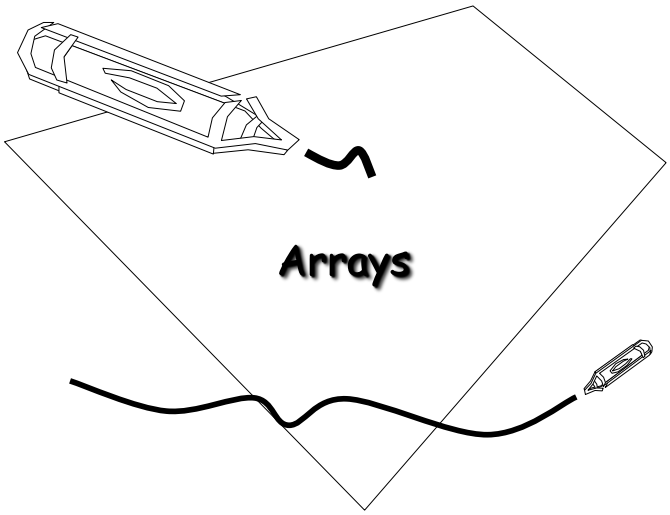


Arrays

- Arrays allow data and objects to be grouped together in fixed size collections
  - Multiple elements of data of the same type
  - Index individual elements by their position
  - Perform same operation(s) on each of the elements of the collection

Arrays …

- Arrays of primitives vs. arrays of objects
  - Array elements of primitives are values
    - initial value is "zero"
  - Array elements of objects are references
    - initial reference is `null`

Arrays …

- Declaring Arrays:
  - `int[] intArray;`
  - `int intArray[];`
  - `String[] stringArray;`
  - `String stringArray[];`

Arrays …

- Creating Arrays:
  - `intArray = new int[10];`
  - `stringArray = new String[10];`

Arrays …

- Initializing Arrays:

```
intArray[0] = 3;
intArray[1] = 7;
intArray[2] = 1;
intArray[3] = 9;
…
intArray[8] = 2;
intArray[9] = 7;
```

Arrays …

- The number of elements in an array:
  - **`public int length;`**

Arrays …

- Indexing/initializing array elements:

```
int[] a = new int[10];
for (int i = 0; i < a.length; ++i) {
  a[i] = i;
}
int[] b = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

StringBuffer[] c = new StringBuffer[4];
for (int i = 0; i < c.length; ++i) {
  c[i] = new StringBuffer(String.valueOf(i));
}
StringBuffer[] d = {new StringBuffer("0"),
                    new StringBuffer("1"),
                    new StringBuffer("2"),
                    new StringBuffer("3")};
```

Arrays …

- Finding the maximum value in an array:

```
int[] numbers = new int[SIZE];
int max = numbers[0];
int maxIndex = 0;
for(int i = 1; i < numbers.length; ++i) {
  if(numbers[i] > max) {
    max = numbers[i];
    maxIndex = i;
  }
}
// max contains the maximum value in the array
// maxIndex contains the index of the maximum value
```

Arrays …

- Let's look at arrays in DrJava

Arrays as parameters

- Passing an array as an argument to a method:

```
public void setAllAmounts(int[] amount) {
  for(int i = 0; i < rainfall.length; ++i)
    rainfall[i] = amount[i];
}

public void otherCode() {
  int[] myArray;
  myArray = {2, 5, 3, 0};
  setAllAmounts(myArray);
}
```

Arrays as parameters …

- Returning an array as the result from a method:

```
public int[] compareToAverage() {
  int[] result = new int[rainfall.length];
  double average = getAverage();
  for (int i = 0; i < rainfall.length; ++i) {
    result[i] = (rainfall[i] == average) ? 0 :
                (rainfall[i] < average) ? -1 : 1;
  }
  return result;
}
```

Array indices

- Array index can be an integral expression of type **`char`**, **`short`**, **`byte`**, or **`int`**.
- It is programmer's responsibility to make sure that an array index does not go out of bounds. The index must be within the range 0 through the array's length minus 1.
- Using an index value outside this range throws an **`ArrayIndexOutOfBoundsException`**. Prevent this error by using the public instance method **length**.

Arrays of objects

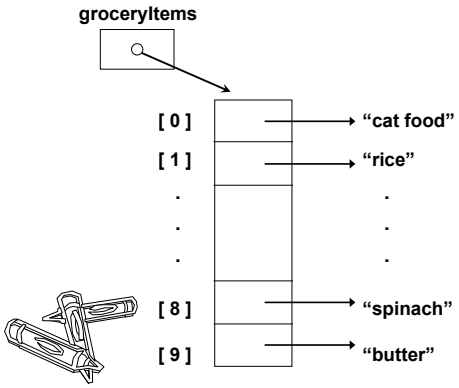
- The elements of an array can be object references.
- The following declaration and initialization:

```
Integer[] items = new Integer[10];
```

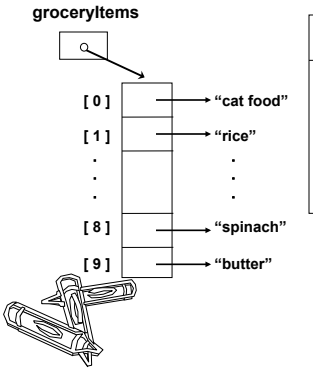
  does not create the **`Integer`** objects
  - it merely creates the reference
- Each object in the referenced array must be instantiated separately:

```
for(int i = 0; i < items.length; ++i) {
    items[i] = new Integer(i);
}
```

```
String[] groceryItems = new String[10];
```

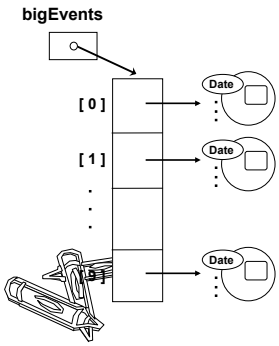



```
String[] groceryItems = new String[10];
```



Expression	Class/Type
groceryItems	Array
groceryItems[0]	String
groceryItems[0].charAt(0)	char

```
Date[] bigEvents = new Date[10];
```

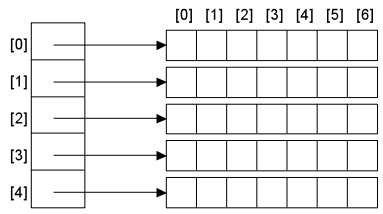


Expression	Class/Type
bigEvents	Array
bigEvents[0]	Date
bigEvents[0].month	String
bigEvents[0].day	int
bigEvents[0].year	int
bigEvents[0].month.charAt(0)	char

Multi-Dimensional Arrays

- Java does not support multi-dimensional arrays, but rather supports arrays of objects and since an array is an object it supports arrays of arrays

Multi-Dimensional Arrays ...



Multi-Dimensional Arrays ...

- Declaring and initializing a two-dimensional array (method #1):

```
final int numRows = 5, numCols = 7;
int[][] matrix = new int[numRows][numCols];
for (int i = 0; i < numRows; ++i) {
  for (int j = 0; j < numCols; ++j) {
    matrix[i][j] = i*numCols + j;
  }
}
```

Multi-Dimensional Arrays ...

- Declaring and initializing a two-dimensional array (method #2):

```
final int numRows = 5, numCols = 7;
int[][] matrix = new int[numRows][];
for (int i = 0; i < numRows; ++i) {
  matrix[i] = new int[numCols];
  for (int j = 0; j < numCols; ++j) {
    matrix[i][j] = i*numCols + j;
  }
}
```