CS 234

- Review - Arrays

Arrays and Array Lists

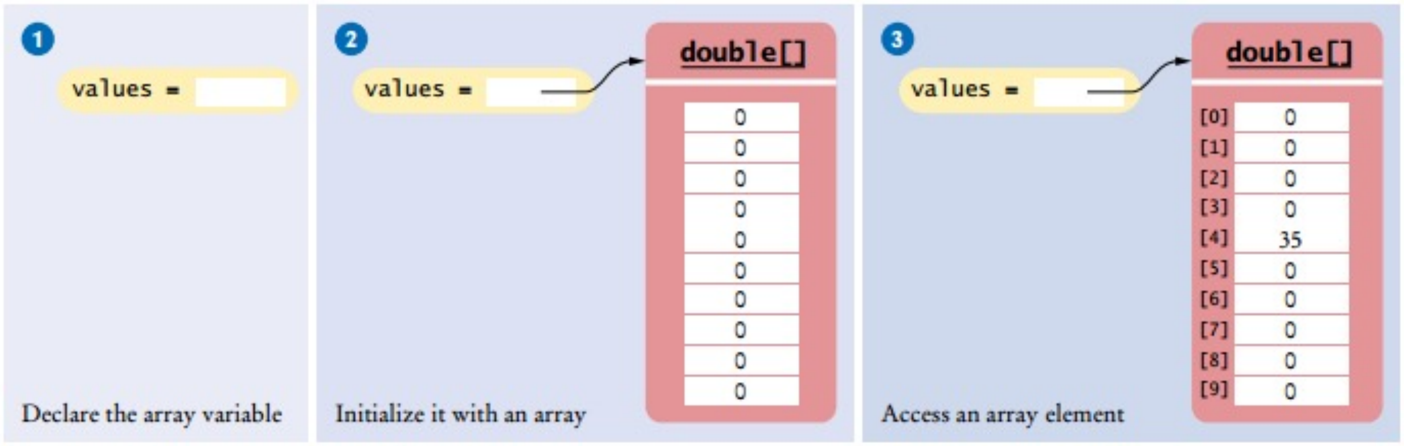
- Why do we need them?
- 1D Arrays
- Enhanced For Loop
- Passing Arrays to Methods
- 2D Arrays
- Array lists

Why do we need them?

- Write a program that reads the grades for 10 students, calculate the group average and count how many students are below and how many are above the average.
- Int student1, student2, ….., student10 ?

Array: a **finite** sequence of values of the **same** type

1D Arrays



```
double[] grades = new double[10];
```

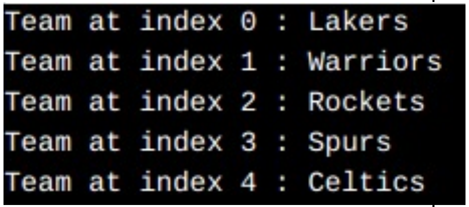
or

```
double[] grades = { 7.0, 8.5, 9.2, 9.5, 7.9, 8.1, 5.2, 6.8, 9.2,8.3 };
```

An array variable points to a memory location

1D Arrays

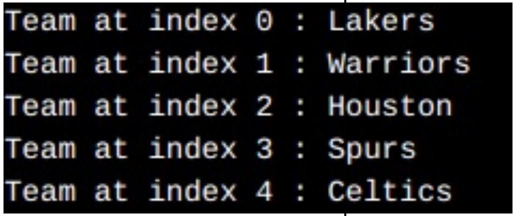
```
public class Main
{
    public static void main(String[] args)
    {
        String[] teams;
        teams = new String[5];

        teams[0] = "Lakers";
        teams[1] = "Warriors";
        teams[2] = "Rockets";
        teams[3] = "Spurs";
        teams[4] = "Celtics";

        for (int i = 0; i < teams.length; i++)
            System.out.println("Team at index " + i + " : "+ teams[i]);
        }
}
```



```
Team at index 0 : Lakers
Team at index 1 : Warriors
Team at index 2 : Rockets
Team at index 3 : Spurs
Team at index 4 : Celtics
```

