Worksheet: Implementing the Bag Data Structure with Pair Programming

CS 152 Data Structures

Overview

This worksheet will guide you through implementing the `Bag` data structure step-by-step, testing it, and submitting your work. Follow the instructions carefully, and ask for help from the professor or TA if needed.

Instructions

Pick a partner of your choice to pair with on this activity. At least one of you should have a computer and ideally, you have met the prerequisites for beginning Assignment 1. This includes having Assignment 1 configured with CodeGrade and your GitHub repository. Please note: although you are working as a pair, you are both responsible for submitting your work to GitHub by the official assignment due date. Submit the completed worksheet by the end of class. If you don't finish, you can keep the worksheet and complete it on your own time, and submit it at the beginning of the next class.

Pair Programming Team

Developer 1 Name: ______________________________
Developer 2 Name: ______________________________

Part 1: Understanding the Bag Data Structure

Before you start coding, think about how the `Bag` should work internally. Recall that a `Bag` is a collection where items can appear multiple times, so you need to decide how to store the items and their counts.

Questions to Consider

1. What internal data structure will you use to store the items in the `Bag`? Why?
2. How will you track the number of occurrences of each item?
3. How will you ensure the `Bag` only accepts valid items?

Write your answers below:

Part 2: Step-by-Step Implementation

Implement the methods in `bag.py` one at a time. After completing each method, run the associated test in `test_bag.py` using the **Test Explorer** in Visual Studio Code (it is on the left side of VS Code and has a beaker symbol).

2.1 Implement the `__init__` Method

Steps:

1. Open `bag.py` and navigate to the `__init__` method:
   ```
   def __init__(self) -> None:
   ```
2. Remove the `raise NotImplementedError` statement.
3. Initialize the internal data structure that you will be using to store the items and their counts.
4. Add any other necessary instance variables.
5. Since the Bag may be instantiated with items, remember to iterate over those items and manually add them to the Bag.

2.2 Implement the `add` Method

Steps:

1. Open `bag.py` and navigate to the `add` method:
   ```
   def add(self, item: T) -> None:
   ```
2. Remove the `raise NotImplementedError` statement.
3. Add the item to the `Bag` and update its count. Remember to check if the item is already in the `Bag` and increment its count. If it is not in the `Bag`, add it with a count of 1. If the item is `None`, raise a `TypeError`.

Verify: Run the following tests (see Part 3 for more guidance on running tests):

- `test_add_item_increases_count`
- `test_add_none_raises_type_error`

Questions:

- What does `item: T` mean in the type hint?
- What does `-> None` indicate about the return value?

2.3 Implement the `remove` Method

Steps:

1. Define the `remove` method:

   ```
   def remove(self, item: T) -> None:
       # Implementation here
   ```

2. This method should:

   - Decrease the count of the specified item in the `Bag`.
   - Raise a `ValueError` if the item is not in the `Bag`.

Verify: Run the following tests (see Part 3 for more guidance on running tests):

- `test_remove_item_decreases_count`
- `test_remove_nonexistent_item_raises_value_error`

Questions:

- Why does `remove` return `None`?

Remaining Methods

Repeat the process for each of the following methods:

- `count` **Test:** `test_count_returns_correct_number_of_occurrences`
- `__len__` **Test:** `test_len_returns_total_number_of_items`
- `distinct_items` **Test:** `test_distinct_items_returns_unique_items`
- `__contains__` **Test:** `test_contains_checks_item_membership`
- `clear` **Test:** `test_clear_removes_all_items`

—

Part 3: Running Tests in Visual Studio Code

1. Open the **Test Explorer** in Visual Studio Code.
2. Run individual tests after completing each method.
3. If a test fails, review your implementation, fix the issue, and re-run the test.
4. Use the error messages to help you identify and fix the problem.
5. Use breakpoints and the debugger to step through your code and understand what's happening. To debug a test, add a breakpoint in the test or in the code, then right-click the test and select **Debug Test**.
6. If you encounter an error you can't resolve, ask the professor or TA for help.

—

Part 4: Committing and Pushing Your Code to GitHub

1. Open the **Source Control** tab in Visual Studio Code.
2. Add a meaningful message to the commit message box; i.e.:

```
"Completed Bag implementation"
```

3. Commit and Push your changes to GitHub.

—

Part 5: Viewing Your Score in CodeGrade

1. Visit the **Bag assignment** in Canvas.
2. Click the [**Load in New Window/Tab**] button at the bottom of the assignment.
3. Review your CodeGrade results and score.

Troubleshooting: If you don't see your score or encounter issues, raise your hand in class or visit the professor or TA during office hours.

—

Need Help?

- Ask questions in class.
- Visit the professor or TA during office hours.
- Use the resources available in Canvas.