CS102 Programming II

REVIEW: SINGLE DIMENSION ARRAYS

Prof. Dr. Basit Qureshi
PhD FHEA SMIEEE MACM
www.drbasit.org

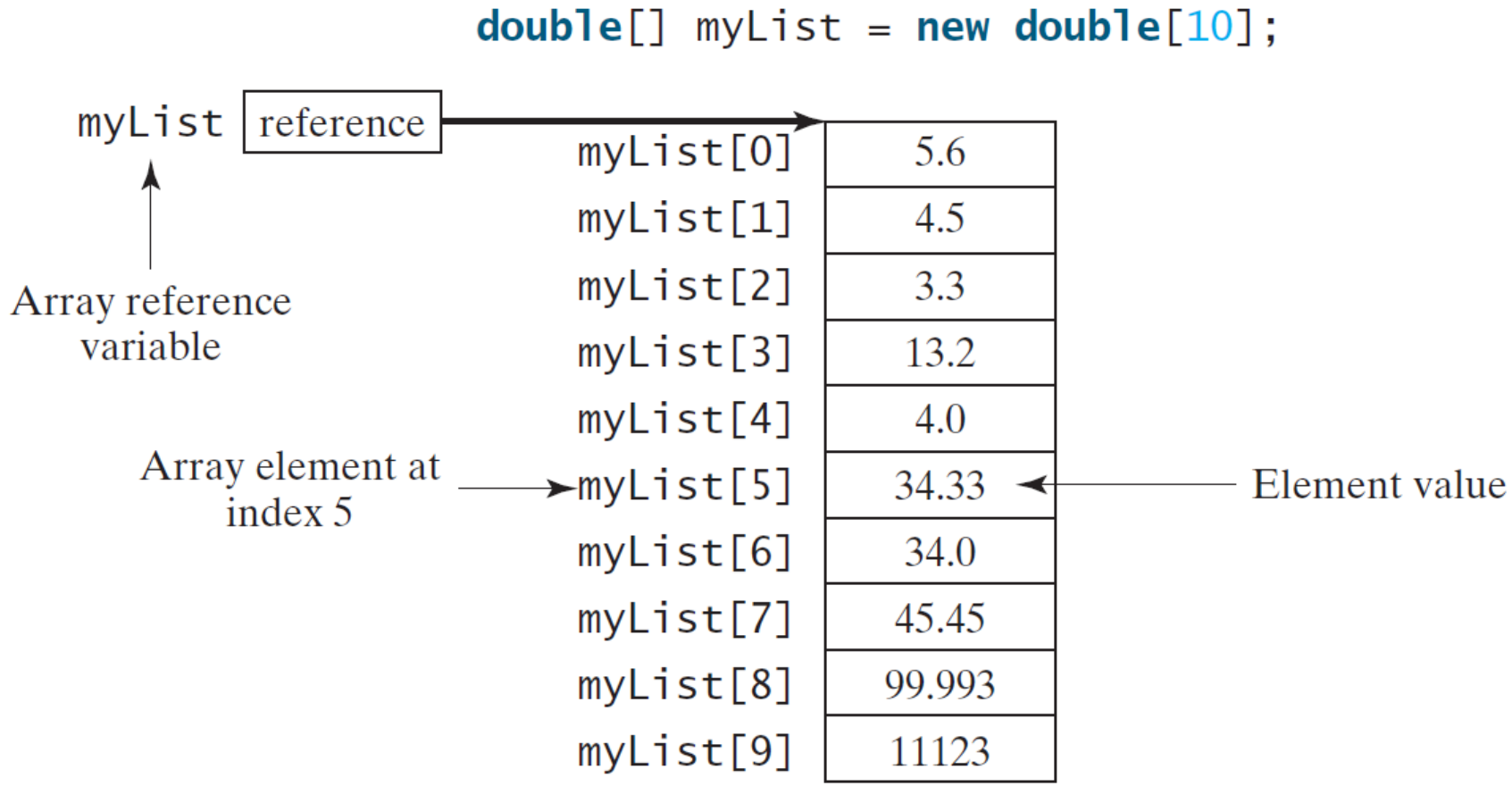
جامعة الأمير سلطان
PRINCE SULTAN UNIVERSITY

TOPICS

- Review class covering single-dimension arrays, methods, files.
- Example programs for arrays and files.

