Lecture 28

- Covers
  - Partially filled arrays
- Reading: Savitch Chapter 6

Partially filled arrays

- Sometimes we do not know ahead of time how many elements we wish to store in an array
- For example, how many numbers we need to average, how many employees to calculate pay for, or how many names to put in an address book

Partially filled arrays

- In these cases, we may choose to create an array large enough to hold the maximum that we could possibly need
- Then, we need to keep track of how many values we have placed into the array as .length holds the maximum we could store, not the actual number stored

Partially filled arrays

- There are two ways to keep track of this
  - An integer counter that is incremented each time we add a value to the array
  - A sentinel value that indicates the last element in the array

Partially filled arrays

- A counter

```
char[ ] text = new char[8];
int currentTextSize = 0;
text[currentTextSize++] = 'h';
text[currentTextSize++] = 'e';
text[currentTextSize++] = 'l';
text[currentTextSize++] = 'l';
text[currentTextSize++] = 'o';
for (int i = 0; i < currentTextSize; ++i)
    System.out.print(text[i]);
```

currentTextSize keeps track of how many characters have been put in the character array

0	1	2	3	4	5	6	7
‘h’	‘e’	‘l’	‘l’	‘o’			

Partially filled arrays

- A sentinel value

```
char[ ] text = new char[8];
text[0] = 'h'; text[1] = 'e';
text[2] = 'l'; text[3] = 'l';
text[4] = 'o'; text[5] = '\0';
for (int i = 0; text[i] != '\0'; ++i)
    System.out.print(text[i]);
```

The sentinel character '\0' is placed to indicate the end of the string. C and C++ both implement character strings in this fashion

0	1	2	3	4	5	6	7
‘h’	‘e’	‘l’	‘l’	‘o’	‘\0’		

Example

- Write a wrapper class to manage insertion into a partially filled String array
- The class MyStringVector must allow insertion at the end or beginning of a partially filled array
- It should change and retrieve elements, and should have a method to display the vector
- When the array attribute is created by the constructor it should be of size 16
- When the number of strings added to MyStringVector becomes greater than 16 it should create a larger array

Defining the class header

```
public class MyStringVector
{

}
```

Defining the attributes

```
public class MyStringVector
{
   String[ ] contents;
   int currentSize;

}
```

reference to an array of Strings

variable to keep track of the number of elements put into the array

Defining the constructor

```
MyStringVector( )
{
    contents = new String[16];
    currentSize = 0;
}
```

Defining the methods

```
public String getValue(int index)
{
    if (index >= 0 && index < currentSize)
    {
        return contents[index];
    }
    else
    {
        return null;
    }
}
```

Defining the methods

```
public void setValue(int index, String value)
{
    if (index >= 0 && index < currentSize)
    {
        contents[index] = value;
    }
}
```