1D Arrays

```
public class Main
{
    public static void main(String[] args)
    {
            String[] teams;
            teams = new String[5];

            teams[0] = "Lakers";
            teams[1] = "Warriors";
            teams[2] = "Rockets";
            teams[3] = "Spurs";
            teams[4] = "Celtics";

             String[] cities = teams;
             cities[2] = "Houston";

            for (int i = 0; i < teams.length; i++)
                    System.out.println("Team at index " + i + " : "+ teams[i]);

            }
}
```



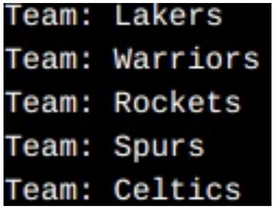
```
Team at index 0 : Lakers
Team at index 1 : Warriors
Team at index 2 : Houston
Team at index 3 : Spurs
Team at index 4 : Celtics
```

`cities` is just a reference to a memory location

Enhanced For Loop

```
public class Main
{
   public static void main(String[] args)
   {
         String[] teams;
         teams = new String[5];

         teams[0] = "Lakers";
         teams[1] = "Warriors";
         teams[2] = "Rockets";
         teams[3] = "Spurs";
         teams[4] = "Celtics";

         for (String team:teams)
                System.out.println("Team: "+ team);
         }
}
```

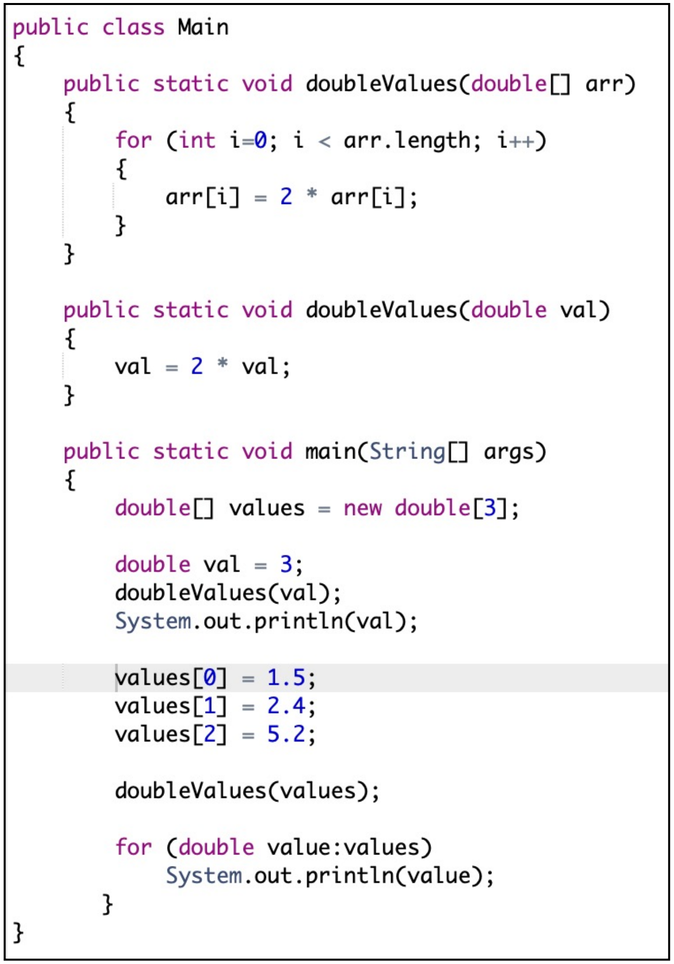
It has a very specific purpose



```
Team: Lakers
Team: Warriors
Team: Rockets
Team: Spurs
Team: Celtics
```

```
for (int i = 0; i < teams.length; i++)
                System.out.println("Team :" + i + " : "+ teams[i]);
```

