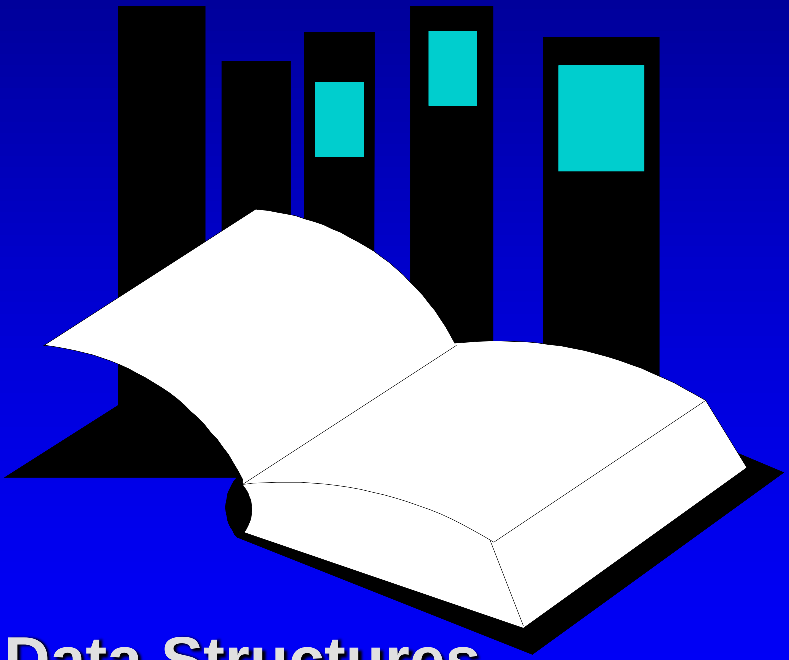
Collection Classes

- A **Collection class** is a data type that is capable of holding a group of items.
- In Java, Collection classes can be implemented as a class, along with methods to add, remove, and examine items.



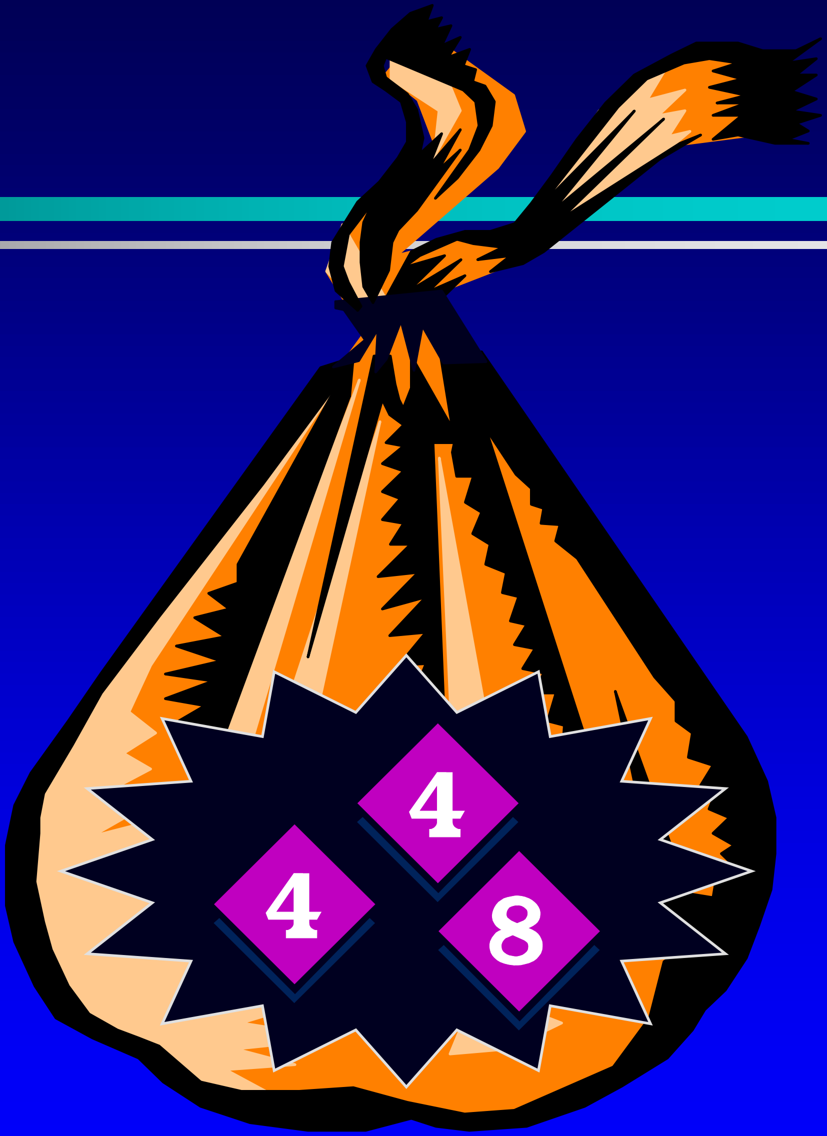
Data Structures and Other Objects Using Java

Bags

- For the first example, think about a bag.

