Object-Oriented Programming I

Arrays

Slides by Magdin Stoica
and Georg Feil

Learning Outcomes

1. Explain the need for methods of organizing large amounts of data in a computer program
2. Characterize an array as both a variable and a container of variables
3. Define the syntax used to declare an array of variables of a given type
4. Define the principal characteristics of an array
5. Compare and contrast an array with an object as two distinct methods of grouping related variables
6. Create programs that use arrays of primitive typed variables
7. Create programs that use arrays of objects to store and access elements, iterate through the elements of an array and find a specific element

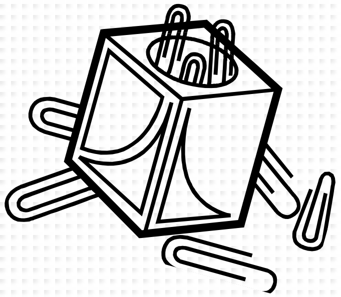
Reading Assignments

- Introduction to Java Programming (required)
  - Chapter 6: Single-Dimensional Arrays
    - Section 6.1, 6.2 (not including 6.2.7), 6.5 – 6.8
    - Section 6.12 (the Arrays class)
- Head First Java
  - Chapter 3. Primitives and References: Know Your Variables
    - Section 3.8. An array is like a tray of cups
    - Section 3.9. Arrays are objects too
- Java Arrays Tutorial
  - http://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

Exercise 1: Pass the clips

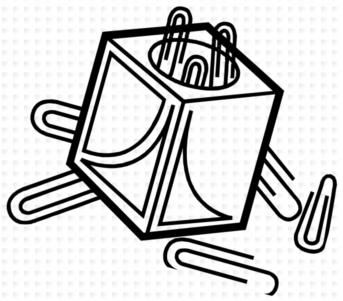
- In groups of 3 or 4, consider each student in the group as being an object of a Java program
- One of the group members will be the “main” object which chooses how many paperclips to “create” and use.
  - Start with 1 clip and progressively increase the number
  - Pass the group of paperclips to another “object” (person) one at a time, holding only one at a time in your hand
- The object of the exercise is to find the most efficient way of passing the clips between objects, between the members of the group.

Pass the Paperclips



Exercise 1: Pass the clips

- What did you do to pass the clips more efficiently?



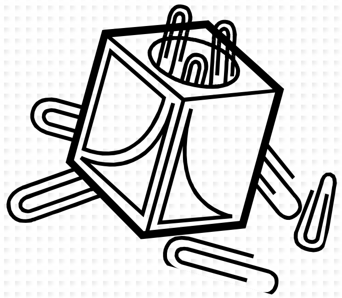
Arrays

What are arrays and why do we need them?

Hint: Define 5 int variables in your program
Hint: Define 10 int variables in your program
Hint: Define 100 int variables in your program

What is an array?

- An array is a container of related variables of the same type sharing a common name.
  - Set of variables with a common name
  - All variables in the container have the same type
  - The number of variables is fixed once established
- Think of an array as a "box of variables"
  - Think of the real world. What is the purpose of a box?



- The array has a data type, a type that represents how big the box is and what kind of elements can it hold, e.g. "array of doubles size 50"
- The array is a variable itself, the variable that represents the box
- The array is an object and therefore can have fields
  - length is a public field that all arrays have which "remembers" the number of variables in the array

The Element

- Each variable in the array is called an element of the array
  - Think of the paperclip as the element of the array
- All elements must have the same data type
  - Arrays can hold variables of any type
    - Primitive types like int and double
    - Objects like String, Person, Employee
    - Other arrays!
- All elements are part of the same box.
  - To initialize an element you must “place it” in the box.
  - To use an element you must “reach inside” the box and grab one. Once you have an element, you can read it or you can change it
- How many elements can you have in an array? As many as you want. But once the box is created it won’t grow.

All elements inside an array have the same type

The number of elements in an array cannot change!

Defining an Array Type (pseudocode)

- When defining an array type you must specify the type of elements the array will hold

```
<type-of-element>[]
```

What kind of elements can the box hold?