Passing Arrays to Methods



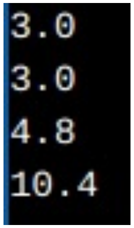
```java
public class Main
{
    public static void doubleValues(double[] arr)
    {
        for (int i=0; i < arr.length; i++)
        {
            arr[i] = 2 * arr[i];
        }
    }

    public static void doubleValues(double val)
    {
        val = 2 * val;
    }

    public static void main(String[] args)
    {
        double[] values = new double[3];

        double val = 3;
        doubleValues(val);
        System.out.println(val);

        values[0] = 1.5;
        values[1] = 2.4;
        values[2] = 5.2;

        doubleValues(values);

        for (double value:values)
            System.out.println(value);
       }
}
```

Do you remember method overloading?

Do you remember the scope of variables?

Do you remember what is stored in an array variable?

So, what is the output for this program?



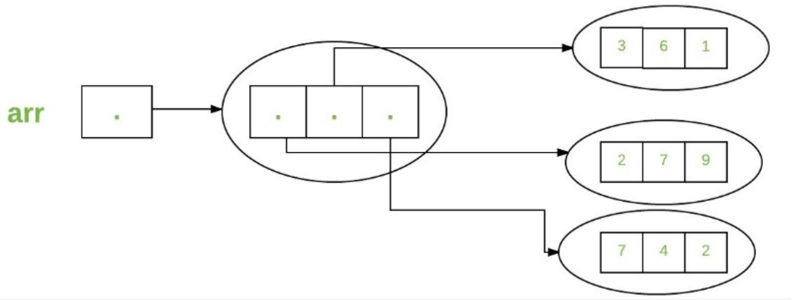
```
3.0
3.0
4.8
10.4
```

Partially filled array

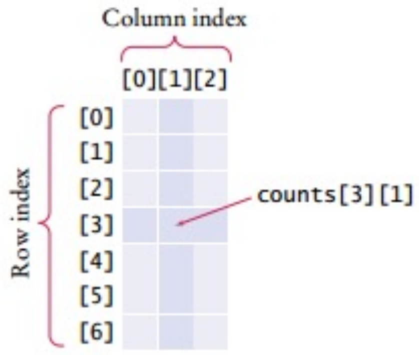
- Is a common practice to create a large array if we do not know the number of elements beforehand
- The array could have only a few values
  - We saw that Java fills up the array with default values (i.e., 0s)
- If we have these types of arrays, we need to **keep track** of how many elements
  - Therefore, we need an additional variable

2D Arrays

A 2D array is basically an array of arrays

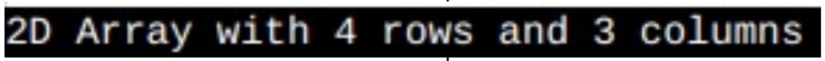


int[ ][ ] counts = new int[7][3]



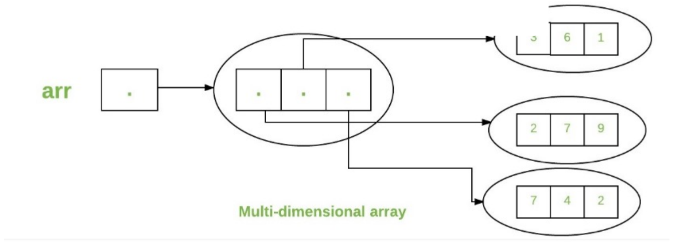
2D Arrays

```
public class Main
{
            public static void main(String[] args) {

                int[][] counts = {
                            { 1, 0, 1 },
                            { 1, 1, 0 },
                            { 0, 0, 1 },
                            { 1, 0, 0 }
                                };
                int rows = counts.length;
                int columns = counts[0].length;

                System.out.printf("2D Array with %d rows and %d columns",
                                    rows, columns);

            }
}
```



```
2D Array with 4 rows and 3 columns
```

How do you traverse a 2D array?