Bags

- For the first example, think about a bag.
- Inside the bag are some numbers.

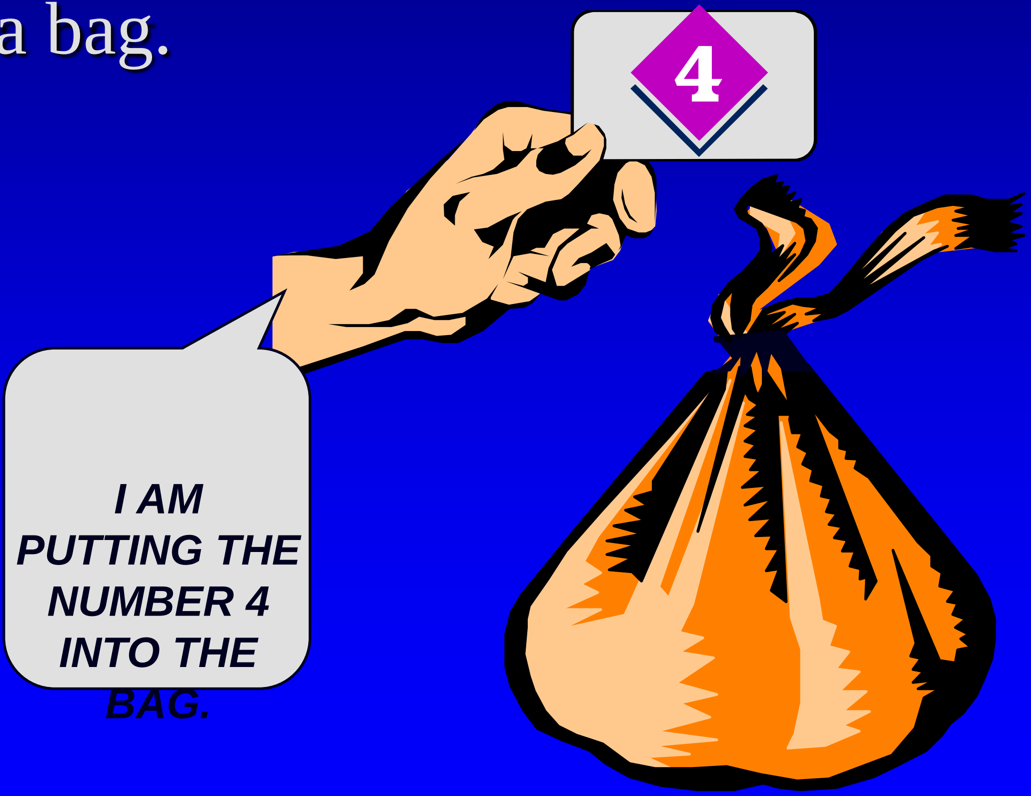


Initial State of a Bag

- When you first begin to use a bag, the bag will be empty.
- We count on this to be the **initial state** of any bag that we use.

Adding Numbers into a Bag

- Numbers may be added into a bag.



Adding Numbers into a Bag

- Numbers may be added into a bag.