Square brackets indicate you are working with an array

Creating and Initializing an Array (pseudocode)

- To create and initialize an array you must define a variable of an array type and you must initialize it
  - An array is an object so it must be created using new
  - When initializing an array the number of elements must be specified. How big is the box?

```
<type-of-element>[] arrObj = new <type-of-element>[<num>];
```

Array type, read as "array of <type-of-element>"

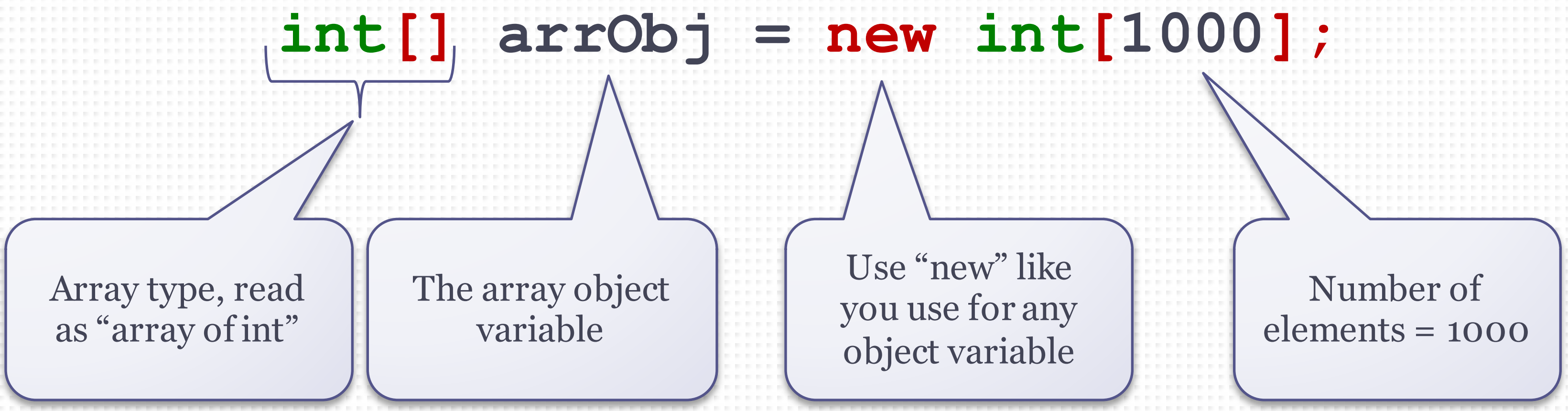
The array object variable

Use "new" like you use for any object variable

How many elements? How big is the box?

Creating and Initializing an Array (real Java)

- To create and initialize an array you must define a variable of an array type and you must initialize it
  - An array is an object so it must be created using new
  - When initializing an array the number of elements must be specified. How big is the box?



```
int[] arrObj = new int[1000];
```

Array type, read as "array of int"
The array object variable
Use "new" like you use for any object variable
Number of elements = 1000

[illegible]

[illegible] create [illegible]

[illegible] new

[illegible]

Examples: Creating Arrays

-
  ```
  // the following array variable is declared but not initialized
  int[] nums;
  nums= new int[50];  // now it is initialized
  ```

-
  ```
  String[] names = new  String[100];
  ```

-
  ```
  // read the number of elements from the user and create an array of
  // that size
  int elemNum = input.nextInt();
  int[] values = new  int[elemNum];
  ```

[illegible]

[illegible]

[illegible] empty [illegible]

You have to put

[illegible]

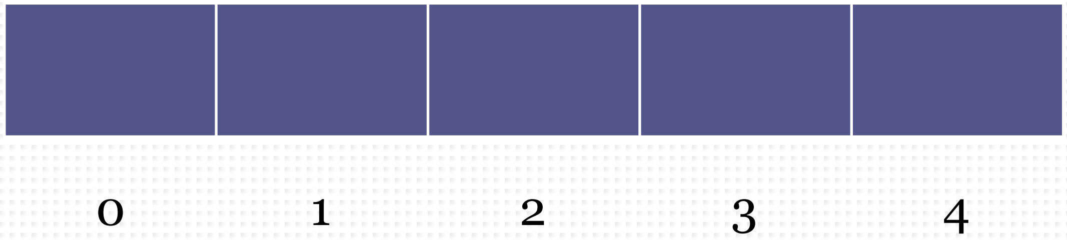
in the box

Accessing an Element

- To access an element, use the array variable followed by an index number in square brackets
  - The index is the location of the element you are trying to access

arrVar[<index>]