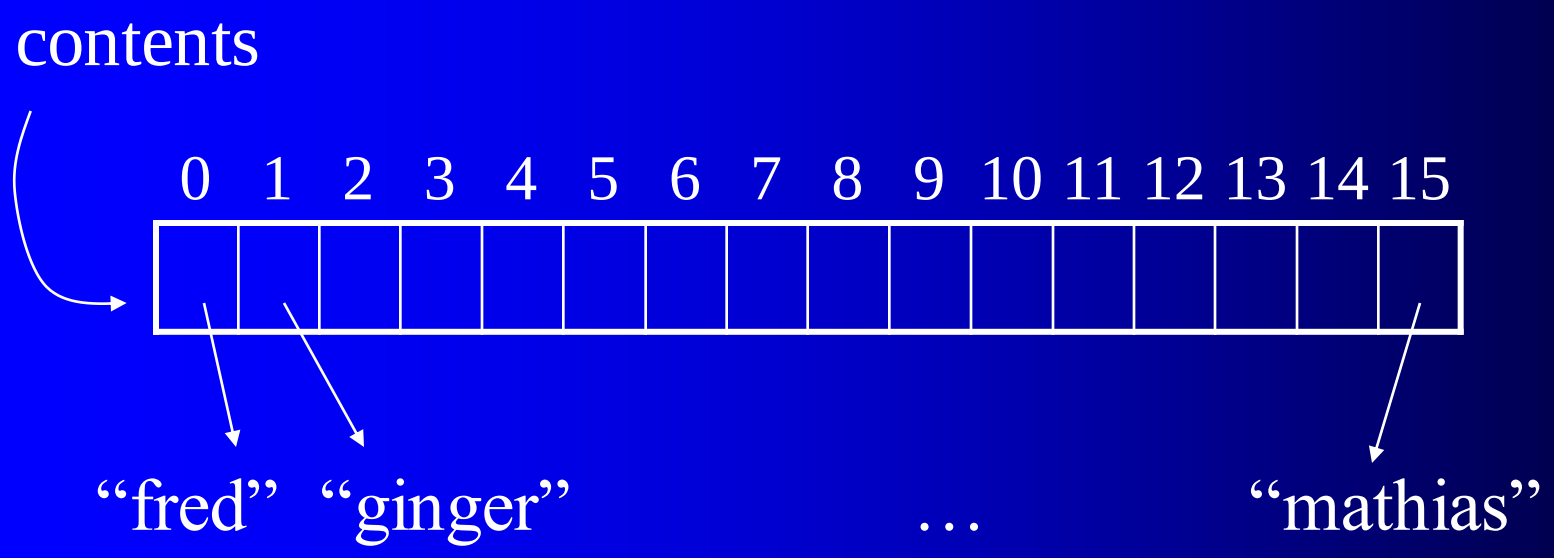
Defining the methods

```
public void display( )
{
    for (int i = 0; i < currentSize; ++i)
    {
        System.out.print(contents[i] + " ");
    }
    System.out.println( );
}
```

