Arrays

Example

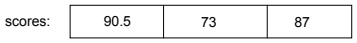
- Write a program to keep track of all students' scores on exam 1.
- Need a *list* of everyone's score
- Declare 14 double variables?
- What about next semester?

Arrays

- Declare a list of variables – all with the *same* type
- Size is determined at time of declaration

```
double[] scores = new double[14];
```

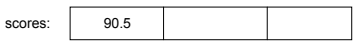
Example



Example

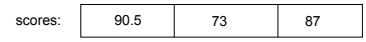

```
//declare the array
double[] scores = new double[3];
//put 90.5 in the first box
```

Example




```
//declare the array
double[] scores = new double[3];
//put 90.5 in the first box
scores[0] = 90.5;
```

Example




```
//declare the array
double[] scores = new double[3];
//put 90.5 in the first box
scores[0] = 90.5;
scores[1] = 73;
scores[3] = 87;
```

Alternative

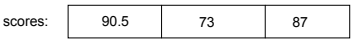
```
double[] scores = {90.5, 73, 87};
```

- Initializes elements of the array to values given when array is created

Subscripts

- Subscript describes which box of the array you are dealing with
- Array of size N has subscripts
  - 0, 1, 2, … (n-1)
  - array of size 3 – 0, 1, 2
  - subscript added to base address of array
- Subscript can be a variable or expression which produces an integer
  - scores[i];
  - scores[i+5];
- If subscript specified does not exist – runtime error (ArrayIndexOutOfBoundsException)

Example




```
//assume i = -2;
sum = scores[1] + scores[2];
sum = scores[1] + scores[i*8];
scores[2] = scores[1] + 6;
```

Accessing Array Elements

- Print all elements of the array *scores*
  - Use only one System.out.println statement

Accessing Array Elements

- Print all elements of the array *scores*
  - Use only one System.out.println statement

```
double[] scores = {90.5, 73, 82};
for(int i = 0; i < 3; i++) {
    System.out.println("Score " + (i+1) + ": " + scores[i]);
}
```

foreach

```
double[] scores = {90.5, 73, 82};
for(double d : scores) {
    System.out.println("Score " + d);
}
```

Passing Array Elements

- Write a method to add two elements of an array

Passing Array Elements

- Write a method to add two elements of an array
  - How is the method called?

```
double add(double num1, double num2) {
   return (num1 + num2);
}
```

Passing Array Elements

- Write a method to add two elements of an array
  - How is the method called?

```
add(scores[0], scores[1]);
…
double add(double num1, double num2) {
   return (num1 + num2);
}
```

Passing Arrays

- Would like to pass an entire array into a method
  - Sum all elements, prompt user for values

Example

```
sum(scores);
```

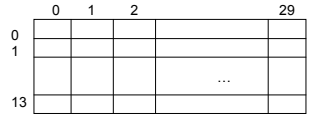
```
static double sum(double[] scores) {
    double sum = 0;
    for(double d: scores) {
        sum+=d;
    }
    return sum;
}
```

Arrays of objects

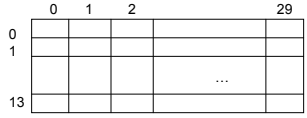
```
Flight[] flights = new Flight[10];
```

Multidimensional Arrays

- double[][] warmups = new double[14][30];
- Declares a multidimensional array with 14 rows and 30 columns

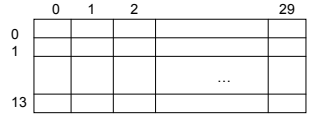


Example



- Set first warmup score for first student to 3

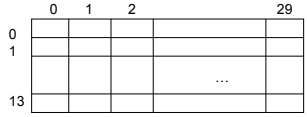
Example



- Set first warmup score for first student to 3

  warmups[0][0] = 3;

Example



- Print all scores for all students

Example

- Print all scores for all students

```
static void printscores(double[][] scores) {
  for(int row = 0; row < scores.length; row++) {
    for(int col = 0; col < scores[row].length; col++) {
      System.out.println("Item " + scores[row][col]);
    }
  }
}
```

ArrayList

- Like an array, but it can dynamically change size
- Stores object references
  - cannot store primitive types without a wrapper
- By default, not declared to store a particular type
  - `ArrayList al = new ArrayList();`
- Because it is a *generic type*, we can explicitly specify the type
  - `ArrayList<String> stringal = new ArrayList<String>();`