recibe el argumento `tiempo` de tipo int expresado en segundos. Ejecuta el método `acelerar` de la clase `Automovil`, y luego ejecuta el método `gastar` de cada uno de los objetos guardados en la lista `ruedas` entregandole como argumento el string “acelerar”. No retorna nada.

- `frenar(tiempo)`: Recibe el argumento `tiempo` de tipo int expresado en segundos. Ejecuta el método `frenar` de la clase `Automovil`, y luego ejecuta el método `gastar` de cada uno de los objetos guardados en la lista `ruedas` entregandole como argumento el string “frenar”. No retorna nada.

4. Finalmente debes rellenar el **código principal** según las siguientes instrucciones:

   - Instanciar vehículos

   En la parte 4.1 indicada en el archivo `main.py` crear dos objetos, uno de clase `Moto`, y uno de clase `Camion`. Los atributos iniciales con los que se completen ambos objetos son valores aleatorios y del tipo indicado en el enunciado (int, string, etc). Luego agregar ambos objetos a la lista `vehículos` existente (esta comienza vacía).

   - Completar función `accion`

   En la parte 4.2 indicada en el archivo `main.py`, rellena cada opción de la función `accion`, según lo indicado a continuación:

   - **Acelerar (opción 2):** Primero se debe permitir al usuario que escoja un tiempo para acelerar el vehículo actual y luego hay que llamar al método `acelerar(tiempo)` del vehículo. Finalmente se debe imprimir un mensaje que diga “Se ha acelerado por **X** segundos llegando a una velocidad de **Y** km/h.” Donde X es la cantidad de segundos e Y es la velocidad actual del vehículo.
   - **Frenar (opción 3):** Primero se debe permitir al usuario que escoja un tiempo para frenar el vehículo actual y luego hay que llamar al método `frenar(tiempo)` del vehículo. Finalmente se debe imprimir un mensaje que diga “Se ha frenado por **X** segundos llegando a una velocidad de **Y** km/h.” Donde X es la cantidad de segundos e Y es la velocidad actual del vehículo.
   - **Avanzar (opción 4):** Primero se debe permitir al usuario que escoja un tiempo para avanzar el vehículo actual y luego hay que llamar al método `avanzar(tiempo)` del vehículo. Finalmente se debe imprimir un mensaje que diga “Se ha avanzado **X** segundos a una velocidad de **Y** km/h.” Donde X es la cantidad de segundos e Y es la velocidad actual del vehículo.
   - **Cambiar Rueda (opción 5):** Se llama al método `reemplazar_rueda()` del vehículo actual. Finalmente se debe imprimir un mensaje que diga “Se ha reemplazado una rueda con éxito.”

- **Mostrar Estado (opción 6):** Se debe imprimir la información del vehículo actual en el siguiente orden:
    - Año
    - Velocidad
    - Kilometraje

  Además por cada `rueda` del vehículo hay que imprimir su estado actual.

Nota: Asuma que los tiempos ingresados por el usuario siempre serán mayores o igual a 0.