Arrays

Arrays

An *array* is simply a collection of objects of the same class (or values of the same type).

- Just like a Python list, a linear array is indexed from 0 to the length - 1.
- Unlike a Python list, objects (or values) must be of the same kind.
- Unlike a Python list, an array cannot grow, i.e., the length is fixed.

Arrays

An array is declared like a variable in the following way:

```
Class[] name = new Class[size];
```

This is actually an abbreviation of

```
Class[] name;
name = new Class[size];
```

Arrays

Example. To declare an array of 100 strings named `myArray`, put

```
String[] myArray = new String[100];
```

This is an abbreviation of

```
String[] myArray;
myArray = new String[100];
```


Arrays

Example. A `for` loop to print all strings in the array `myArray`:

```
for (int i = 0; i < myArray.length; i++)
  System.out.println(myArray[i]);
```


Multi-dimensional arrays

Example. To declare a 5 x 10 array of integers named `table`, put

```
int[][] table = new int[5][10];
```

Multi-dimensional arrays

Example. Nested `for` loops to print all integers in the array `table`:

```
for (int i = 0; i < 5; i++) {
  for (int j = 0; j < 10; j++)
    System.out.print(table[i][j] + " ");
  System.out.println();
}
```


Multi-dimensional arrays

Example. Nested `for` loops to print the sum of all numbers in the array `table`:

```
int sum = 0;
for (int i = 0; i < 5; i++)
  for (int j = 0; j < 10; j++)
    sum = sum + table[i][j];
System.out.println("Sum: " + sum);
```


Multi-dimensional arrays

Example. Nested `for` loops to print the largest number in the array `table`:

```
int max = 0;
for (int i = 0; i < 5; i++)
  for (int j = 0; j < 10; j++)
    if (table[i][j] > max)
      max = table[i][j];
System.out.println("Largest: " + max);
```

Multi-dimensional arrays

Example. Nested `for` loops to print the sum of numbers in each row of the array `table`:

```
int sum = 0;
for (int i = 0; i < 5; i++) {
  sum = 0;
  for (int j = 0; j < 10; j++)
    sum = sum + table[i][j];
  System.out.println("Row " + i + ": " + sum);
}
```


Sorting

Sorting

Given a linear list of *comparable* objects of the same class (or values of the same type), we wish to sort (or reärrange) the objects in the increasing order.

For simplicity, let's just assume that we are given an array of *n* integers.

Sorting

How should we do this? We already know how to do, kind of.

We know how to find the smallest number. We should be able to repeat this to find the smallest, the second smallest, the third smallest, etc.

But we also need to move these numbers to the correct locations.

Selection sort

- Find the least value in the array.
- Swap it with the value in the first index.
- Repeat the steps above for the remainder of the array, starting from the second index and advancing each time.

Selection sort

Suppose we are given an array `A` of `n` integers.

```
for (int i = 0; i < n - 1; i++) {
  int minIndex = i;
  for (int j = i + 1; j < n; j++)
    if (A[j] < A[minIndex])
      minIndex = j;
  if (minIndex != i)
    swap A[i] with A[minIndex];
}
```

Bubble sort

- By comparing and (swapping if necessary) adjacent pairs of numbers, "bubble up" the largest value in the array.
- Repeat the above step for the remainder of the array, ending at the second-to-the last index and advancing each time.

Bubble sort

Suppose we are given an array `A` of `n` integers.

```
for (int i = 1; i < n; i++)
  for (int j = 0; j < n - i; j++)
    if (A[j] > A[j + 1])
      swap A[j] with A[j + 1];
```