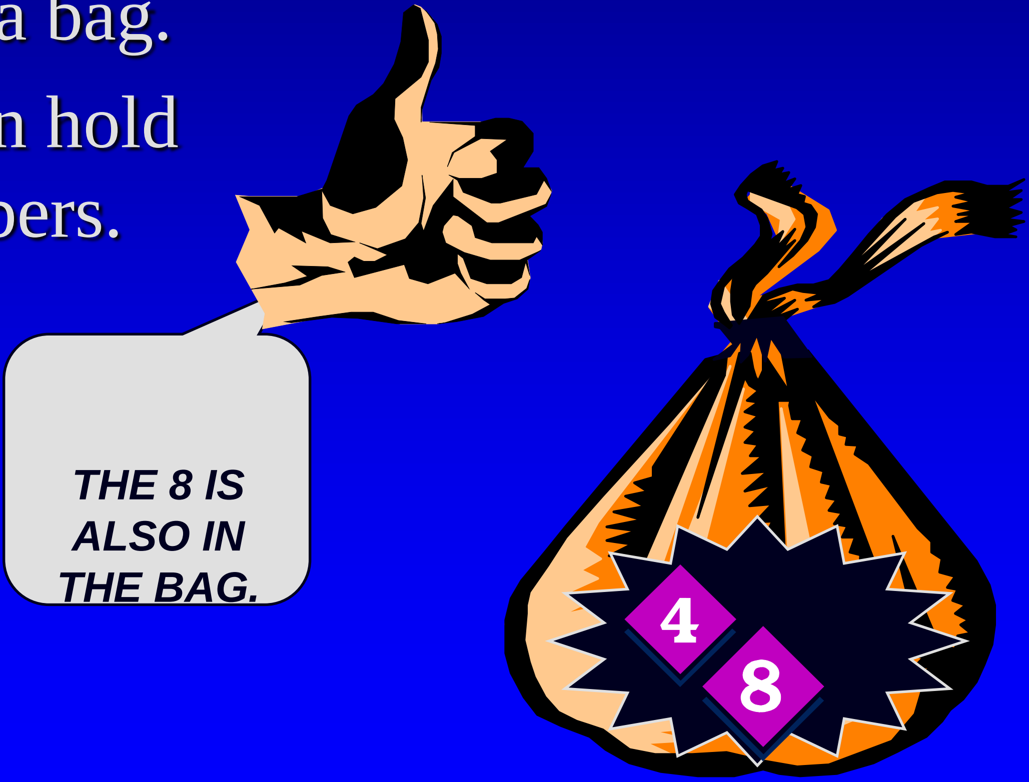
THE 4 IS IN THE BAG.
4

Adding Numbers into a Bag

- Numbers may be added into a bag.
- The bag can hold many numbers.

added Numbers into a Bag

- Numbers may be added into a bag.
- The bag can hold many numbers.



added Numbers into a Bag

- Numbers may be added into a bag.
- The bag can hold many numbers.
- We can even place the same number more than once.

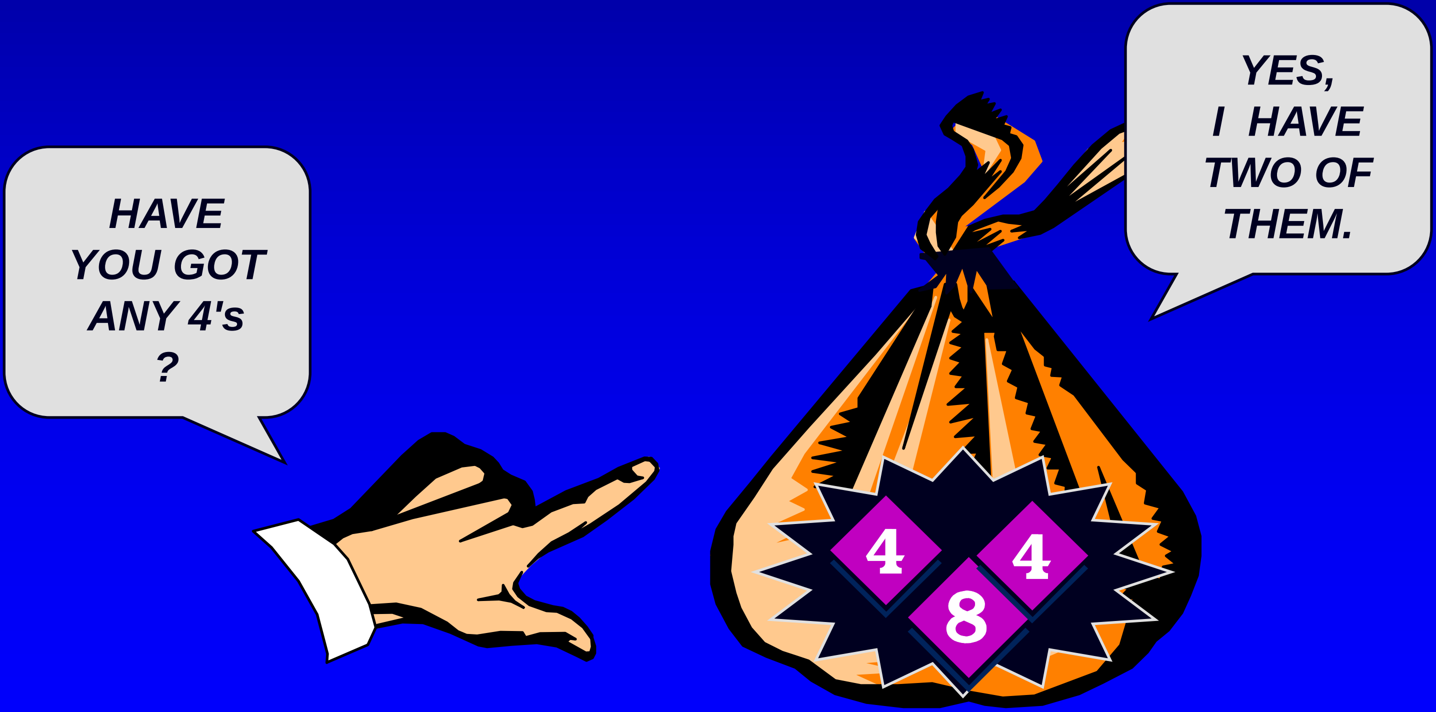
Adding Numbers into a Bag

- Numbers may be added into a bag.
- The bag can hold many numbers.
- We can even place the same number more than once.

