Arrays
Arrays store multiple values

- An array lets you associate one name with a fixed (but possibly large) number of values
- All values must have the same type
  - *ints*, *Strings*, *Person* (but can have different subclasses of *Person*)
- The values are distinguished by a numerical *index* between 0 and array size minus 1
  - An array index is sometimes called a *subscript*
Strings and arrays

• Strings and arrays both have special syntax
• Strings are objects, and can be used as objects
• Arrays are objects, but
  – Arrays are created using special syntax:
    – new type[size] instead of new Person()
• If an array holds elements of type T, then the array’s type is “array of T”
  – An array’s size is not part of its type
Array assignment

• When you assign an array value to an array variable, the types must be compatible
• The following is not legal:
  
  ```java
  double[ ] dub = new int[10]; // illegal
  ```
• The following is legal:
  
  ```java
  int[ ] myArray = new int[10];
  ```
  
  ...and later in the program,
  
  ```java
  myArray = new int[500]; // legal!
  ```
  
  – This is legal because the array’s size is not part of its type, but part of its value
• Array assignment does not copy values
  
  ```java
  int[ ] a1; int[ ] a2;
  a1 = new int[10];
  a2 = a1; // a1 and a2 refer to the same array
  ```
Length of an array

- Arrays are objects
  - Every array has an instance constant, `length`, that tells how large the array is
  - Example:
    
    ```java
    for (int i = 0; i < scores.length; i++)
    System.out.println(scores[i]);
    ```

- Use of `length` is always preferred over using a constant such as 10

- Arrays have a `length variable`, Strings have a `length()` method
Stepping through an array

• The for loop is ideal for visiting every value in an array
• The form is: `for (int i = 0; i < myArray.length; i++) {...}`
  – Make sure you use `length` instead of a hardcoded number to end the loop
• In general we like to use meaningful names for variables, but in his case, the name `i` is traditional, and better
  – `i` is instantly recognizable as the index of an enclosing for loop
  – Inner (nested) loops should use `j`, then `k`
• Example: find the largest value in an array

```java
int largestScore = scores[0];
for (int i = 1; i < scores.length; i++) {
    if (scores[i] > largestScore) {
        largestScore = scores[i];
    }
}
```
Arrays of objects

- Suppose you declare and define an array of objects:
  ```java
  PezDispenser[] arrayOfPez = new PezDispenser[3];
  ```
- You have given a value to the array named `arrayOfPez`, but you haven’t yet given values to each element in that array
- There is nothing wrong with this array, but
  - it has 3 null references to `PezDispensers` in it
  - you have not yet defined 3 `PezDispensers`
  - For example, `arrayOfPez[2].capacity` will give you a `nullPointerException`
- Eclipse example: `ArrayOfPez.java`
Initializing arrays in one line

• There is a special syntax for giving initial values to the elements of arrays
  – This syntax can be used in place of new `type[size]`
  – It can only be used in an *array declaration*
  – The syntax is: `{ value, value, ..., value }`

• Examples:

  ```java
  int[ ] primes = { 2, 3, 5, 7, 11, 13, 19 };
  String[ ] languages = { "Java", "C", "C++" };
  PezDispenser[] pezArray = {new PezDispenser("Yoshi", 12), new PezDispenser("Bunny", 12)};
  ```
Arrays of arrays

• The elements of an array can themselves be arrays
  ```java
  int[][] table = new int[3][2];
  int[][] table = {{1, 2}, {3, 6}, {7, 8}};
  ```
  – Note: `table.length` is 3
  – But `table[0].length` is 2

• However, you can also have non-rectangular arrays
  ```java
  table[2] = new int[50];
  ```

• **Remember**: each array element is a pointer to an array of type `int[]` but the array size is not part of its type!
Array methods

• Very few methods/attributes
  – Basically just **length**
  – No **add**, **remove**, **reverse**, etc.
  – If you use an index outside of 0 up to **myArray.length-1**, you’ll get an **ArrayIndexOutOfBoundsException**

• Cannot be easily resized
  – Have to create a new array, copy everything from the old array and add the new elements
  – But this is what Python is doing under the hood anyway
  – Python might save you some coding time but not actual running time