Bag ADT

Learning Objectives

- Define an ADT for a data structure
- Implement an ADT in multiple ways
- Use connected Nodes to help store data in structured way
- Evaluate the benefits of different implementations of an interface

Data Structures

- An object type used to store, retrieve, and represent data
  - **String**:
    - a data structure consisting of an ordered sequence of **chars**
    - Includes many methods (substring, charAt, length, etc.)
  - **Arrays:**
    - Ordered, fixed-length sequence of data of a single type
    - Methods: none that we've used!

A Data Structure of Our Own

- Each type of data structure has its own uses and limitations
    - We use Strings and arrays in different contexts
- We can define a data structure based on what we want it to do
    - Selecting its behaviors → choosing its methods
    - Defining a type based only on its behaviors is **exactly** the use case for an **Abstract Data Type**

The Bag Data Structure

- An **unordered** data structure that can contain only objects of a single type.
- Models how a physical **bag** can store objects
    - What behaviors does a bag have from the perspective of a storage device?

Modeling with Bag

- `Bag` objects are used to **store several elements of the same type**
- A `Bag` has a **variable size**: it grows or shrinks automatically to fit its contents
- Unlike our ordered data structures, elements in a bag do not have a **position!**
    - A bag contains a value or it does not: **membership**, not **order.**

Book Bag Behavior

Behavior	Description
Count elements	How many elements are held in this bag?
Remove a particular element	Look for a particular element in the bag and remove it.
Remove a random element	Reach into the bag and pull out an element at random.
Check membership	Is a given element found among the elements contained in the bag?
Check emptiness	Does the bag have any elements at all?
Describe	Generate a text/String description of the contents of this bag

Book Bag Behavior

Behavior	Description	Method signature
Count elements	How many elements are held in this bag?	int size()
Remove a particular element	Look for a particular element in the bag and remove it.	Book remove(Book target)
Remove a random element	Reach into the bag and pull out an element at random.	Book removeRandom()
Check membership	Is a given element found among the elements contained in the bag?	boolean contains(Book target)
Check emptiness	Does the bag have any elements at all?	boolean isEmpty()
Describe	Generate a text/String description of the contents of this bag	String toString()

Headers in the Bag ADT

- The ADT provided has some new terms in the method headers:
    - You don’t have to follow this directly in the assignments, but you might find the keywords helpful when writing these headers
- **@param:** an input parameter
- **@return:** a description of what value to return
- **@precondition:** something that is assumed to be true of the object/input before calling the method
- **@postcondition:** something that is assumed to be true of the object/output after calling the method

Bag Methods: `size()`

- The `size` method returns the number of elements inside a **Bag**
- An empty **Bag** has a size of 0

Example:

```
Bag bookBag = new ArrayBag(); or
Bag bookBag = new NodeBag();

System.out.println(bookBag.size());➔ 0
```

Bag Methods: `add(element)`

- The `add(element)` method **adds** the parameter element to the bag
- Adding an element to the bag will increase the size of the bag by 1

Bag Methods: `add(element)`

The elements must be of type **Book**

Example:

```
Bag bookBag = new ArrayBag();

System.out.println(bookBag.size());➔ 0
```

Bag Methods: `add(element)`

The elements must be of type **Book**

Example:

```
Bag bookBag = new ArrayBag();
System.out.println(bookBag.size());➔ 0
bookBag.add(gilead);
System.out.println(bookBag.size());➔ 1
```

Bag Methods: `add(element)`

The elements must be of type **Book**

Example:

```
Bag bookBag = new ArrayBag();
System.out.println(bookBag.size());➔ 0
bookBag.add(gilead);
System.out.println(bookBag.size());➔ 1
```

Bag Methods: `add(element)`

Example:

```
Book bookBag = new Bag ();
bookBag.add(gilead);
```

Bag Methods: `add(element)`

Example:

```
Book bookBag = new Bag ();
bookBag.add(gilead);

bookBag.add(actualAir);
```

Bag Methods: `add(element)`

Example:

```
Book bookBag = new Bag ();
bookBag.add(gilead);

bookBag.add(actualAir);

bookBag.add(outline);
```

Bag Methods: `remove(target)`

- The `remove(target)` method removes one element from the bag that's equal to the target
- The method will need to move all the elements in the implementation after the removal to fill up the gap.
- The method decreases the size of the bag by 1

List Methods: remove(index)

Example:

```
Book bookBag = new Bag ();
bookBag.add(gilead);
bookBag.add(actualAir);
bookBag.add(outline);
```


```
studentsList.remove(actualAir);
```