INTRODUCING ARRAYS

Array is a data structure that represents a collection of the same types of data.



DECLARING ARRAY VARIABLES

- `datatype[] arrayRefVar;`

  Example:

  ```
  double[] myList;
  ```

- `datatype arrayRefVar[];` // This style is allowed, but not preferred

  Example:

  ```
  double myList[];
  ```

CREATING ARRAYS

```
arrayRefVar = new datatype[arraySize];
```

Example:

```
myList = new double[10];
```

`myList[0]` references the first element in the array.

`myList[9]` references the last element in the array.

DECLARING AND CREATING IN ONE STEP

- `datatype[] arrayRefVar = new datatype[arraySize];`

```
double[] myList = new double[10];
```

- `datatype arrayRefVar[] = new datatype[arraySize];`

```
double myList[] = new double[10];
```

THE LENGTH OF AN ARRAY

Once an array is created, its size is fixed. It cannot be changed. You can find its size using

```
arrayRefVar.length
```

For example,

```
myList.length returns 10
```

USING INDEXED VARIABLES

After an array is created, an indexed variable can be used in the same way as a regular variable. For example, the following code adds the value in myList[0] and myList[1] to myList[2].

```
myList[2] = myList[0] + myList[1];
```

SHORTHAND NOTATION

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```

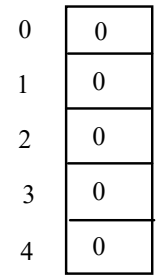
This shorthand notation is equivalent to the following statements:

```
double[] myList = new double[4];

myList[0] = 1.9;

myList[1] = 2.9;

myList[2] = 3.4;

myList[3] = 3.5;
```

TRACE PROGRAM WITH ARRAYS

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the array is created



CS102 Programming II

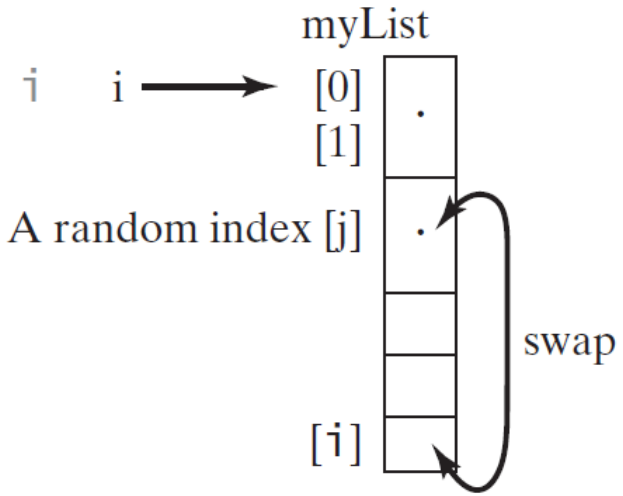
EXAMPLES: SINGLE DIMENSION ARRAYS

Prof. Dr. Basit Qureshi
PhD FHEA SMIEEE MACM
www.drbasit.org

جامعة الأمير سلطان
PRINCE SULTAN UNIVERSITY

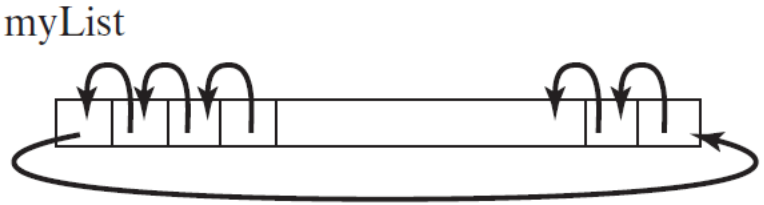
RANDOM SHUFFLING

```
for (int i = myList.length - 1; i > 0; i--) {
  // Generate an index j randomly with 0 <= j <= i
  int j = (int)(Math.random()
    * (i + 1));

  // Swap myList[i] with myList[j]
  double temp = myList[i];
  myList[i] = myList[j];
  myList[j] = temp;
}
```