NOW THE BAG HAS TWO 4'S AND AN 8...
4
4
8

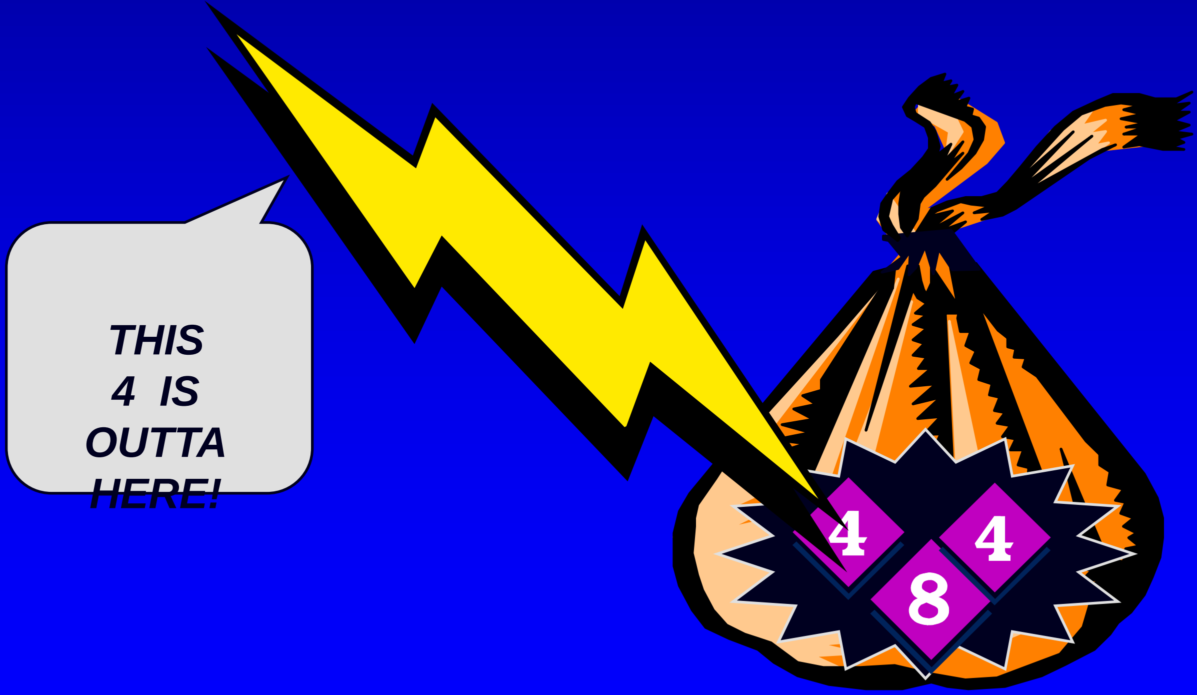
Examining a Bag

- We may ask about the contents of the bag.



Removing a Number from a Bag

- We may remove a number from a bag.



Removing a Number from a Bag

- We may remove a number from a bag.
- But we remove only one number at a time.

ONE 4 IS GONE, BUT THE OTHER 4 REMAINS.
4
8

How Many Numbers

- Another operation is to determine how many numbers are in a bag.

Summary of the Bag Operations

- A bag can be put in its **initial state**, which is an empty bag.
- Numbers can be **added** into the bag.
- You may check how many **occurrences** of a certain number are in the bag.
- Numbers can be **removed** from the bag.
- You can check **how many** numbers are in the bag.

The Bag Class

- Java classes (introduced in Chapter 2) can be used to implement a Collection class such as a Bag.
- The class definition includes:
  - **The heading of the definition**

```
public class Bag
```

The Bag Class

- Java classes (introduced in Chapter 2) can be used to implement a Collection class such as a Bag.
- The class definition includes:
    - **The heading of the definition**
    - **A constructor**

```
class Bag
{
   public Bag(  )...
```

The Bag Class

- Java classes (introduced in Chapter 2) can be used to implement a Collection class such as a Bag.
- The class definition includes:
  - **The heading of the definition**
  - **A constructor**
  - **Public methods**

```
public class Bag
{
    public Bag(  )...
    public void add(...
    public void remove(...
    ...and so on
```

The Bag Class

- Java classes (introduced in Chapter 2) can be used to implement a Collection class such as a Bag.
- The class definition includes:
  - **The heading of the definition**
  - **A constructor**
  - **Public methods**
  - **Private instance variables**

```
public class Bag
{
    public Bag(  )...
    public void add(...
    public void remove(...
    ...and so on
```

We'll look at private instance variables later.

