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?

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

Creating Arrays

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

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

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

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

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

```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

Basic Array Operations (1)

**Array Access**

```java
variable = arrayName[index];
```

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

```java
int i = intArray[0];
String s = strArray[15];
Oval o = ovalArray[500];
```

Basic Array Operations (2)

**Array Replacement**

```java
arrayName[index] = expression;
```

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

```java
intArray[intArray.length - 1] = 3;
strArray[15] = "Hello " + "there";
ovalArray[1000] = new Oval(0,0,10,10);
```

`arrayName.length` gives the length of 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

- 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] );
  }
  ```

**This Week:**

- **Meetings this week:**
  - Monday/Tuesday/Friday: regular classroom
  - Wednesday: in the CS Lab (16 Wing)

- **Next week:** *Spring Break!*

- **Program 03:** due Wednesday 11 March by 11:59 PM

- **Office Hours:** Wing 212
  - Monday/Wednesday/Friday: 11:00 AM–12:00 PM
  - Tuesday: 3:15 PM–4:15 PM

- **Lab and Tutor Hours:** On my website