Python programming

1 Preword

1.1 Project description

We will be creating a train station management application. This project will have functionality to be a ”manager” of trains inside train station. The application will consist of a manager which will hold information about the trains and their carriages. Our app will allow to add new trains, remove trains from storage, do some basic calculations for statistics and export the content from manager to an external file.

Please follow Python conventions (variable naming, programming good practices etc.). Think about how the classes created might utilize some Object Oriented Programming concepts. Also think about how the created structures could be utilized somewhere else, not just for these tasks (remember that an application will always grow in size/complexity).

2 Important!

All created classes must be in separate python files!

3 Task list

- Create a class called `Carriage` which will represent a generic train carriage. The carriage will be represented by tare(dead load of empty freight railroad car), length and volume [0.5 pts];
  - Create a method called **set_volume** that will set the instance attribute *volume* to the provided input if the value is above 0 (valid value of volume). [0.1 pts];

- *EXTRA:* Modify the ***__init__*** method so that we could omit volume input and assume default value is 0 (empty) [0.1 pts]

- Create a class called `BoxCar` which will represent a box car carriage. The box car is a specific carriage type that (to limit task implementation) can hold three types of cargo - **cattle, paper, food**. [0.5 pts]
  - `BoxCar` instances should have a unique identifier with format `BC-`*number*, number should start from zero. [0.1 pts]
  - Create a method called **add_cargo** which will accept cargo type and cargo weight. Method should add a new carriage if the type follows the accepted types of cargo noted above [0.5 pts]
  - *EXTRA: Create a* ***special method*** *that will return a representation of such BoxCar object in a string format as:* ***[Box Car] ID:BC-0 Cargo type: cattle Volume: 20*** [0.3 pts]

- Create a class called `Cistern` which will represent an cistern carriage. The cistern is a specific carriage type that (to limit task implementation) can hold three types of cargo - **oil, milk, water** [0.5 pts]
  - `Cistern` instances should have a unique identifier with format `CT-`*number*, number should start from zero. [0.1 pts]
  - Create a method called `add_cargo` which will accept cargo type and cargo weight. Method should add a new carriage if the type follows the accepted types of cargo noted above [0.5 pts]
  - *EXTRA: Create a* ***special method*** *that will return a representation of such Cistern object in a string format as:* ***[Cistern] ID:CT-0 Cargo type: oil Volume: 15*** [0.3 pts]

- Create a class called `Hopper` which will represent an hopper carriage. The hopper is a specific carriage type that (to limit task implemen- tation) can hold three types of cargo - **grain, sand, fertilizer**. [0.5 pts]
  - `Hopper` instances should have a unique identifier with format `HP-`*number*, number should start from zero. [0.1 pts]
  - Create a method called `add_cargo` which will accept cargo type and cargo weight. Method should add a new carriage if the type

follows the accepted types of cargo noted above [0.5 pts]

- *<u>EXTRA:</u> Create a* ***special method*** *that will return a representation of such Hopper object in a string format as:* ***[Hopper] ID:HP-0 Cargo type: sand Volume: 20*** [0.3 pts]

- Create a class called `Gondola` which will represent an gondola carriage. The gondola is a specific carriage type that (to limit task implemen- tation) can hold three types of cargo - **coal, gravel, metal ore** [0.5 pts]
  - `Gondola` instances should have a unique identifier with format `GD-`*number*, number should start from zero. [0.1 pts]
  - Create a method called `add_cargo` which will accept cargo type and cargo weight. Method should add a new carriage if the type follows the accepted types of cargo noted above [0.5 pts]
  - *<u>EXTRA:</u> Create a* ***special method*** *that will return a representation of such Hopper object in a string format as:* ***[Gondola] ID:GD-0 Cargo type: coal Volume: 20*** [0.3 pts]

- Create a class called `TrainLinkedList` which will represent the train with its carriages. The class will be a linked list type of structure which you have to implement. [2 pts]
  - Create class called `Node` which will hold the data (carriage) as well as the pointer to next element. [0.5 pts]
  - `TrainLinkedList` instances should have a unique identifier with format `TrainLinkedList-`*number*, number should start from zero. [0.1 pts]
  - Create a method called `add_first` which will take a carriage object as input and add this carriage as first element in list [1 pts]
  - Create a method called `add_last` which will take a carriage object as input and add this carriage as last element in list [1 pts]
  - Create a method called `add_after` which will take a new carriage object and carriage object which to store after as input add this carriage after the noted carriage given in input. [1.5 pts]
  - Create a method called `add_before` which will take a new carriage object and carriage object which to store after as input add this carriage before the noted carriage given in input. [1.5

pts]

- Create a method called `remove_carriage` which will take carriage object which to remove as input and remove this object from the linked list. [1 pts]
- Create a method called `remove_carriage_by_position` which will take index/position which to remove as input and remove this object at that index from the linked list. [1 pts]
- Create a method called `remove_end` which will remove last object from the linked list. [0.5 pts]
- Create a method called `remove_first` which will remove first object from the linked list. [0.5 pts]
- Create a method called `find_carriage_by_id` which will carriage identifier as input and return this carriage object from the linked list. [1 pts]
- Create a method called `get_carriage_count` which will return carriage count in the linked list. [0.5 pts]
- *EXTRA:* Create a special method that will return a representation of TrainLinkedList instance in a string format as: `<[Box Car] ID:BC-0 Cargo type: cattle Volume: 20> -> <[Hopper] ID:HP-0 Cargo type: sand Volume: 20> -> <[Gondola] ID:GD-0 Cargo type: coal Volume: 20> -> <[Cistern] ID:CT-0 Cargo type: oil Volume: 15> -> None` [0.5 pts]

- Create a class called `TrainManager` which will represent a train manager. This class will consist of train list/array which will consist of those TrainLinkedList objects. [1 pts]
  - Create a method called `add_train` which will take a train object and add it to the managers train list. [0.5 pts]
  - Create a method called `remove_train` which will take a train object identifier and remove from the managers train list. If not found should raise ValueError with message **"Couldn't find such train in the train list!"** [0.5 pts]
  - Create a method called `avg_mean_carriages` which will calculate the average mean of carriages from the train list. [1 pts]

3.1 Driver code & sample data

Thecode you can use is in the *main.py*. It will contain basic logic flows that your solution should follow.

4 Task delivery

You can turn in the application code as zip folder containing necessary files