Class #24: Introduction to Arrays

Software Design I (CS 120): D. Mathias, 14 Oct 19

Creating Multiple Objects

- Suppose we want to write a program that works with 50 different integers
  - Each may need to be changed at some point
  - Each will need own name so we can access it
- One approach is to create **50 different variables**
  - Very time consuming!
  - What if we don't know whether we need 50, or 100, or 1000 beforehand?

```
int num1;
int num2;
int num3;
int num4;
int num5;
int num6;
int num7;
int num8;
int num9;
...  
int num50;
```

Another Approach: Using Arrays

- Rather than create our 50 distinct int variables, we can create a single array of integers
  - A collection of objects, all of the same type
  - Length of the array set at initialization and never changes
  - Each ‘cell’ of the array contains a single element
  - The integer index of the slot is part of the “name” of the element

\[
\begin{bmatrix}
\end{bmatrix}
\]

\[
\text{Array } A = \begin{bmatrix}
\end{bmatrix}
\]

- A[0] is the 1st element
- A[2] is the 3rd element
- A[length - 1] is the last element

Creating Arrays

- Like a Java class instance, arrays are declared and instantiated

```
// Array Declaration
TypeName[] arrayVariableName;

// Array Instantiation
arrayVariableName = new TypeName[arraySize];
```

Note: size must be a fixed, non-negative integer

```
int[] intArray;
String[] strArray;
Oval[] ovalArray;
```

```java
intArray = new int[20];
strArray = new String[100];
ovalArray = new Oval[1000];
```
Creating Arrays

- We can create an array of 100 integers, or 1,000 doubles, or 2,200 Strings, or...
  - ```java
    int[] integerArray = new int[100];
    double[] doubleArray = new double[1000];
    String[] stringArray = new String[2200];
  ```

- We can create arrays with some random size as well (we just can’t change it after it is first set)
  - Since we can never change this size once an array is created, we must ensure that whatever size we choose is enough for our needs
  - ```java
    int arraySize = (int)(Math.random() * 100 + 1);
    double[] doubleArray = new double[arraySize];
  ```

Adding elements at initialization

- One way to add data to an array is to **explicitly** input it
  - Can be done once, at time that array is declared and initialized
    - ```java
        int[] intArray = { 0, 15, 30, 45, 60 };
        String[] strArray = { “Hello”, “There” };
        Oval[] ovalArray = { new Oval(0, 0, 10, 10) };
      ```
  
  **Important Note:** each array now has **fixed size** (5, 2, or 1)

- Another possibility: add/change data **after** initialization
  - Uses basic array operations: access and replacement
    - ```java
        intArray[intArray.length - 1] = 3;
        ovalArray[1000] = new Oval(0, 0, 10, 10);
      ```

Basic Array Operations (1)

**Array Access**
- `variable = arrayName[index];`
- ```java
    int i = intArray[0];
    String s = strArray[15];
    Oval o = ovalArray[500];
  ```

**Array Replacement**
- `arrayName[index] = expression;`
- ```java
    intArray[intArray.length - 1] = 3;
    ovalArray[1000] = new Oval(0, 0, 10, 10);
  ```

- `index` must be in range: `0...(length - 1)`
- Variables being assigned must each be of **same type** as elements stored in the array
- `arrayName.length` gives the length of array

- `index` must be in range: `0...(length - 1)`
- The **expression** must evaluate to object of **same type** as the elements stored in array
Filling Arrays

- While we can always fill an array by initializing it explicitly and directly, this is really only manageable for smaller arrays:

  ```java
  int[] integerArray = { 1, 2, 3, 4, 5 }; double[] doubleArray = { 1.1, 2.2, 3.3, 4.4, 5.5 }; 
  ```

- For larger arrays, we most often will write code to do this process for us, often using loops to do the filling.

  Since array index values are just integer values 0…(length – 1), it is natural to loop over these values when we fill the array:

  ```java
  int[] integerArray = new int[500];
  for ( int i = 0; i < integerArray.length; i++ )
  { integerArray[i] = (int)( Math.random() * 1000 ); }
  ```

An Example: Creating an Array of Words

- If we let a user enter any String they like, we can then create an array to hold all the words from that input:

  ```java
  Scanner scan = new Scanner( input.getText() ); int counter = 0;
  while ( scan.hasNext() ) { scan.next(); counter++; }
  scan = new Scanner( input.getText() ); String[] words = new String[counter];
  ```

- Step One: count how many words there are
- Step Two: reset the Scanner
- Step Three: create the array

An Example: Filling an Array of Words

- Once we create array, we can iterate over it:

  ```java
  words = new String[counter];
  for ( int i = 0; i < words.length; i++ )
  { String str = scan.next(); words[i] = str; }
  for ( int w = 0; w < words.length; w++ )
  { System.out.println( words[w] ); }
  ```

- Loop One: add elements to array
- Loop Two: access all elements of the array

This Week:

- Meetings this week:
  - Monday/Tuesday/Wednesday: regular classroom
  - Friday: in the CS Lab (16 Wing)
- Program 04: due 11:59 PM, Wednesday 16 October
- Reading: Ch. 5 due Thursday 17 October by Noon
- Office Hours: Wing 212
  - Monday/Friday: 2:15 PM–3:15 PM
  - Tuesday: 1:30 PM–2:30 PM
  - Wednesday: 12:05 PM–1:00 PM
- Lab and Tutor Hours: On my website