7.3. Programming with arrays and Classes
Wrapper Classes for Arrays

- Arrays can be made into objects by creating a wrapper class
  - » similar to wrapper classes for primitive types
- In the wrapper class:
  - » make an array an instance variable
  - » define constructors
  - » define accessor methods to read and write element values and parameters
- an example of creating a wrapper class for an array of objects of type `OneWayNoRepeatsList`
  - » the wrapper class defines two constructors plus the following methods:

    `addItem`, `full`, `empty`, `entryAt`, `atLastEntry`, `onList`, `maximumNumberOfEntries`, `numberOfEntries`, and `eraseList`

Listing 7.8 Using the Class OneWayNoRepeatsList - ListDemo.java

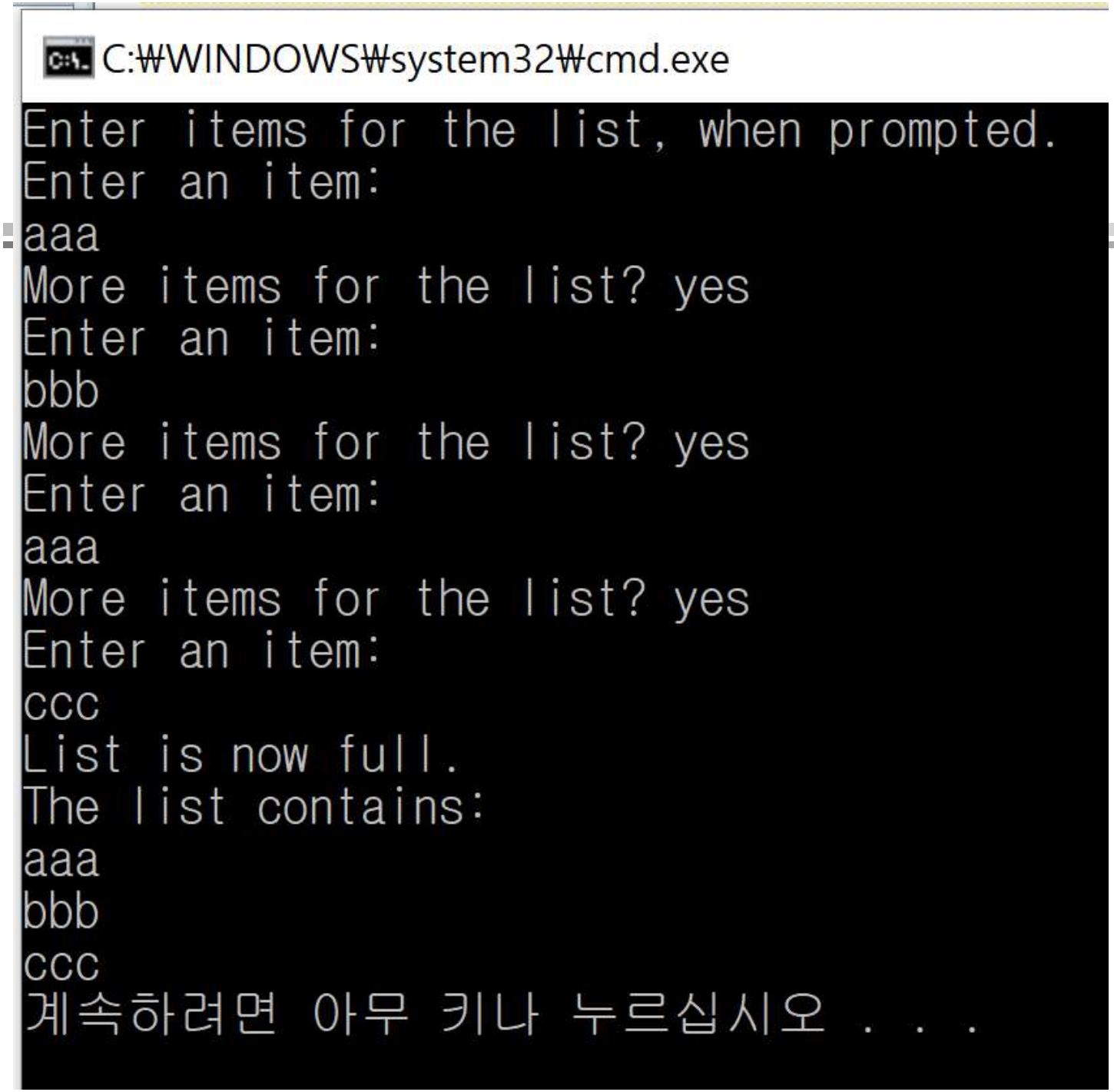
```
import java.util.Scanner;

public class ListDemo
{
        public static final int MAX_SIZE = 3; //Assumed > 0

   public static void main(String[] args)
   {
      OneWayNoRepeatsList toDoList =
                  new OneWayNoRepeatsList(MAX_SIZE);

      System.out.println("Enter items for the list, when prompted.");
      boolean moreEntries = true;
      String next = null;
      Scanner keyboard = new Scanner(System.in);
```

```
while (moreEntries && !toDoList.isFull())
      {
         System.out.println("Enter an item:");
         next = keyboard.nextLine( );
         toDoList.addItem(next);
         if (toDoList.isFull( ))
         {
            System.out.println("List is now full.");
         }
         else
         {
            System.out.print("More items for the list? ");
            String ans = keyboard.nextLine( );
            if (ans.trim().equalsIgnoreCase("no"))
               moreEntries = false; //User says no more
         }
      }
      System.out.println("The list contains:");
      int position = toDoList.START_POSITION;
      next = toDoList.getEntryAt(position);
      while (next != null) //null indicates end of list
      {
         System.out.println(next);
         position++;
         next = toDoList.getEntryAt(position);
      }
   }
}
```