The Bag’s Constructor

- Places a bag in the initial state (an empty bag)

```
//  Postcondition:  The Bag has been initialized
//  and it is now empty.
public Bag( )
{
 . . .

}
```

The Add Method

- Adds a new number to the bag

```
public void add(int newEntry)
{

   . . .
}
```

The Size Method

- Counts how many integers are in the bag.

```
//   Postcondition:  The return value is the number
//   of integers in the Bag.
public int size(  )
{

    . . .
}
```

The countOccurrences Method

- Counts how many copies of a number occur

```
//   Postcondition:  The return value is the number
//   of copies of target in the Bag.
public int countOccurrences(int target)
{


   . . .
}
```

The Remove Method

- Removes one copy of a number

```
//   Postcondition:  If target was in the Bag, then
//   one copy of target has been removed from the
//   Bag, and the return value is true; otherwise the
//   Bag is unchanged and the return value is false.
public boolean remove(int target)
{

   . . .
}
```

Using the Bag in a Program

- Here is typical code from a program that uses the new Bag class:

```
Bag  ages = new Bag( );

// Record the ages of three children:
ages.add(4);
ages.add(8);
ages.add(4);
```

Documentation for the Bag Class

- The documentation gives **specifications** for the bag methods.
- Specifications are written as **precondition/postcondition** contracts.
- Everything needed to **use** the Bag class is included in this documentation.

Bag’s documentation can be automatically created with the Javadoc tool described in Appendix H.

A Quiz

Suppose that a Mysterious Benefactor provides you with the Bag class, but you are only permitted to read the documentation. You cannot read the class implementation or .java file. Can you write a program that uses the Bag data type ?

- Yes I can.
- No. Not unless I see the class implementation for the Bag.

A Quiz

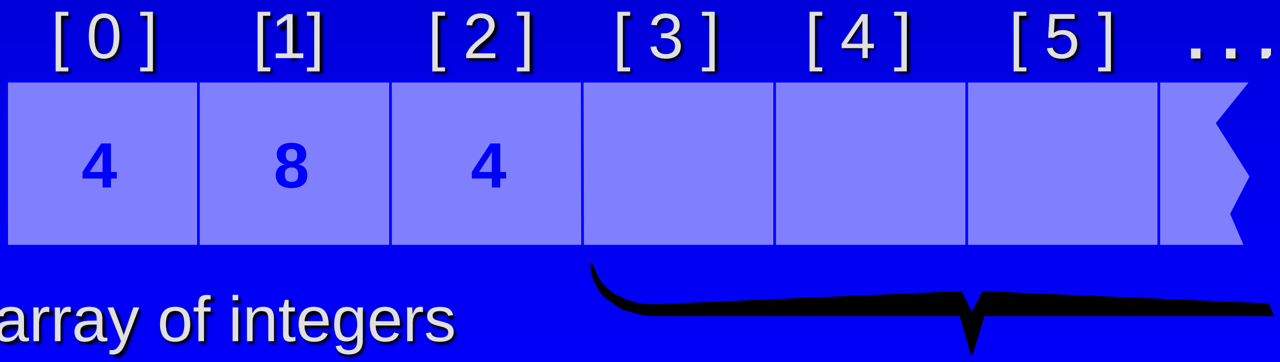
Suppose that a Mysterious Benefactor provides you with the Bag class, but you are only permitted to read the documentation. You cannot read the class implementation or .java file. Can you write a program that uses the Bag data type ?

- Yes I can.

  You know the name of the new data type, and you also know the headings and specifications of each of the operations. This is enough for you to create and use Bags.

Implementation Details

- The entries of a bag will be stored in the front part of an array, as shown in this example.

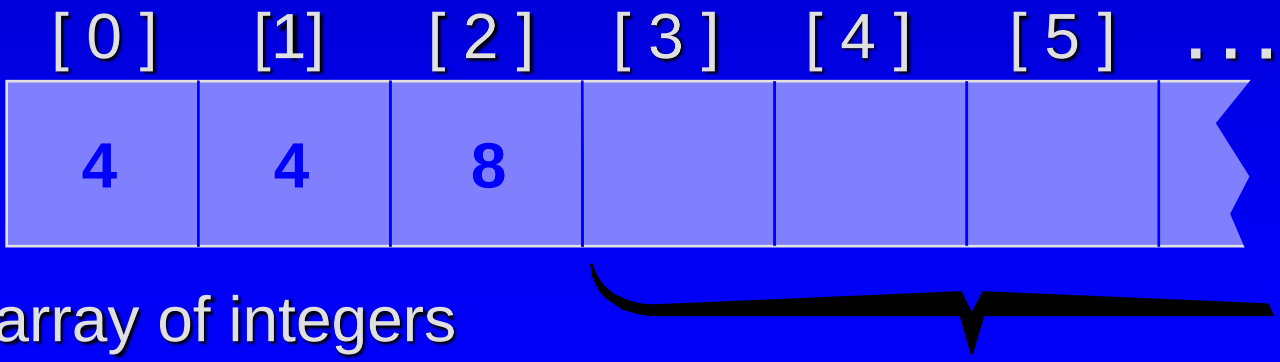


An array of integers

We don't care what's in this part of the array.