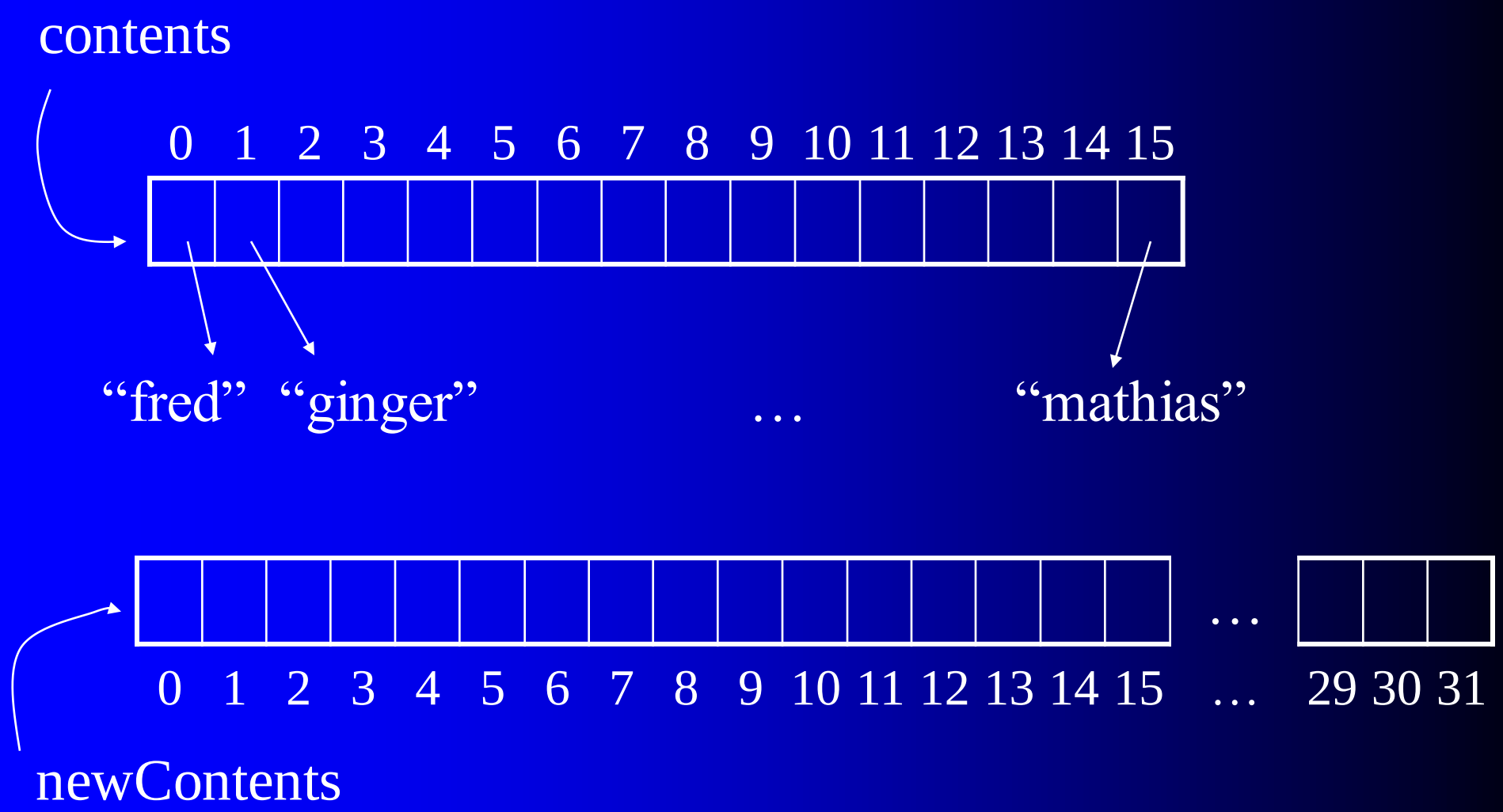
Defining the methods

```
public void addRear(String value)
{
    if (currentSize >= contents.length)
    {
        doubleArray( );
    }
    contents[currentSize] = value;
    ++currentSize;
}
```