C:₩WINDOWS₩system32₩cmd.exe
Enter items for the list, when prompted.
Enter an item:
aaa
More items for the list? yes
Enter an item:
bbb
More items for the list? yes
Enter an item:
aaa
More items for the list? yes
Enter an item:
ccc
List is now full.
The list contains:
aaa
bbb
ccc
계속하려면 아무 키나 누르십시오 . . .

Listing 7.9 . An Array Wrapped in a Class - OneWayNoRepeatsList.java

```
 // Listing 7.9 . An Array Wrapped in a Class
public class OneWayNoRepeatsList
{
    public static int START_POSITION = 1;
    public static int DEFAULT_SIZE = 50;

    private int countOfEntries; //can be less than entry.length.
    private String[] entry;   // make an array an instance variable

    // constructor
    public OneWayNoRepeatsList(int maximumNumberOfEntries)
    {
        entry = new String[maximumNumberOfEntries];
        countOfEntries = 0;
    }
   // constructor
    public OneWayNoRepeatsList( )
    {
        entry = new String[DEFAULT_SIZE];
        countOfEntries = 0;
    }
```

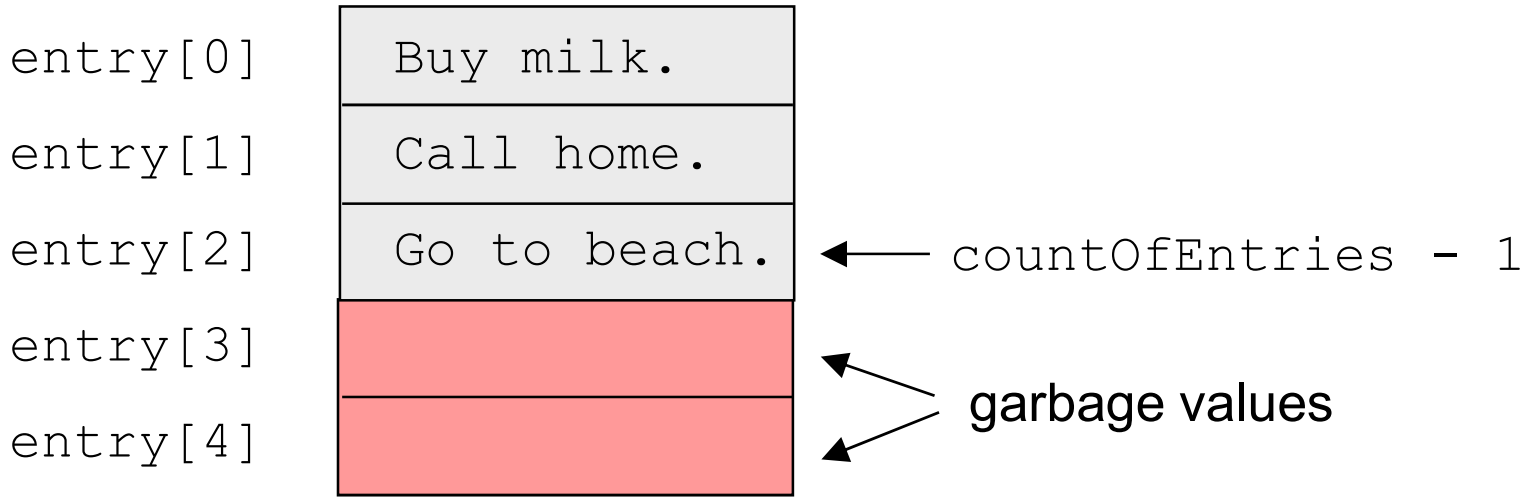
```
public boolean full( )
   {
      return (countOfEntries == entry.length);
   }
   public boolean empty( )
  {   //.....     }
  public void addItem(String item)
  {   //.....     }
  public String getEntryAt(int position)
  {
      if ((1 <= position) && (position <= countOfEntries))
         return entry[position - 1];
      else
         return null;
   }
  public boolean atLastEntry(int position)
  {   //.....     }
   public boolean onList(String item)
  {   //.....     }

  public int maximumNumberOfEntries( )
  {   //.....     }
  public int getNumberOfEntries( )
  {   //.....     }
   public void eraseList( )
  {   //.....     }
}
```

Partially Filled Arrays

- Sometimes only part of an array has been filled with data
- Array elements always contain something, whether you have written to them or not
  - » elements which have not been written to contain unknown (*garbage*) data so you should avoid reading them
- There is no automatic mechanism to detect how many elements have been filled - *you*, the programmer need to keep track!
