Object Oriented Programming

BCIT Outdoor Store is an "outdoor sports" retailer, with shops in various locations around Vancouver and its region. It recently merged with BCIT East Coast Sports, a retailer with shops in various locations in Eastern Canada. The new entity is now called "BCIT Sports".

Your job is to use classes and object oriented programming to create a unified inventory for all the stores.

BCIT Outdoor Store

These shops use a web API to manage their inventory.

The endpoint `https://bcitoutdoor.ca/api/store/STORE_NAME` is available. It returns the store inventory in JSON format:

```
{
  "products": {
    "skis": 10,
    "snowboard": 5,
    "tent": 10
  }
}
```

Please note that the format for API responses is the same as the one used in the other assignment for the exam. However, **you do not need to implement the API to complete this assignment**. You also **do not need the CSV files**. The API calls as well as the file operations are mocked in the tests.

BCIT East Coast Sports

These shops use CSV files to manage their inventory. The files are located in the `stores` folder.

The format of each file is:

```
item,quantity
tent,25
snowboard,10
skis,5
```

Part 1: create the base `Shop` class

This class represents a generic shop in the "BCIT Sports" franchise. It has the attribute `location`: the location of the shop (received by the constructor).

The class also has the following methods:

- `add_product`
    - receives two arguments: the name of a product and its quantity in the store
    - it adds / updates the `inventory`
- `get_inventory`
    - returns the inventory of the shop in the following format:

```
{
    "tent": 25,
    "snowboard": 10,
    "skis": 5
}
```

The way you store the inventory in the class instance is up to you, as long as `get_inventory` returns the correct data.

The class has additional behaviours: read and check the tests to make sure they all pass.

Grading: 1 mark per test in `test_shop.py` (5 total)

Part 2: create the `OutdoorShop` class

- This class must inherit from the `Shop` class.
- It must have an additional method `load`. This method makes a request to the API endpoint for the store, and updates the store inventory accordingly.
- The store inventory should only be updated if the request successfully completes.
- Otherwise, raise a `UserWarning`.

Grading: 1 mark per test in `test_outdoor_sports.py` (3 total)

Part 3: create the `EastCoastShop` class

- This class must inherit from the `Shop` class.
- It must have an additional method `load`. This method reads from the relevant inventory file, and updates the inventory accordingly.
- The store inventory should only be updated if the CSV file is present.
- Otherwise, raise a `UserWarning`.

Grading: 1 mark per test in `test_east_coast.py` (3 total)

Part 4: create the `BCITSports` class

This class represents the whole "BCIT Sports" franchise, and not a single shop. Its goal is to store a "global inventory" of products across all stores. The constructor for this class does not receive any arguments.

The class has the following methods:

- `register_store(<STORE LOCATION>)`: takes a location name as argument (for example: `register_store("burnaby")`).
    - this method allows to update the global inventory (adds / updates the inventory with information from that store).
    - this method must **first** try to consider the location as an `OutdoorShop` store
    - if the API request does not succeed, it should then consider the location as an `EastCoastShop` store.
    - after the global inventory is updated, this method must return `True`
    - and return `False` if the store could not be found (either in the API or in CSV files)
- `products()`: returns a list of all products available (a product is available if it is registered in at least one shop).
- `where(<PRODUCT_NAME>)`: returns a list of the store locations where this product is available. If the product is not available anywhere, return an empty list.
- `available(<PRODUCT_NAME>)`: returns the **total number of items** available for this product, **across all the stores**. If the product is not avalaible anywhere, return 0.

Grading: 7 marks total. The tests in `test_bcit_sports.py` provide guidance on how to implement the class. However, the grade will mostly depend on the quality of the code. Make sure you do not duplicate code and reuse existing methods / behaviours. The `BCITSports` class should be as concise and efficient as possible.

Submission

Submit all the required Python files to D2L *at the same time*.