Array List

- Array has some problems
  - Fixed size
  - No methods for inserting and removing (*)

- An improved array

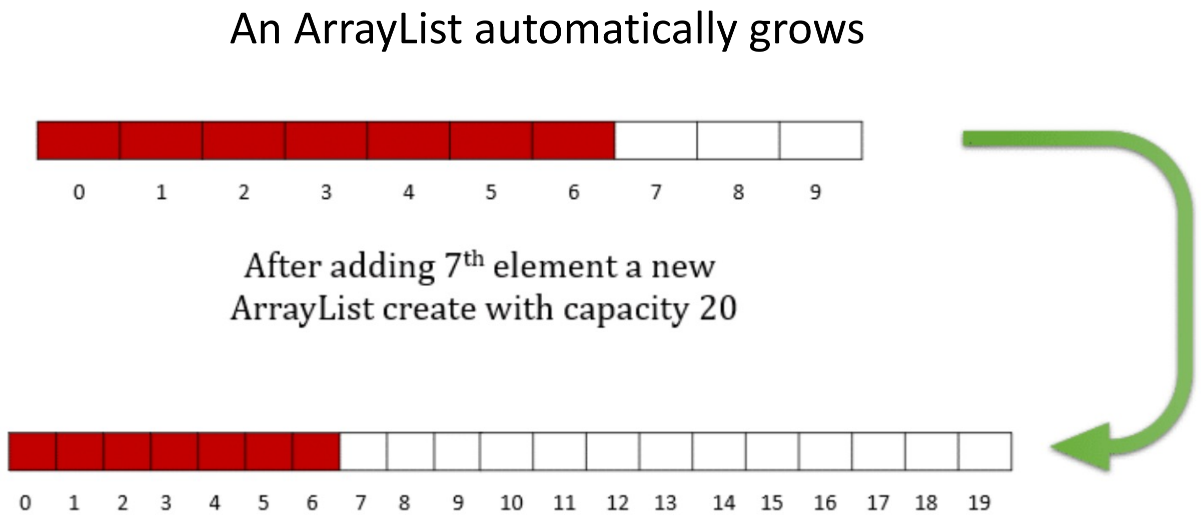
```
import java.util.ArrayList;

ArrayList<String> names = new ArrayList<String>();
```