- An example: the instance variable `countOfEntries` (in the class `OneWayNoRepeatsList`) is incremented every time `addItem` is called (see the text)

Figure 7.4 a Partially Filled Array



`countOfEntries` has a value of `3`.

`entry.length` has a value of `5`.

Searching an Array

- There are many techniques for searching an array for a particular value

- ***Sequential search***:
  - » start at the beginning of the array and proceed in sequence until either the value is found or the end of the array is reached*
    - – if the array is only partially filled, the search stops when the last meaningful value has been checked
  - » it is not the most efficient way
  - » but it works and is easy to program

* Or, just as easy, start at the end and work backwards toward the beginning

Example: Sequential Search of an Array

The `onList` method of `OneWayNoRepeatsList` sequentially searches the array `entry` to see it the parameter `item` is in the array

```
public boolean onList(String item)
{
    boolean found = false;
    int i = 0;
    while ((! found) && (i < countOfEntries))
    {
       if (item.equals(entry[i]))
          found = true;
       else
          i++;
    }

   return found;
}
```

Gotcha: Returning an Array Instance Variable

- Access methods that return references to array instance variables cause **problems for information hiding.**

  Example:

```
public String[] getEntryArray()
{
        return entry;
}
```

  Even though `entries` is declared private, a method outside the class can get full access to it by using `getEntryArray`.

- In most cases this type of method is not necessary anyhow.
- If it is necessary, make the method return a copy of the array instead of returning a reference to the actual array.

```
public String[] getEntryArray()
{
      return entry;
}
```

```
oneWayNoRepeatsList myList = new OneWayNoRepeatsList();
myList.entry[2] = "Party tonight";

String[] a = myList.getEntryArray();

a[2] = "Party tomorrow";
```

```
public String[] getEntryArray()
{
      String[] temp = new String
      int I;
      for (I=0; I<countOfEntries;I++)
            
      return temp;
}
```