SHIFTING ELEMENTS

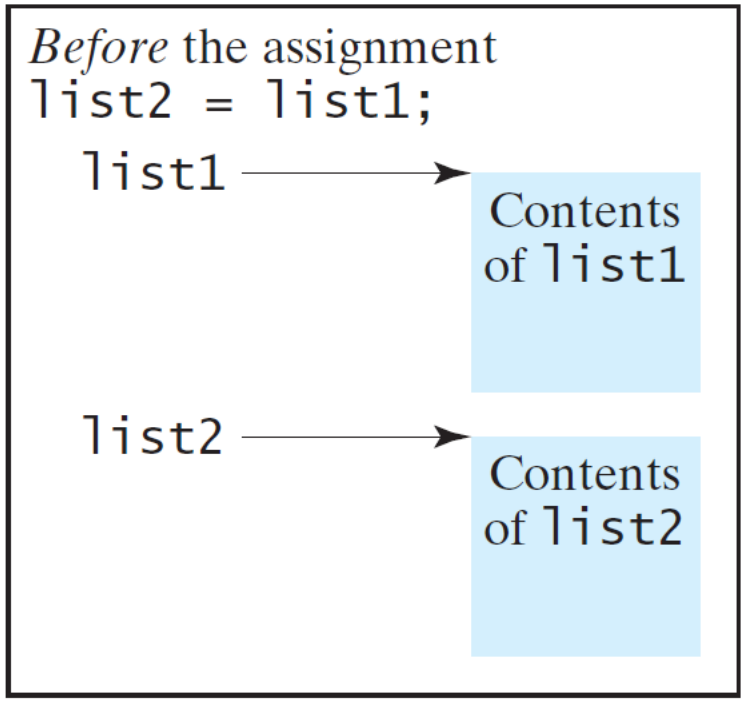
```
double temp = myList[0]; // Retain the first element

// Shift elements left
for (int i = 1; i < myList.length; i++) {
  myList[i - 1] = myList[i];
}

// Move the first element to fill in the last position
myList[myList.length - 1] = temp;
```

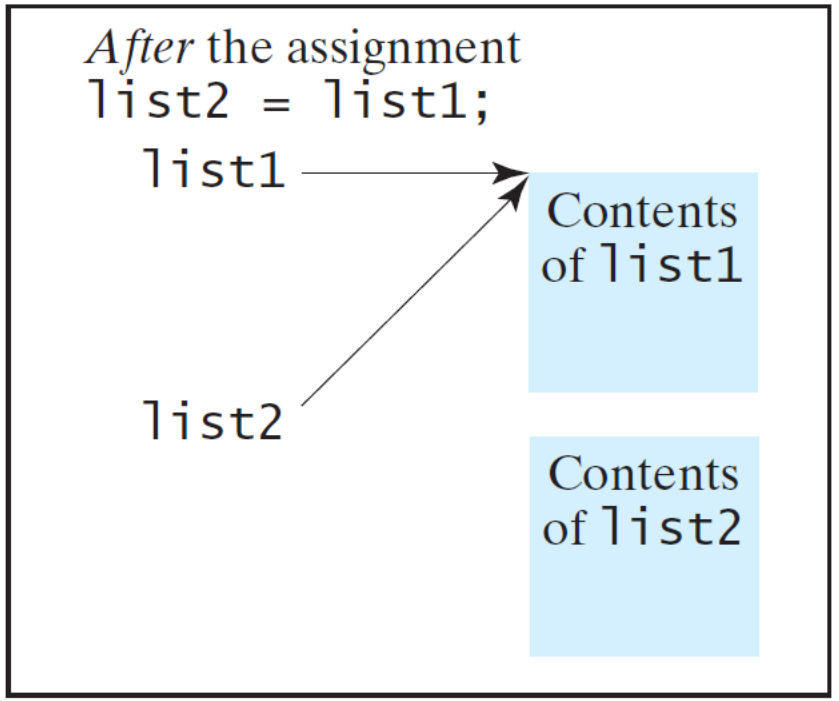


COPYING ARRAYS

Often, in a program, you need to duplicate an array or a part of an array. In such cases you could attempt to use the assignment statement (=), as follows:

list2 = list1;





COPYING ARRAYS

Using a loop:

```
int[] sourceArray = {2, 3, 1, 5, 10};
int[] targetArray = new int[sourceArray.length];

for (int i = 0; i < sourceArrays.length; i++)
   targetArray[i] = sourceArray[i];
```

PASSING ARRAYS TO METHODS

```
public static void printArray(int[] array) {
  for (int i = 0; i < array.length; i++) {
    System.out.print(array[i] + " ");
  }
}
```