Implementation Details

- The entries may appear in any order. This represents the same bag as the previous one. . .

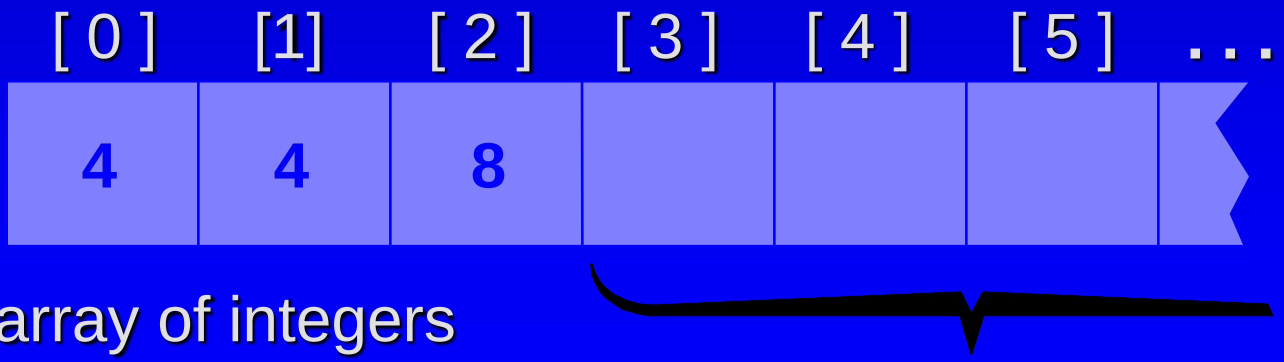


An array of integers

We don't care what's in this part of the array.

Implementation Details

- . . . and this also represents the same bag.

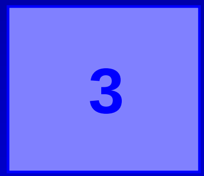


An array of integers

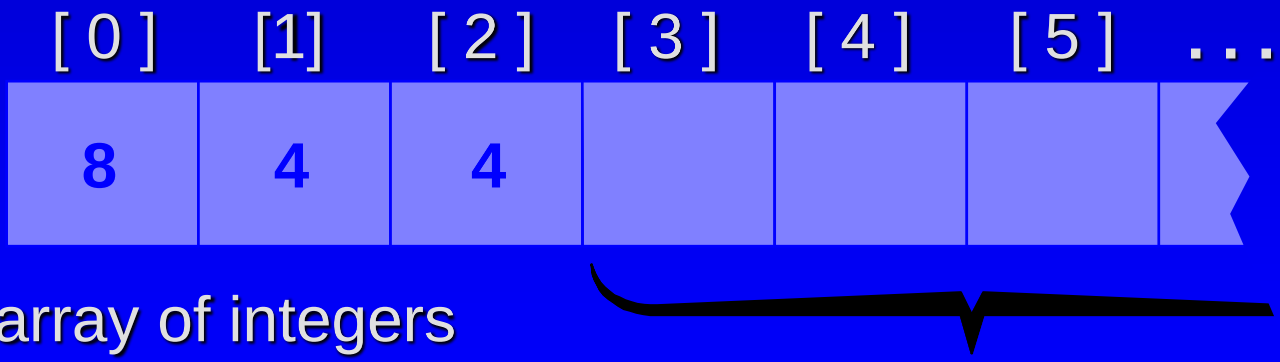
We don't care what's in this part of the array.

Implementation Details

- We also need to keep track of how many numbers are in the bag.



An integer to keep track of the bag's size

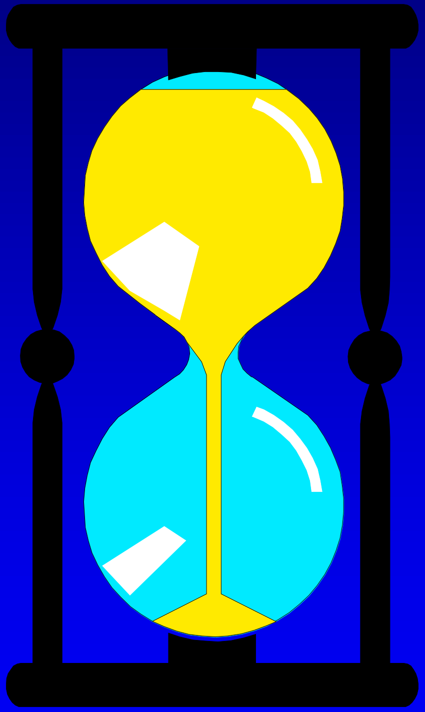


An array of integers

We don't care what's in this part of the array.

