Data Structures In-class Practices

The Data Structure “Bag”

Ling-Chieh Kung

Department of Information Management
National Taiwan University

Problem 1: the interface

- Consider the **`BagInterface`**, **`ArrayBag`**, and **`LinkedBag`** classes introduced in class.
  - If it cannot be used to create objects, what is the point of having an abstract class **`BagInterface`**?
  - Should be private functions in **`ArrayBag`**, if any, be declared in **`BagInterface`**? Why or why not?
  - Why **`LinkedBag`** should have a virtual destructor?
  - Do we really want **`ArrayBag`** and **`LinkedBag`** at the same time?

- **Note.** In Java, abstract classes are called interfaces directly.

Problem 2: `removeAll()`

- Consider the class **`ArrayBag`** (and **`BagInterface`**, of course).
- We want to implement a new instance function **`removeAll()`**, which takes an **`ItemType`** item **`anEntry`** as a parameter. It should remove all copies of **`anEntry`** in the bag.
  - E.g., if the bag contains {0, 1, 2, 2, 3, 4, 4, 9, 4, 2, 9}, and **`anEntry`** is 4, the result should be {0, 1, 2, 2, 3, 9, 2, 9}.
- Please add a line of function declaration in **`BagInterface`**.
  - Write appropriate comments, maybe including pre-condition and post-condition, to let the clients know its behavior.
- Please implement the function in **`ArrayBag`**.
  - Is **`remove()`** useful?

Problem 3: overloading `contains()`

- Consider the class **`ArrayBag`** (and **`BagInterface`**, of course).
- We want to overload the function **`contains()`**, which now takes an **`ItemType`** array **`entries`** and its length **`len`** as parameters. It should return true if all items in **`entries`** exists in the bag with at least one occurrence or false otherwise.
  - E.g., if the bag contains {0, 1, 2, 2, 3, 4, 4, 9, 4, 2, 9}, and **`entries`** contains {2, 3}, it should return true. If **`entries`** contains {2, 5}, it should return false.
- Please add a line of function declaration in **`BagInterface`**.
  - Write appropriate comments, maybe including pre-condition and post-condition, to let the clients know its behavior.
- Please implement the function in **`ArrayBag`**.

Problem 4: `removeAll()` again

- Consider the class **`LinkedBag`** (and **`BagInterface`**, of course).
- We want to implement a new instance function **`removeAll()`** for **`LinkedBag`**.
- Please implement the function in **`LinkedBag`**.
- DO NOT use **`remove()`** and **`getPointerTo()`** (why?).

Problem 5: `getTypeCount()`

- Consider the class **`LinkedBag`** (and **`BagInterface`**, of course).
- We want to implement a new instance function **`getTypeCount()`**, which takes no parameter and returns the number of distinct item types contained in the bag.
  - E.g., if the bag contains {0, 1, 2, 2, 3, 4, 4, 9, 4, 2, 9}, it should return 6.
  - E.g., if the bag contains nothing, it should return 0.
  - Note that the items in your bags are typically unsorted.
- Please add a line of function declaration in **`BagInterface`**.
  - Write appropriate comments, maybe including pre-condition and post-condition, to let the clients know its behavior.
- Please implement the function in **`LinkedBag`**.

Problem 6: `DistinctBag`

- Let's implement a class **`DistinctBag`**.
  - A distinct bag contains at most one copy for each item.
  - When one tries to add another copy of an existing item, it throws an exception.
- We may implement **`DistinctBag`** by inheriting from **`ArrayBag`** (or **`LinkedBag`**, of course).
  - Which function(s) may be used directly?
  - Which function(s) should be overridden? How?
- What if **`ArrayBag`** (or **`LinkedBag`**) has **`removeAll()`**? Is there a way to prevent clients from invoking **`removeAll()`**?