- The array indices start at zero and end at "array length – 1"

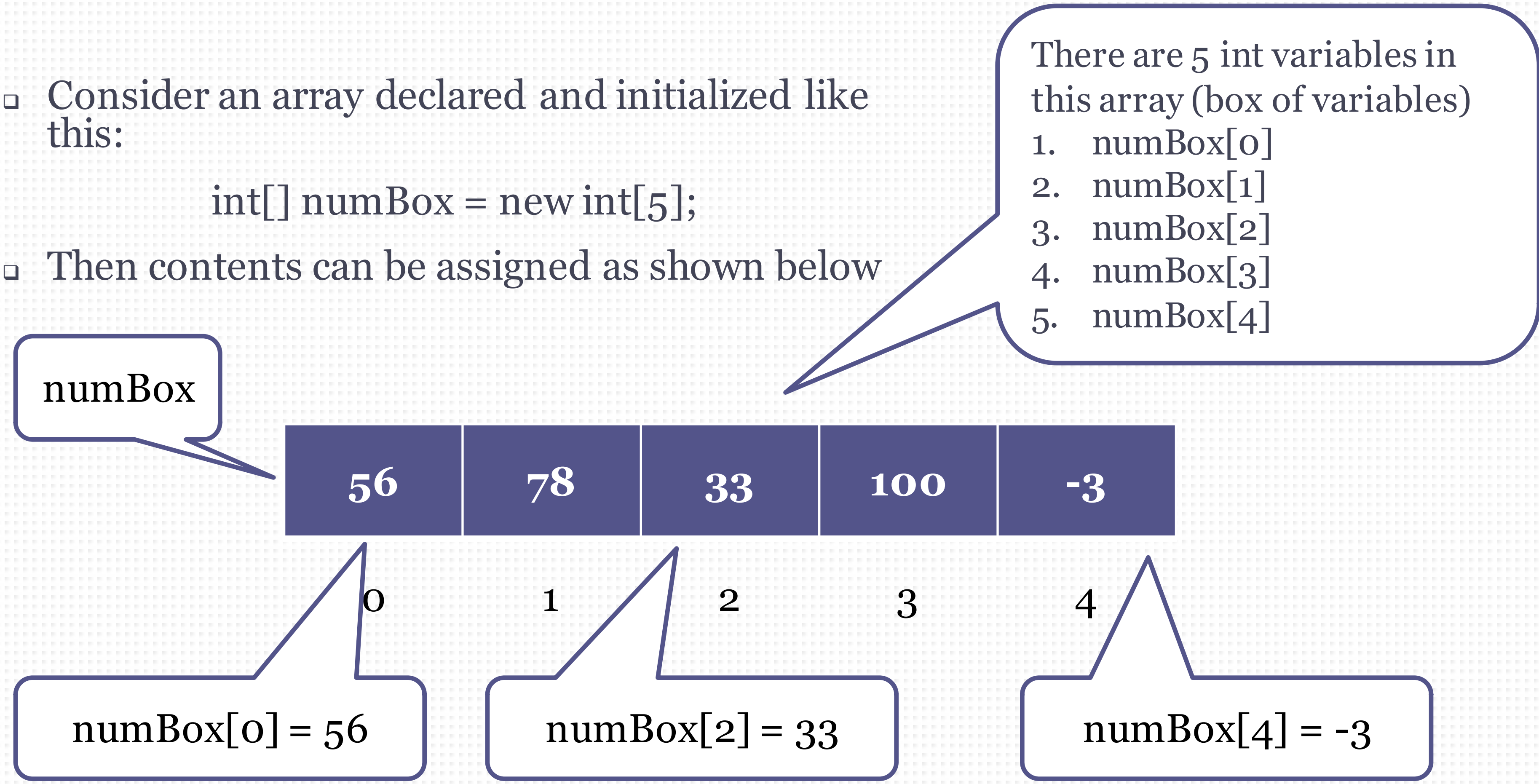


Example: Array of Integers

- Consider an array declared and initialized like this:

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

- Then contents can be assigned as shown below



[illegible]
[illegible] array of ____

[illegible]
[illegible] accessing
one element [illegible]

Iterating through an array

- Iterating through an array means "visiting" each element to process it for a given purpose (e.g. print, change value, use in a calculation)
- If 'arr' is an array object variable, then arr.length gives its size
- The index number in square brackets is always an int, so use an int variable for your loop

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

When iterating through an array,

use a **for** loop.

