Web Service Assignment

Basic

1) implement a RESTful inventory tracking web service. Clients can request information about the entire inventory, or a single item. Resource identifiers are assigned as follows:
    a. "/inventory" refers to the entire inventory
    b. "/inventory/Apples" refers to the item named "Apples" in the inventory

    The data accepted and generated by the web service is encoded using JSON

    You can start the web service by running the main method of the Main class. The web service

    listens for requests on port 8081. So, for example, a GET request to the URL http://localhost:8081/inventory

    Expected Output

```
[
  {
    "name": "Apples",
    "quantity": 3
  },
  {
    "name": "Oranges",
    "quantity": 7
  },
  {
    "name": "Pomegranates",
    "quantity": 55
  }
]
```

    http://localhost:8081/inventory/Apples
    Expected Output

```
{
  "name": "Apples",
  "quantity": 3
}
```

2) Handle HTTP PUT requests as follows:
    a. If resource identifier is "/inventory", replaces the entire inventory with the inventory encoded in the JSON document found in the body of the request
    b. If the resource identifier is "/inventory/itemname", replaces the item with the given item name, replacing it with the item that is encoded in the JSON document found in the body of the request

3) Handle HTTP POST requests as follows:
    a. If the resource identifier is "/inventory", then a single item is ready from the JSON document encoded in the body of the request, and the item is added to the inventory.
4) Handle DELETE requests as follows:
    a. If the resource identifier is "/inventory", all items are deleted from the inventory
    b. If the resource identifier is "/inventory/itemname", the named item is deleted.

Advance

5) A real inventory web service would almost certainly use a database to store the inventory information. However, in web applications and web services, it is generally useful to have a "persistence layer" abstraction that shields the application from knowledge of how data is being stored.
   In this assignment, the IDatabase interface describes the persistence operations:

```
/**
 * Persistence operations for fruit web service.
 */
public interface IDatabase {
    /**
     * Get a single {@link Item} for the given item name.
     *
     * @param itemName the item name
     * @return the {@link Item}, or null if there is no such item
     */
    public Item getItem(String itemName);

    /**
     * Get the current inventory (list of {@link Item}s.
     *
     * @return the current inventory (list of {@link Item}s
     */
    public List<Item> getInventory();
}
```

   Because this is an interface, it can be implemented in a variety of ways. For this assignment, it is implemented by a FakeDatabase class, which simply stores the items in the inventory in an ArrayList:

```
/**
 * Implementation of the {@link IDatabase} interface that stores
 * objects using in-memory data structures.  It doesn't
 * provide actual persistence, but is useful as a proof
 * of concept.
 */
public class FakeDatabase implements IDatabase {
    private List<Item> inventory;

    public FakeDatabase() {
        inventory = new ArrayList<Item>();
```

```
        // Populate initial inventory
        inventory.add(new Item("Apples", 3));
        inventory.add(new Item("Oranges", 7));
        inventory.add(new Item("Pomegranates", 55));
    }

    @Override
    public Item getItem(String itemName) {
        for (Item item : inventory) {
            if (item.getName().equals(itemName)) {
                // return a copy
                return new Item(item.getName(), item.getQuantity());
            }
        }

        // no such item
        return null;
    }

    @Override
    public List<Item> getInventory() {
        // return a copy
        return new ArrayList<Item>(inventory);
    }
}
```

The idea is that the single instance of IDatabase can be changed from being a FakeDatabase object to a real database at some future point without affecting any of the code that uses the persistence layer. (You won't need to do this for this assignment.)

Testing

You can use the curl program to test your implementation. (You can't use a web browser for testing because a web browser cannot generate a PUT request.)

Let's say that you have a text file called replaceApples.txt with the following JSON data:

```
{
  "name": "Apples",
  "quantity": 16
}
```

You can send this as the body of a PUT request using the following curl command:

```
curl -X PUT -d @replaceApples.txt http://localhost:8081/inventory/Apples
```

Note that you must run the command from the directory containing replaceApples.txt.

Note that using the -X POST option will send a POST request.