# Programming in Java: lecture 8

- Arrays
- Creating and using Arrays
- Programming with Arrays
- Random Access
- Arrays of Objects
- Variable Arity methods
- Dynamic Arrays and ArrayList

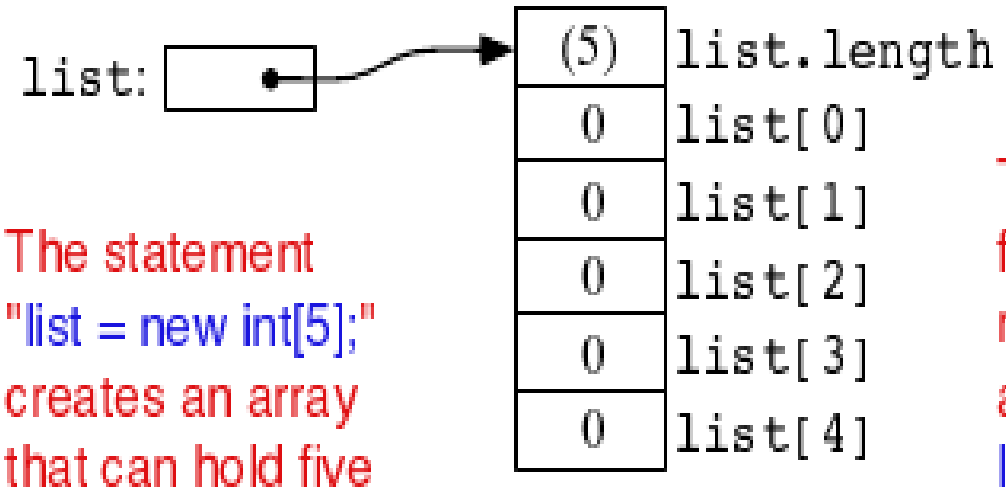
Slides made for use with ”Introuction to Programming Using Java, Version 5.0” by David J. Eck
Some figures are taken from ”Introuction to Programming Using Java, Version 5.0” by David J. Eck
Lecture 3 covers Section 5.5 to 5.7

# Arrays

- A list of items of the same type
  - Other programming languages
    - records
    - struct
  - Arrays are Objects
  - ArrayList
  - Vector
  - ArrayList<BaseType>

# Arrays are Objects

- int[] list; // declares the reference
- list = new int[5]; // creates an Array of size five
- list[0] = 44; // puts 44 in the first position of list




The statement "list = new int[5];" creates an array that can hold five ints, and sets list to refer to it.

The array object contains five integers, which are referred to as list[0], list[1], and so on. It also contains list.length, which gives the number of items in the array. list.length can't be changed.

# Initialization

⟨*array-variable*⟩ [ ⟨*integer-expression*⟩ ]

- Shape[] shapes = new Shape[10];
  - has null values
- double[] doubles = new double[100];
  - has 0 values
- boolean[] b = new boolean[30];
  - has default of false

# Arrays Initializers

```
int[] list = { 1, 4, 9, 16, 25, 36, 49 };
```

```
Color[] palette = {
            Color.BLACK,
            Color.RED,
            Color.PINK,
            new Color(0,180,0),  // dark green
            Color.GREEN,
            Color.BLUE,
            new Color(180,180,255),  // light blue
            Color.WHITE
          };
```

# Array Initializers

```
list = new int[] { 1, 8, 27, 64, 125, 216, 343 };
```

- 

```
new ⟨base-type⟩ [ ] { ⟨list-of-values⟩ }
```

- In place array creation

```
makeButtons( new String[] { "Stop", "Go", "Next", "Previous" } );
```

# Using Arrays

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

- ArrayIndexOutOfBoundsException

```
int count = 0;
for (int i = 0; i < A.length - 1; i++) {
    if (A[i] == A[i+1])
        count++;
}
```

# Array Copy

```
double[] B = A;
```

```
double[] B = new double[A.length]; // Make a new array object,
                                   //   the same size as A.
for (int i = 0; i < A.length; i++)
   B[i] = A[i];    // Copy each item from A to B.
```

```
public static void arraycopy(Object sourceArray, int sourceStartIndex,
        Object destArray, int destStartIndex, int count)
```

```
double B = new double[A.length];
System.arraycopy( A, 0, B, 0, A.length );
```

# Arrays in for-each loops

```
for ( BaseType item : anArray ) {
    .
    .   // process the item
    .
}
```

- same as

```
for ( int index = 0; index < anArray.length; index++ ) {
   BaseType item;
   item = anArray[index];  // Get one of the values from the array
      .
      .   // process the item
      .
}
```

# for each loops

```
int sum = 0;    // This will be the sum of all the positive numbers in A
for ( int item : A ) {
   if (item > 0)
      sum = sum + item;
}
```

```
int[] intList = new int[10];
for ( int item : intList ) {   // INCORRECT! DOES NOT MODIFY THE ARRAY!
   item = 17;
}
```

# Random Access

- You can go directly to any data value
- Other data structures where this is not the case
  - linked lists
- Inserting and removing is expensive
- Access is cheap
- Best for static size

# Variable Arity Methods

```
public static double average( double... numbers ) {
   double sum;       // The sum of all the actual parameters.
   double average;   // The average of all the actual parameters.
   sum = 0;
   for (int i = 0; i < numbers.length; i++) {
      sum = sum + numbers[i];  // Add one of the actual parameters to the sum.
   }
   average = sum / numbers.length;
   return average;
}
```

# Dynamic Arrays

- Arrays containing an array

# Parameterized Types

```
ArrayList<ColoredRect> rects;
```

# Example

- Team programming