Start at **zero**,

use **less than length**

as the condition,

and **increment** by one

Exercise 2: Declaring Arrays

- Create a Java Program called Arrays101 and implement the following functionality
- Create an array of integers named “numbers” that can hold 5 integers
  1. Declare and create the array variable (using ‘new’)
  2. Assign the following values in order: 56, 78, 33, 100, -3
- Print the value of the array variable
  - What “value” do you get?
- Print the value of each array element on a separate line
- Print the size of the array using the “length” field of the array
  - This is the number of the elements in the array

Exercise 3: Arrays and Loops

- Did you use a loop in the previous exercise? If not...
- Use a loop to print the contents of the array
  - What is the starting point for the loop counter?
  - What is the condition on the upper limit?
  - What is the increment?
- Use a loop to print the contents of the array in reverse order
  - What is the starting point for the counter?
  - What is the condition on the lower limit?
  - What is the increment?

Exercise 4: Changing Elements

Modify the previous exercise to initialize the array using a loop:

- Step 1: Use a loop to change the content of the array so that it holds powers of 2: 2^0, 2^1, 2^2, 2^3, 2^4
- Step 2: Change the array to hold the powers of 2 from 0 to 49? What happens? What element data type do you need?
- Step 3: Instead of powers of 2, fill the array with random integers from 1 to 100

Exercise 5 - Searching: Find a number

- Continue the previous exercise…
  - Ask the user for a number
  - Check whether the number exists in the 'numbers' array that was randomly filled

- If the number is found, print out where it was found (which element number)
- Also print out if the number was not found

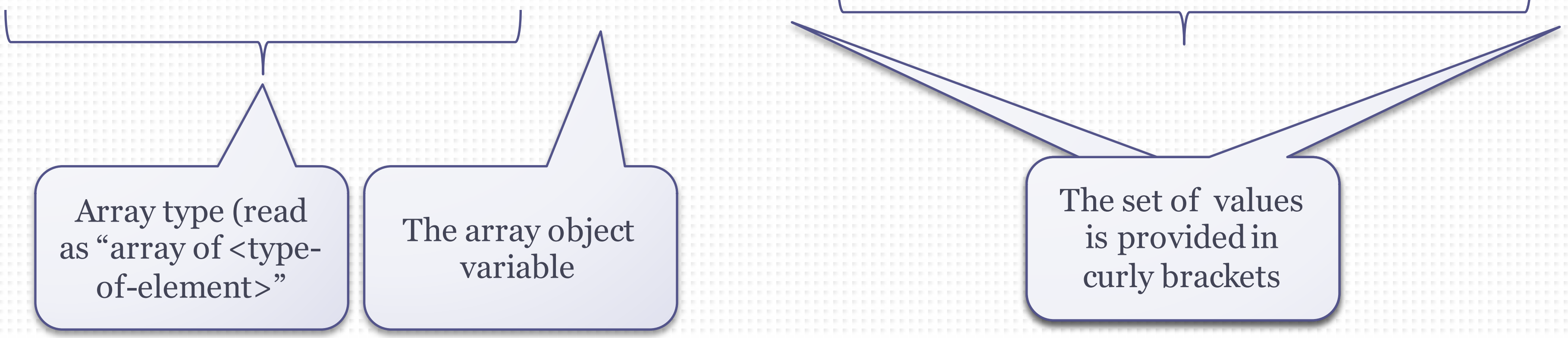
Recommended Exercise

- Complete Exercise 3 with arrays of different types:
  - Array of float or double
  - Array of boolean variables
  - Array of characters (char)
  - Array of Strings

When you know the elements

- Arrays can be initialized with one statement when you know the exact elements you need to place in the array
  - This happens fairly rarely in practice
  - It's more useful for simple exercises
- If you know the exact elements you can declare and initialize an array using the following statement

```
<type-of-element>[] arrObj = { <val_1>, <val_2>, … ,<val_n> }
```


Array type (read as “array of <type-of-element>”
The array object variable
The set of values is provided in curly brackets

Examples: Array Initialization

- int[] grades = {9, 10, 8, 6, 9};

