ENSF 593/594

3 – Introduction to Arrays and Lists

Introduction to Arrays

- Although arrays are not objects, but they are treated much like objects.
  - Manipulated by reference.
  - Created dynamically at run time with the new operator.
  - Garbage collected when no longer referred to.
  - Cannot be subclassed.
- You can create an array of primitive types or object references.

One Dimensional Arrays

- Declaring an array of primitive types:

```
char []charArray = {‘h’, ‘e’,’l’, ‘l’, ‘o’};
byte [] byteArray = {(byte) ‘H’, (byte) ‘i’};
int[] myarray = new int[5];

int myarray[] = new int[5];
```

array reference

- The length of the array is set when it is created using new, and cannot be changed.
  - Could create another array of a different size, and assign it to the array reference `myarray`.

One Dimensional Arrays (continued)

- The length of the array is available using the length field. E.g. `myarray.length`
- Array elements are numbered 0 to length-1, and are accessed using array subscripts. E.g.

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

If an array subscript is ever out of range, an IndexOutOfBoundsException is thrown.

One Dimensional Arrays (continued)

- Declaring an array of object references:

```
Point[] p = new Point[3];
```

array data type

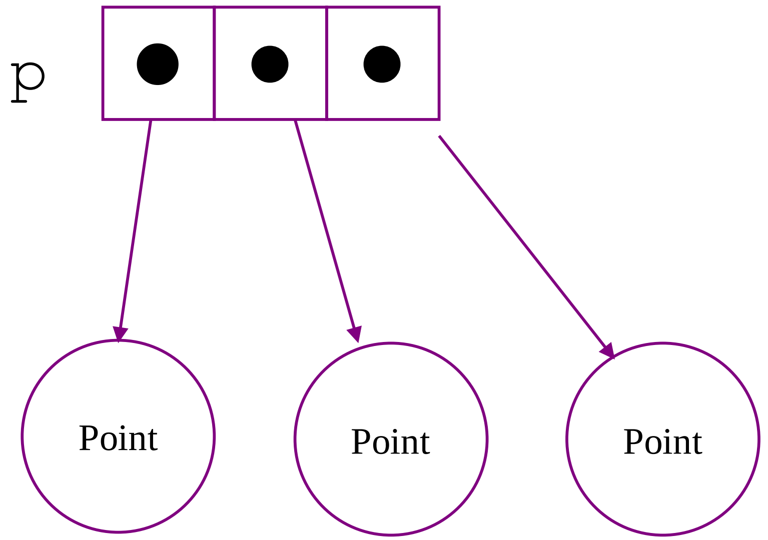
array reference

creates the array dynamically

This does *not* create 3 Point objects, but only an array of object references. You must create the actual objects in a loop:

```
for (int i = 0; i < p.length; i++)
    p[i] = new Point();
```

One Dimensional Arrays (continued)



One Dimensional Arrays (continued)

- Arrays can be created and initialized when declared using braces:

```
String[] st = {"Larry", "Curly", "Moe"};
```

- This is the same as:

```
String[] st = new String[3];

st[0] = "Larry";

st[1] = "Curly";

st[2] = "Moe";
```

One Dimensional Arrays (continued)

- To access an element within an array, use the [] operator.
- A convenient way to traverse an array is to use the ***for*** loop operator. Use the length attribute to get the number of elements in an array.

```
int[] a;

a = new int[] {1,2,3};// an array with three
                      // elements holding 1, 2, 3
for (int i = 0; i < a.length; ++i)
    System.out.println(a[i]);
```

One Dimensional Arrays (continued)

- The class System has a method called ***arraycopy*** that allows copying elements of one array to another. Copies all three elements of **a** into array **b**, starting at the second element of b.

```
int[ ] a = {5, 21, 30};
int[ ] b = new int[5];

System.arraycopy(a, 0, b, 1, a.length);

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

Class java.util.Arrays

- Class **java.util.Arrays** is a utility class that provides several useful methods, including:
  - **binarySearch**: Searches a specified array for a specified value in a SORTED array.

```
int [] b = new int[5] {2. 3, 23, 5, ;
int index = Arrays.binarySearch(b, 23);
```

  - **equals -** Returns true if two specified arrays are ***equal*** to one another.
  - **fill -** Assigns a specified value to each element of a specified array.

```
int [] b = new int[5];
Arrays.fill(b, 23);
```

  - **sort -** Sorts a specified array into ascending order. Example:

```
int [] array = new int[5]  {3, 2, 1, 4, 5};
Arrays.sort(array);
```

  - These algorithms work on arrays of Objects and also on array of every primitive data type.

Multidimensional Arrays

- Tables with rows and columns
    - Two-dimensional array
    - Declaring two-dimensional array:

```
int b[][] = { { 1, 2 }, { 3, 4 } };
  – 1 and 2 initialize b[0][0] and b[0][1]
  – 3 and 4 initialize b[1][0] and b[1][1]
int b[][] = { { 1, 2 }, { 3, 4, 5 } };
  – row 0 contains elements 1 and 2
  – row 1 contains elements 3, 4 and 5
```

Multidimensional Arrays

- Can be allocated dynamically

```
int b[][];
b = new int[ 3 ][ 4 ];
```

  - Rows can have different number of columns

```
int b[][];
b = new int[ 2 ][ ];    // allocates rows
b[ 0 ] = new int[ 5 ]; // allocates row 0
b[ 1 ] = new int[ 3 ]; // allocates row 1
```