An Exercise

Use these ideas to write a list of private instance variables could implement the Bag class. You should have two instance variables.

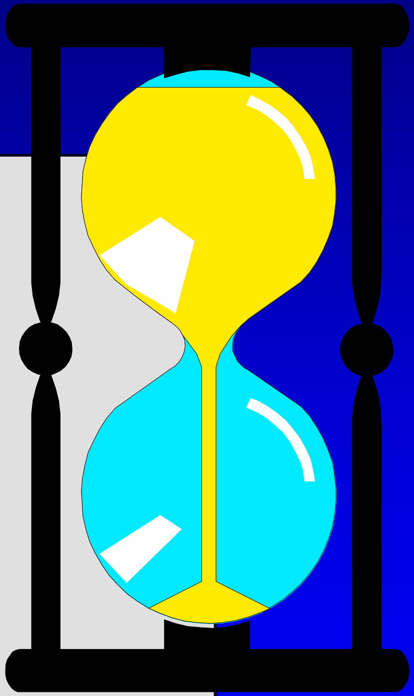


You have 60 seconds to write the declaration.

An Exercise

One solution:

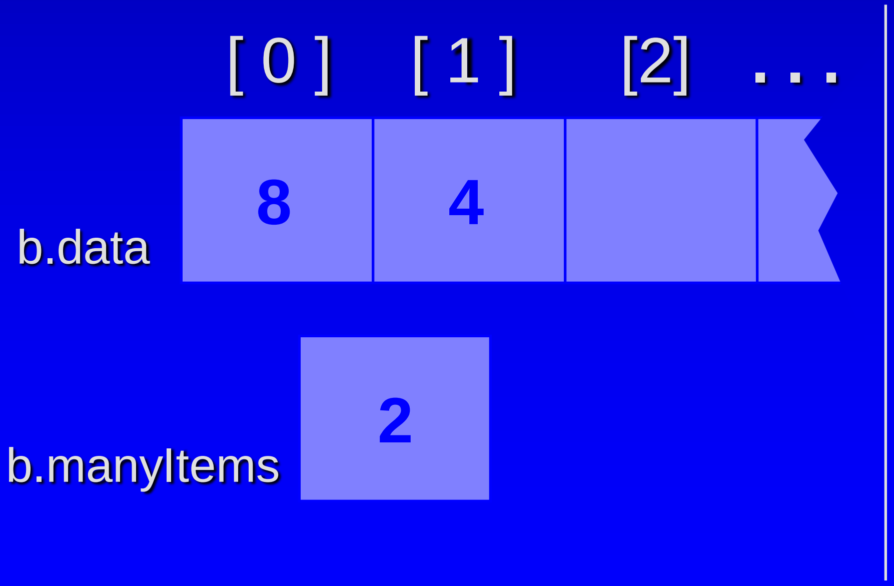
```
public class Bag
{
    private int[ ] data;
    private int manyItems;
    ...
}
```



An Example of Calling Add

```
public void add(int newEntry)
```

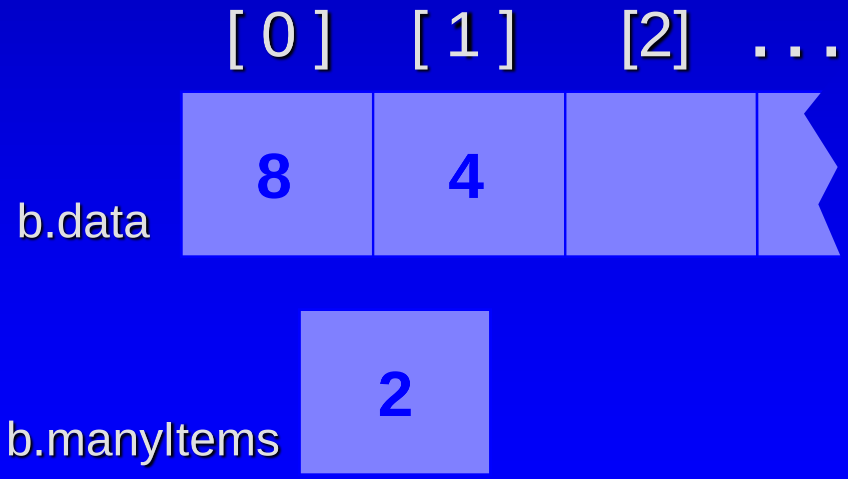
Before calling add, we might have this bag b:



An Example of Calling Add

```
public void add(int newEntry)
```

We activate
b.add(17)