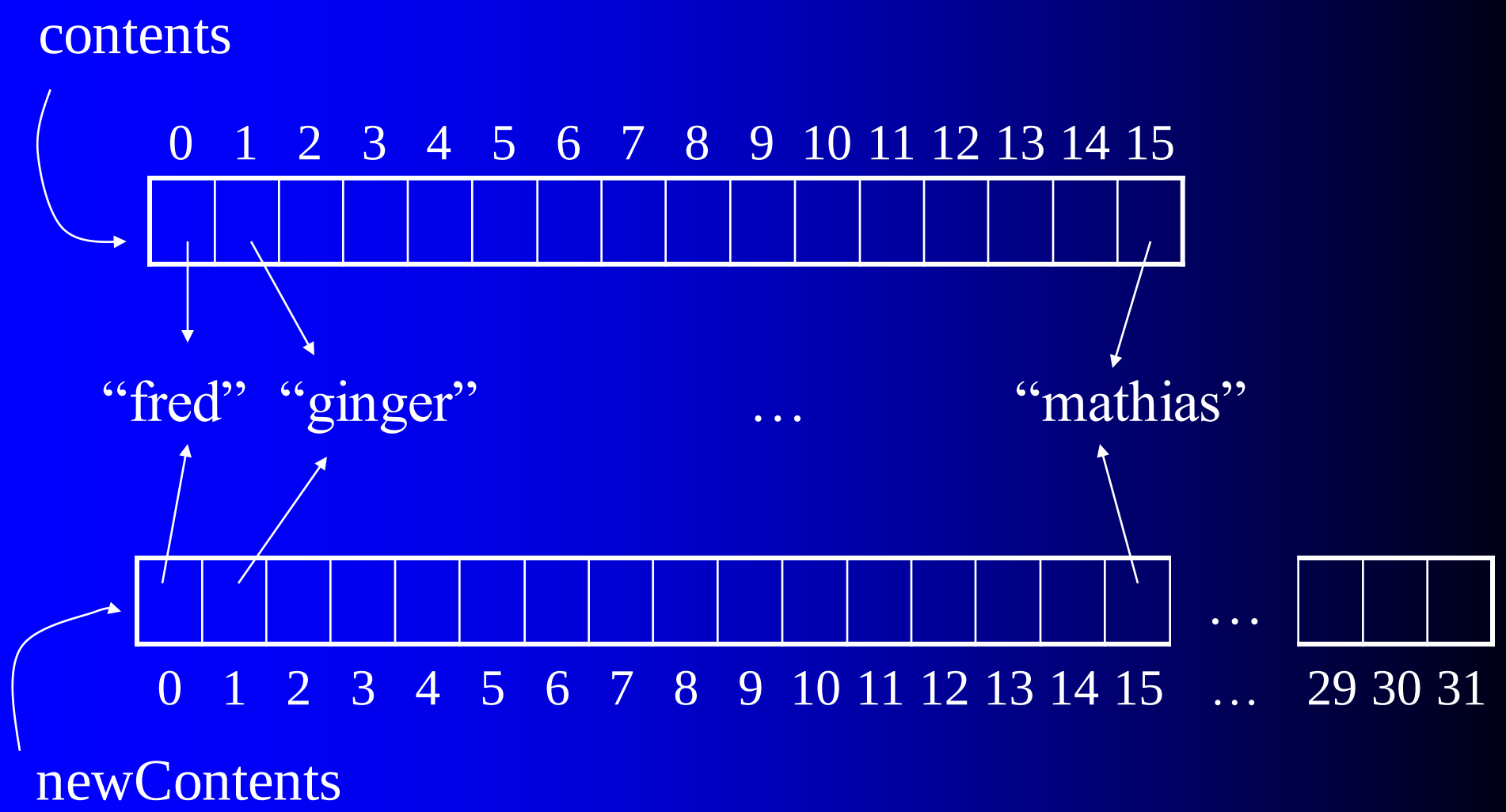
Doubling the array



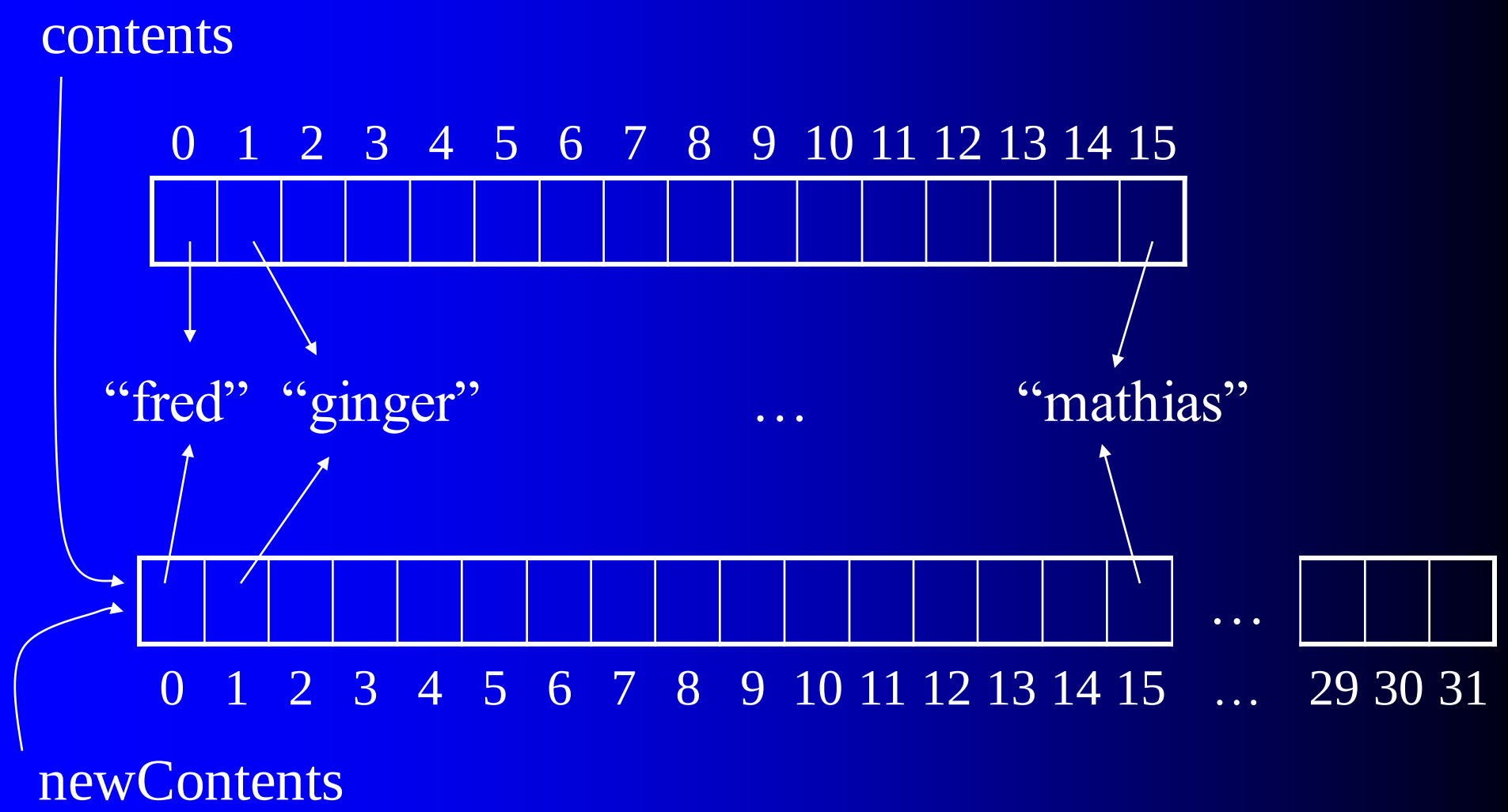
Doubling the array



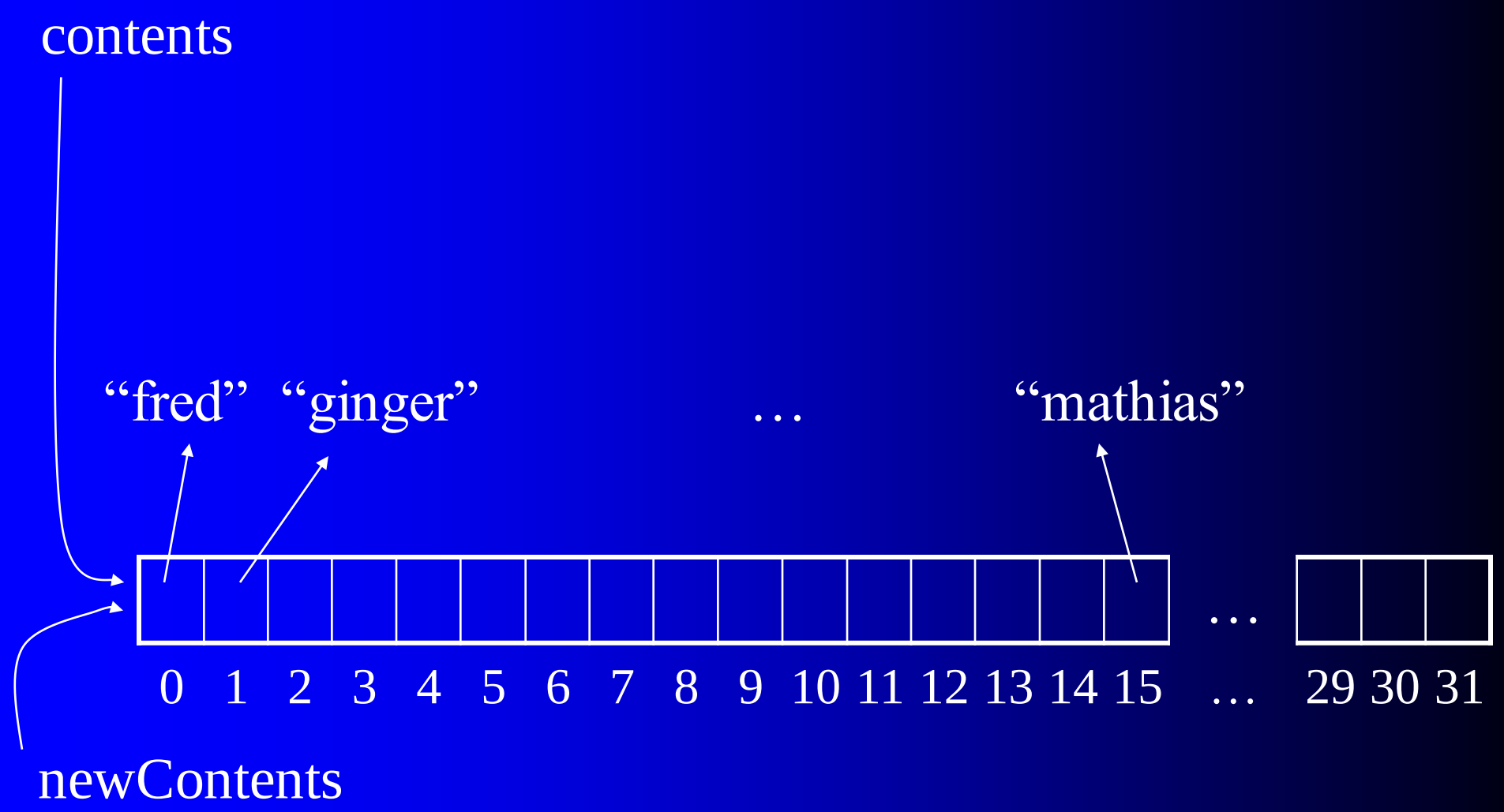
