Term Project: Iteration 1 – Contact & Event Classes

(Spring 2024)

Project Description:

In this iteration, we will create the two basic classes needed for the project: the Contact class and the Event class.

The Contact class will store information pertaining to a contact, such as contact name, phone number, and email address. This information will be read from a dictionary and stored as attributes within the Contact object.

The Event class will store information pertaining to an event, such as event name, date, time, duration, and location. This information will be read from a dictionary and stored as attributes in the Event object.

Instructions – Contact Class:

In the file named Contact.py, you will find a Contact class. The class will have two methods already defined:

- `def __str__(self) -> str`
- `def __eq__(self, other: 'Contact') -> bool`

These methods define the format for printing a Contact object and determining equality with other Contact objects. Do not modify these methods.

In the Contact class, add code to handle the following:

1. Create a constructor that accepts a dictionary object as an argument.
2. For each key in the dictionary, store the associated value in a protected attribute. When storing these values, keep the following in mind:
    a. The dictionary will be in the format of a single contact from the contacts.json file. For example, keys will include FirstName, LastName, etc.
    b. When you store the associated values as attributes, make sure to give those attributes the correct names so that they will be compatible with the overall project. The correct names can be found in the `__str__` and `__eq__` methods. For example, the value associated with the FirstName key should be stored as an attribute called `self._first_name`
    c. Remember that each Contact object will contain information for a **single** contact, not the entire list. You will not need to loop through every contact in a file; only read the necessary information for one contact.

3. If the expected key is not in the dictionary, assign an appropriate default value to the attribute. For example, if the constructor receives a dictionary object with no FirstName key, you might set `self._first_name` equal to an empty string (“”), or `self._uid` equal to -1.
4. In addition to the values read from the dictionary, create another attribute called `self._last_contact` to store the last date of communication with the contact. Give this attribute an appropriate default value, such as an empty string (“”) or None.
5. Create a getter for each of the above attributes using the @property decorator. Additionally, create a setter for `self._last_contact` so that we can update it.

Instructions – Event Class:

In the file named Event.py, you will find an Event class. The class will have two methods already defined:

- `def __str__(self) -> str`
- `def __eq__(self, other: 'Event') -> bool`

These methods define the format for printing an Event object and determining equality with other Event objects. Do not modify these methods.

In the Event class, add code to handle the following:

1. Create a constructor that accepts a dictionary object as an argument.
2. For each key in the dictionary, store the associated value in a protected attribute. When storing these values, keep the following in mind:
    a. The dictionary will be in the format of a single event from the events.json file. For example, keys will include Name, UID, Date, etc.
    b. When you store the associated values as attributes, make sure to give those attributes the correct names so that they will be compatible with the overall project. The correct names can be found in the `__str__` and `__eq__` methods. For example, the value associated with the Name key should be stored as an attribute called `self._name`
    c. Remember that each Event object will contain information for a **single** event, not the entire list. You will not need to loop through every event in a file; only read the necessary information for one event.
3. If the expected key is not in the dictionary, assign an appropriate default value to the attribute. For example, if the constructor receives a dictionary object with no Name key, you might set `self._name` equal to an empty string (“”), or `self._uid` equal to -1.
4. Create a getter for each of the above attributes using the @property decorator.

Deliverables:

Submit your updated Contact.py and Event.py files in a single .zip file named userid_iteration_1.zip, where "userid" is your Tech username. For example, "jstrickler_iteration_1.zip." Submit the .zip file to the Iteration 1 dropbox in iLearn.