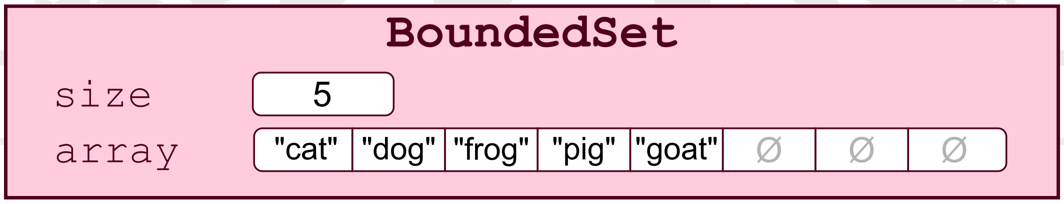
Lab 3: Bounded Set

You will create a `BoundedSet` object (representing a mathematical set), that holds a limited number of objects of a generic type. No duplicate elements are allowed within the object. The generic type will be called `E`.

The easiest way to implement this object is to have two fields:

1. An `int` holding the **current number of valid elements** contained in the `BoundedSet`.
2. An `E` array, which holds the **elements themelves**. (Recall that in Java, you cannot directly create an `E` array, but must use an `Object` array instead.) This will initially be filled with `null` objects, but will gradually fill with real values.



Your `BoundedSet` must implement the following public methods:

- `BoundedSet(int capacity)`, to initialize the object. The argument `capacity` indicates the maximum number of elements this object can hold.
- `boolean add(E element)`, which adds a new element to the `BoundedSet`. It will fail if the element is `null`, if it is already inside the `BoundedSet`, or if the `BoundedSet` is already full. It will return `true` if it succeeded, or `false` if it failed.
- `boolean contains(E element)`, which returns whether or not the element is to be found inside the `BoundedSet`.
- `int getCapacity()`, which returns the maximum number of elements which could be contained in the `BoundedSet`.
- `int getSize()`, which returns the number of elements currently contained in the `BoundedSet`.
- `boolean remove(E element)`, which removes the specified element from the `BoundedSet`. It will fail and return `false` if the element is not found (including if the it is `null`). It will return `true` if it succeeded.
- `String toString()`, which returns a `String` representing this `BoundedSet`. The `String` will consist of a comma-separated list of individual elements, all inside a set of square brackets (`[]`s). Null elements at the end of the array should not be printed out.

In addition, you will probably find it helpful to create a private `findIndex()` method. This method will take an `E` element as its argument, and return the index in which it is located inside of the internal array. Writing this function will make your `add()`, `remove()`, and `contains()` methods much simpler.

Finally, you must write a `main()` method which tests your `BoundedSet`. It will make two `BoundedSet`s: one containing `Integer`s, and one containing `String`s. All of the public methods should be tested for each one.

Note that the objects contained in the `BoundedSet` are not considered to have any order. You may reorder them however you choose. (This can make some methods work more quickly. For example, your `remove()` method doesn't have to preserve the elements in order. So it may perform the removal by copying the last element over the one to be removed, then setting the last element to `null` and decrementing the size.