Doubling the array



Doubling the array



Doubling the array



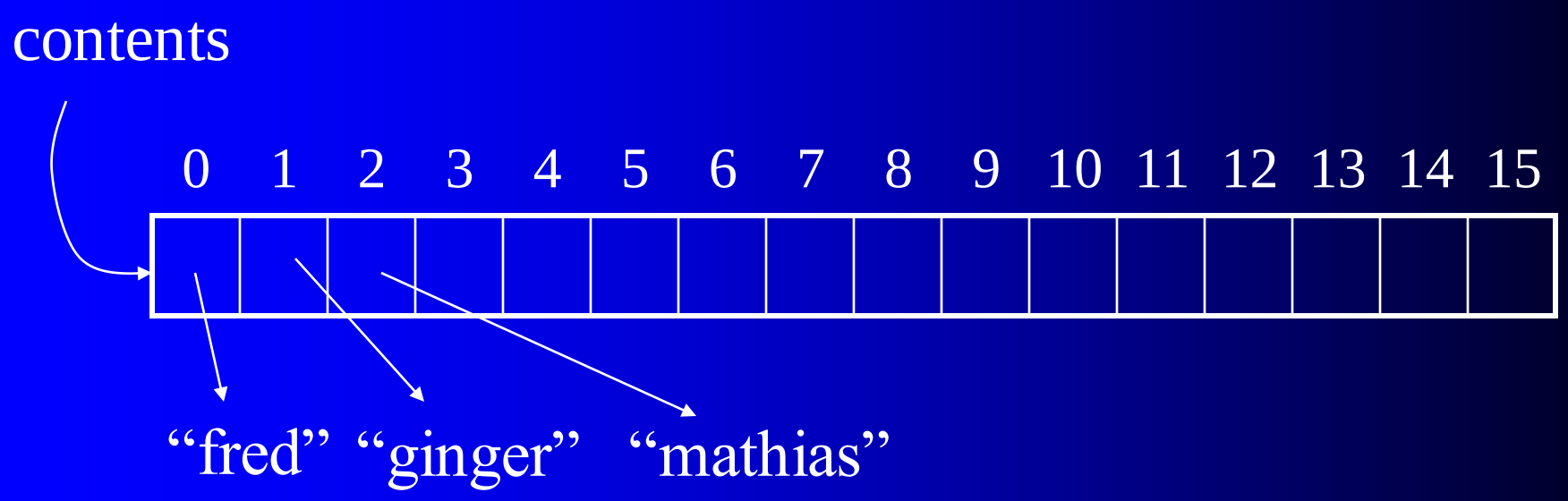
Defining the methods

```
 private void doubleArray( )
{
    String[ ] newContents = new String[contents.length * 2];
    for (int i = 0; i < contents.length; ++i)
    {
        newContents[i] = contents[i];
    }
    contents = newContents;
}
```