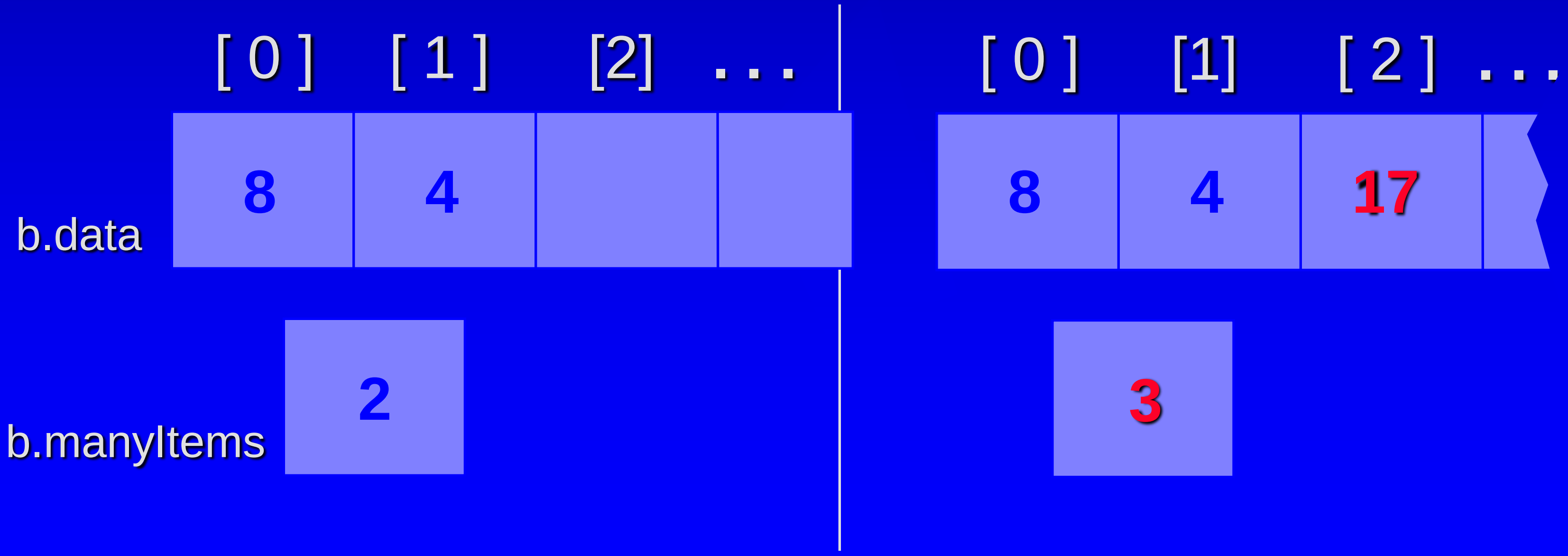


What values will be in b.data and b.manyItems after the method finishes ?

An Example of Calling Add

```
public void add(int newEntry)
```

After activating b.add(17), we will have this bag b:



Pseudocode for add

Œ Make sure there is room for a new entry in the array.

? Place newEntry in the appropriate location of the data array.

Ž Add one to the instance variable manyItems.

What is the "appropriate location" of the data array ?

Pseudocode for add

Œ Make sure there is room for a new entry in the array.

? Place newEntry in the appropriate location of the data array.

Ž Add one to the instance variable manyItems.

```
data[manyItems]  = newEntry;
manyItems++;
```

Pseudocode for add

Œ Make sure there is room for a new entry in the array.

? Place newEntry in the appropriate location of the data array.

Ž Add one to the instance variable manyItems.

```
data[ manyItems++]  = newEntry;
```

The Other Bag Operations

- Read Section 3.2 for the implementations of the other bag methods.
- Remember: If you are just **using** the Bag class, then you don’t need to know how the operations are implemented.
- Later we will **reimplement** the bag using more efficient algorithms.
- We’ll also have a few other operations to manipulate bags.

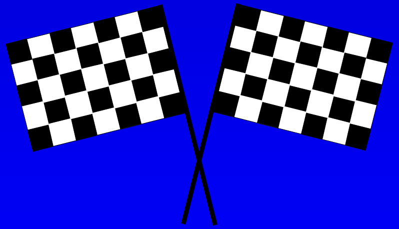
Other Kinds of Bags

- In this example, we have implemented a bag containing **integers**.
- But we could have had a bag of **float numbers**, a bag of **characters**, a bag of **Strings** . . .

Suppose you wanted one of these other bags. How much would you need to change in the implementation ?

Summary

- A Collection class is a class that can hold a group of items.
- Collection classes can be implemented with a Java class.
- The author of the class should provide documentation that another programmer can read to use the class.
- Other details are given in Section 3.2, which you should read.



THE END