- boolean[] flags = {true, true, false, true, false};

- String[] days = {“Mon”, “Tue”, “Wed”, “Thu”, “Fri”, “Sat”, “Sun”};

- Player[] playerList = { new Player(“Larry”), new Player(“Curly”),
  new Player(“Moe”) };

Example: Accessing elements

- String[] nameList = {“John”, “Mary”, “George”}
- nameList[0] → represents “John”
- nameList[1] → represents “Mary”
- nameList[2] → represents “George”
- Access all elements by iterating through the array

```
 String[] nameList = {"John", "Mary", "George"};
for(int index = 0; index < nameList.length; index++)
{
   System.out.println("nameList[" + index + "] = " + nameList[index]);
}
```

Exercise 6 – Easy num-to-string conversion

- Write a method called numToMonth that converts a number from 1 to 12 to the corresponding month name *January, February, March, … December*
  - In the past we might have done this with switch
  - Use an array of strings with an initializer instead
  - Remember: Array indexes start at 0, you'll have to adjust for the fact that our month numbers start at 1
- Write a main method to test your numToMonth method

Array and Array Element Data Types

- Suppose we create this array:

  char[] letters = { ‘A’, ‘B’, ‘C’, ‘D’ };

- What is the data type of letters?
  - char[] or “array of char”
- What is the data type of letters[2]?
  - Its data type is char
  - This is one element of the array, one of the things in the container
- What is the value of letters[2]?
  - ‘C’

Array and Array Element Data Types (cont'd)

- Suppose int n = 1, what is the data type of letters[n]?
  - Also char, all elements of the array letters are type char
  - The value of 'n' doesn't matter
- We can use a local variable to work with an element of the array,
  - char ch = letters[n];
  - … do something with 'ch', change its value etc.
  - letters[n] = ch;
- But we don't have to, we can work directly with letters[n]