Invoke the method

```
int[] list = {3, 1, 2, 6, 4, 2};
printArray(list);
```

Invoke the method

```
printArray(new int[]{3, 1, 2, 6, 4, 2});
```

Anonymous array

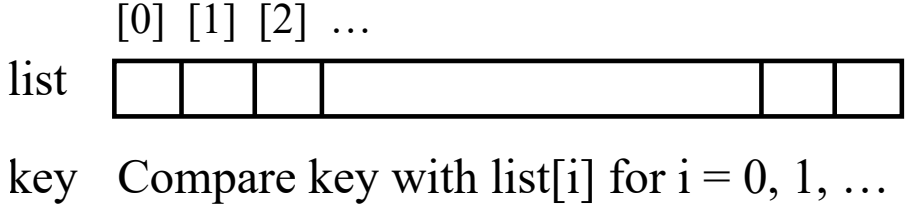
RETURNING AN ARRAY FROM A METHOD

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);

public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

SEARCHING ARRAYS

Searching is the process of looking for a specific element in an array; for example, discovering whether a certain score is included in a list of scores. Searching is a common task in computer programming. There are many algorithms and data structures devoted to searching. In this section, two commonly used approaches are discussed, *linear search* and *binary search*.

```
public class LinearSearch {
  /** The method for finding a key in the list */
  public static int linearSearch(int[] list, int key) {
    for (int i = 0; i < list.length; i++)
      if (key == list[i])
        return i;
    return -1;
  }
}
```



THE ARRAYS.SORT METHOD

Since sorting is frequently used in programming, Java provides several overloaded sort methods for sorting an array of int, double, char, short, long, and float in the java.util.Arrays class. For example, the following code sorts an array of numbers and an array of characters.

```
double[] numbers = {6.0, 4.4, 1.9, 2.9, 3.4, 3.5};
java.util.Arrays.sort(numbers);

char[] chars = {'a', 'A', '4', 'F', 'D', 'P'};
java.util.Arrays.sort(chars);
```

Java 8 now provides Arrays.parallelSort(list) that utilizes the multicore for fast sorting.

COMMAND-LINE PARAMETERS

```
class TestMain {
  public static void main(String[] args) {
  ...
  }
}

java TestMain arg0 arg1 arg2 ... argn
```

CS102 Programming II

REVIEW: FILES

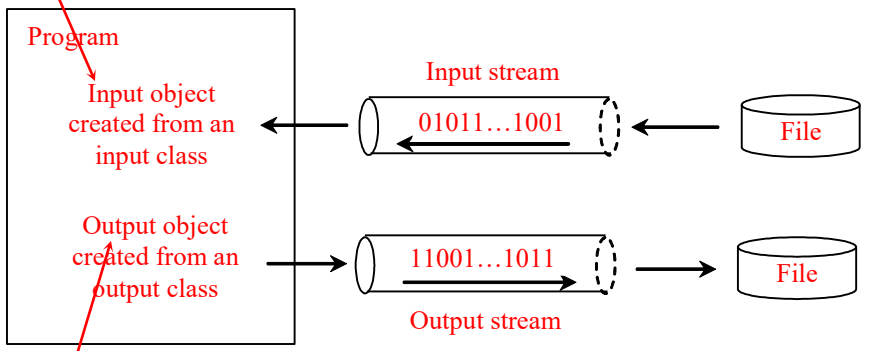
Prof. Dr. Basit Qureshi
PhD FHEA SMIEEE MACM
www.drbasit.org

جامعة الأمير سلطان
PRINCE SULTAN UNIVERSITY

HOW IS I/O HANDLED IN JAVA?