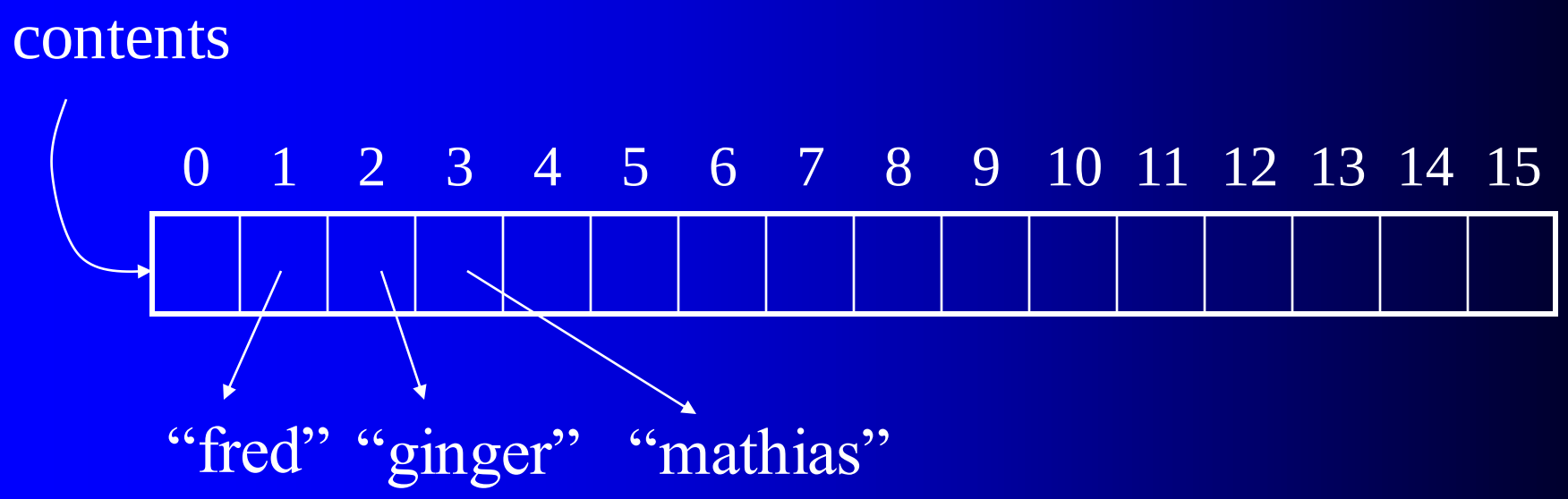
Inserting at the front

- To insert at the front, first we have to make space at the front by moving all the elements up by one space

Inserting at the front



Inserting at the front



** Made room for a new String at the front*

Defining the methods

```
public void addFront(String value)
{
    if (currentSize >= contents.length)
    {
        doubleArray( );
    }
    for (int i = currentSize; i > 0; --i)
    {
        contents[i] = contents[i-1];
    }
    contents[0] = value;
    ++currentSize;
}
```

Class exercises

- Define a method to delete an element at the rear
- Define a method to delete an element at the front

Next lecture

- Searching arrays
- Sorting arrays