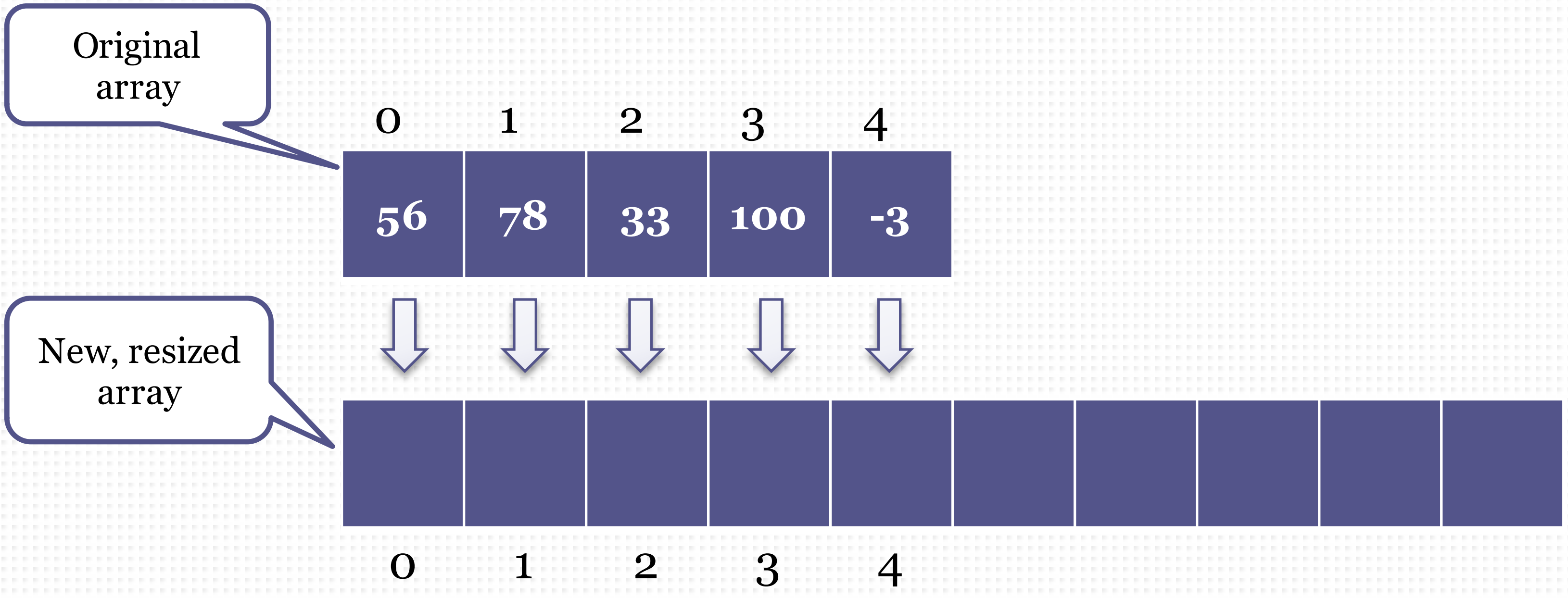
Arrays as field variables

- So far we've been making our arrays local variables
- Often we store information in an array that's a field
  - This is a good way for a class to store and manage data

The Arrays utility class

Resizing an array

- Arrays have a fixed, unchangeable length.
- To resize, a new array must be created and elements must be copied from the original to the resized array



The ‘Arrays’ Class from Java Library

- The Arrays class contains a bunch of useful helper methods for working with arrays
  - Copy / resize arrays
  - Sorting: put the array contents in alphabetical or numeric order
  - Searching: search a sorted array for a specific item
  - Convert to string: can be used to print out array contents
  - Compare arrays: are all elements the same?
  - Fill many array elements with the same value
- You’ll need to import java.util.Arrays
- Many overloads for diff. types, see javadoc.org for details

The ‘Arrays’ Class from Java Library

- Some of the useful methods:
  - static String toString(int[] a)   // Convert int array to String
  - static String toString(double[] a)   // Same but for doubles
  - static int[] copyOf(int[] original, int newLength)
    // Copy & resize array of ints
  - static void sort(int[] a)   // sort into ascending num. order
  - static void sort(Object[] a)   // sort *objects* like strings
  - static int binarySearch(int[] a, int key)
    // efficiently search a sorted list for ‘key’
  - static boolean equals(int[] a, int[] a2)   // compare arrays

- Note all these methods are static, you don’t need to make an Arrays object! (In fact you can’t)

Exercise 8: Using the Arrays class

- A: Starting with the program from Exercise 3,
  - Change the print statement which prints the entire array to use Arrays.toString()
  - Use Arrays.sort() to sort the array and print it again
  - Use Arrays.copyOf() to resize the array, then print the resized copy using Arrays.toString()

- B: Continuing with this program,
  - Prompt the user to input a number
  - Use binarySearch() to find the number in the sorted array (displaying the number's index if found)

A Note on Copying Arrays

- When you need to copy an array you should not do this:

  ```
  int[] newArray = oldArray;
  ```

- If you assign one array variable to another this gives them the same reference or "pointer" to your array data
  - The data is shared, and any changes to one array will affect the other array!
  - *Exercise:* Try it and see, starting with the program from Exercise 3: Make a "copy" of the array by assigning variables, change contents of the copy, then print both the copy & the original using Arrays.toString()
  - Use this instead:

    ```
    int[] newArray = Arrays.copyOf(oldArray, oldArray.length);
    ```

A Note on Copying Arrays (cont'd)

- The same thing happens when you pass an array as a parameter when calling a method
  - The array is not copied, instead a reference to the array is passed to the method
  - If the method changes the array contents, the original array is changed!
  - This can be very useful and is often what you want to do when passing arrays as parameters, but watch out if not
- All of this is true for all Java objects (but *not* primitive types)
  - If you pass an object as a parameter the method can change its contents (fields)

Recommended Exercise: Array of Strings

- Create a “Horoscope” program with two classes
  - HorscopeApp responsible for asking the user for up to 10 predictions
  - Horoscope responsible for “remembering” the predictions as an array of String objects and for providing functionality of making a prediction
- HorscopeApp Class
  - This contains the main method
  - In the main method ask the user for predictions and add them to the horoscope
    - Example: “You will buy a car”, “You will catch a cold”
    - Use a loop to keep asking the user for more predictions until the user says they are done (enters an empty string). Make sure the array bounds are not exceeded
  - Show a random prediction by calling a method of the Horoscope class
- Horoscope Class
  - Define a field to store the predictions as an array of String objects
  - Define an integer field to store the actual number of predictions
  - Write a method to add a new prediction to the array
  - Write a method to show a random prediction