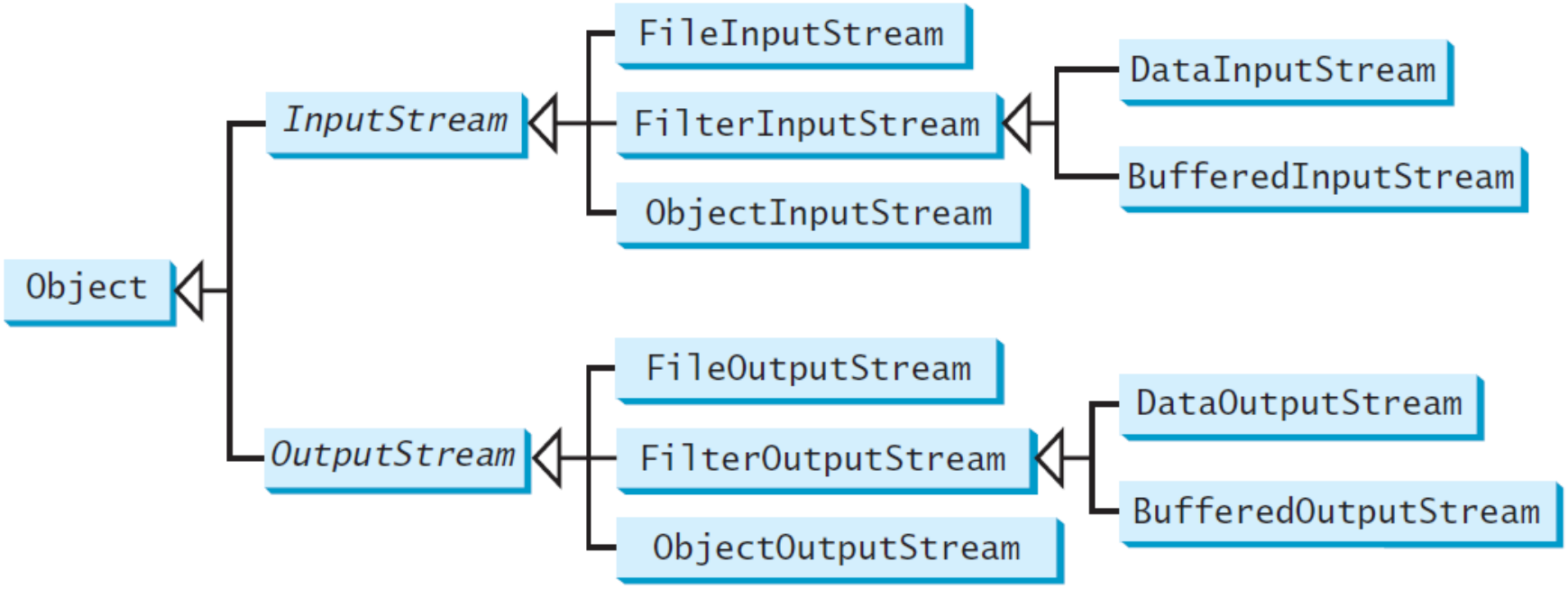
A File object encapsulates the properties of a file or a path, but does not contain the methods for reading/writing data from/to a file. In order to perform I/O, you need to create objects using appropriate Java I/O classes.

```
Scanner input = new Scanner(new File("temp.txt"));
System.out.println(input.nextLine());
```




```
PrintWriter output = new PrintWriter("temp.txt");
output.println("Java 101");
output.close();
```

BINARY I/O CLASSES



FILEINPUTSTREAM

To construct a FileInputStream, use the following constructors:

public FileInputStream(String filename)

public FileInputStream(File file)

A java.io.FileNotFoundException would occur if you attempt to create a FileInputStream with a nonexistent file.

FILEOUTPUTSTREAM

To construct a FileOutputStream, use the following constructors:

```
public FileOutputStream(String filename)
public FileOutputStream(File file)
public FileOutputStream(String filename, boolean append)
public FileOutputStream(File file, boolean append)
```

If the file does not exist, a new file would be created. If the file already exists, the first two constructors would delete the current contents in the file. To retain the current content and append new data into the file, use the last two constructors by passing true to the append parameter.

CREATING / WRITING TO FILE

```
public class Files1 {
    public static void main(String [] s)
    {
        try (
          // Create an output stream to the file
          FileOutputStream output = new FileOutputStream("temp.txt");
            )
        {
          // Output values to the file
          for (int i = 1; i <= 10; i++)
            output.write(i);
        }
        catch (IOException e) {
          System.out.println("Error writing file.");
          e.printStackTrace();
        }

  }
}
```

CREATING / WRITING TO FILE

```
public class Files2 {
    public static void main(String [] s)
    {
        try (
          // Create an input stream for the file
          FileInputStream input = new FileInputStream("temp.txt");
        )
        {
          // Read values from the file
          int value;
          while ((value = input.read()) != -1)
            System.out.print(value + " ");
        }
        catch (IOException e) {
          System.out.println("Error reading file.");
          e.printStackTrace();
        }
  }
}
```

SUMMARY

- Single dimension arrays in Java
- Java examples
- Files