Data type

Data type

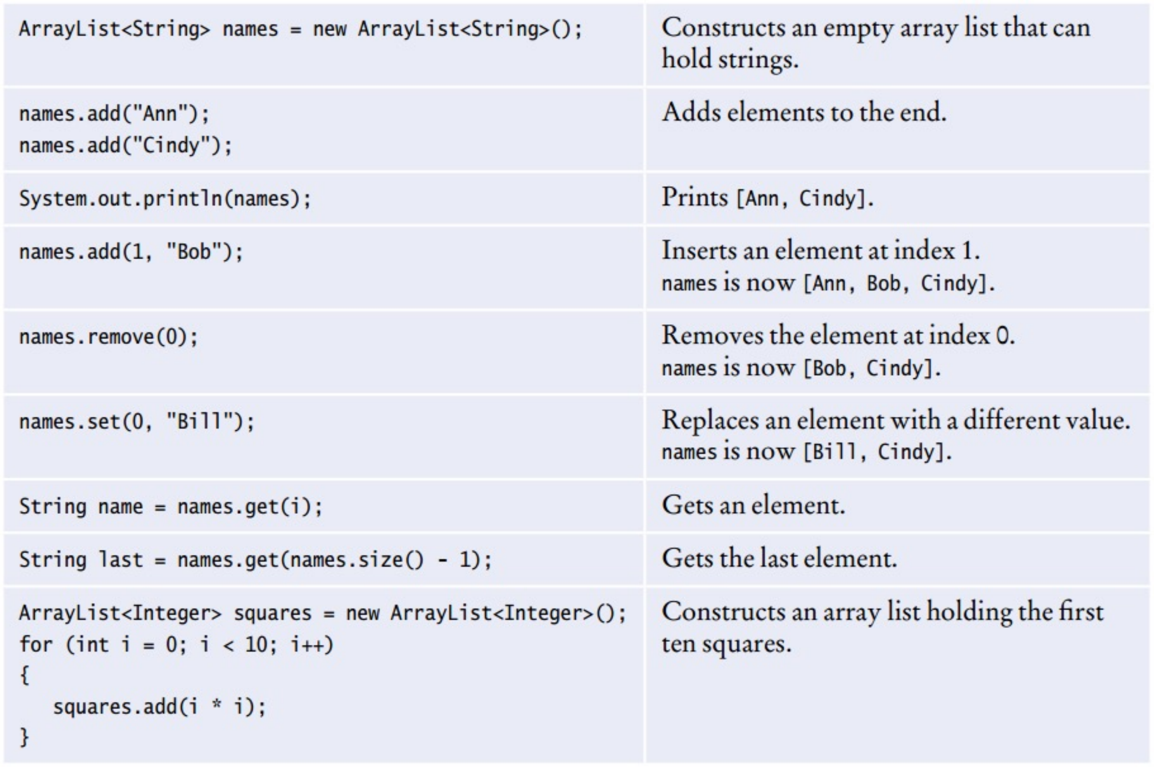
Array List

- Default size: 10 elements
- Load factor: 75%


An ArrayList automatically grows
0
1
2
3
4
5
6
7
8
9
After adding 7th element a new ArrayList create with capacity 20
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

Array List

Methods



`ArrayList<String> names = new ArrayList<String>();`	Constructs an empty array list that can hold strings.

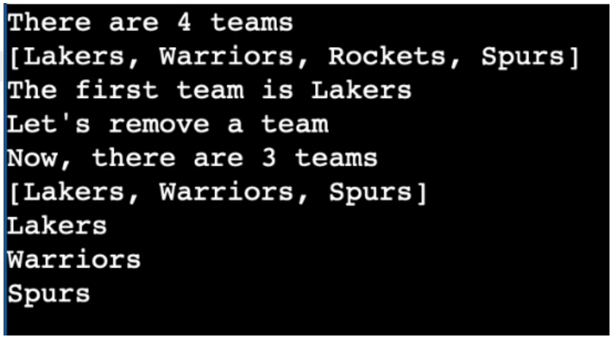
`names.add("Ann");`
`names.add("Cindy");`
	Adds elements to the end.

`System.out.println(names);`	Prints `[Ann, Cindy]`.
`names.add(1, "Bob");`	Inserts an element at index 1. `names` is now `[Ann, Bob, Cindy]`.
`names.remove(0);`	Removes the element at index `0`. `names` is now `[Bob, Cindy]`.
`names.set(0, "Bill");`	Replaces an element with a different value. `names` is now `[Bill, Cindy]`.
`String name = names.get(i);`	Gets an element.
`String last = names.get(names.size() - 1);`	Gets the last element.

`ArrayList<Integer> squares = new ArrayList<Integer>();`
`for (int i = 0; i < 10; i++)`
`{`
`   squares.add(i * i);`
`}`
	Constructs an array list holding the first ten squares.

Array List

```java
import java.util.ArrayList;

public class Main {
  public static void main(String[] args) {
    ArrayList<String> teams = new ArrayList<String>();

    teams.add("Lakers");
    teams.add("Warriors");
    teams.add("Rockets");
    teams.add("Spurs");

    System.out.printf("There are %d teams\n", teams.size());
    System.out.println(teams);

    System.out.printf("The first team is %s\n", teams.get(0));

    System.out.println("Let's remove a team");
    teams.remove(2);
    System.out.printf("Now, there are %d teams\n", teams.size());
    System.out.println(teams);

    // printing each team
    for (String team : teams)
    {
        System.out.println(team);
    }

  }
}
```



```
There are 4 teams
[Lakers, Warriors, Rockets, Spurs]
The first team is Lakers
Let's remove a team
Now, there are 3 teams
[Lakers, Warriors, Spurs]
Lakers
Warriors
Spurs
```