School of Computing

CS1020 Data Structures and Algorithms I
Lecture Note #2

Arrays

Objectives

Using arrays to organise data.

References

Book

- **Array:** Chapter 1, Section 1.1, pages 35 to 38

CS1020 website → Resources → Lectures

- http://www.comp.nus.edu.sg/~cs1020/2_resources/lectures.html

Outline

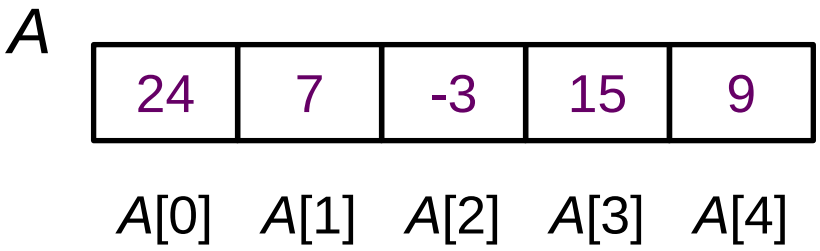
1. Array
   - 1.1 Introduction
   - 1.2 Array in C
   - 1.3 Array in Java
   - 1.4 Array as a Parameter
   - 1.5 Detour: String[] in main() method
   - 1.6 Returning an Array
   - 1.7 Common Mistakes
   - 1.8 2D Array
   - 1.9 Drawback

1 Array

A collection of homogeneous data

Introduction

- Array is the simplest way to store a collection of data of the same type (homogeneous)
- It stores its elements in contiguous memory
  - Array index begins from zero
  - Example of a 5-element integer array *A* with elements filled



Array in C (1/2)

```c
#include <stdio.h>
#define MAX 6

int scanArray(double [], int);
void printArray(double [], int);
double sumArray(double [], int);

int main(void) {
  double list[MAX];
  int size;

  size = scanArray(list, MAX);
  printArray(list, size);
  printf("Sum = %f\n",
         sumArray(list, size));

  return 0;
}
```

```c
// To read values into arr and return
// the number of elements read.
int scanArray(double arr[], int max_size) {
  int size, i;

  printf("How many elements? ");
  scanf("%d", &size);
  if (size > max_size) {
    printf("Exceeded max; you may only enter");
    printf(" %d values.\n", max_size);
    size = max_size;
  }
  printf("Enter %d values: ", size);
  for (i=0; i<size; i++) {
    scanf("%lf", &arr[i]);
  }
  return size;
}
```

sum_array.c

Array in C (2/2)

sum_array.c

```
// To print values of arr
void printArray(double arr[], int size) {
  int i;

  for (i=0; i<size; i++)
    printf("%f ", arr[i]);
  printf("\n");
}

// To compute sum of all elements in arr
double sumArray(double arr[], int size) {
  int i;
  double sum = 0.0;

  for (i=0; i<size; i++)
    sum += arr[i];
  return sum;
}
```

Array in Java (1/2)

- In Java, **array is an object**.
- Every array has a **public** length attribute (it is not a method!)

TestArray1.java

```
public class TestArray1 {

  public static void main(String[] args) {
    int[] arr; // arr is a reference

    // create a new integer array with 3 elements
    // arr now refers (points) to this new array
    arr = new int[3];

    // using the length attribute
    System.out.println("Length = " + arr.length);

    arr[0] = 100;
    arr[1] = arr[0] - 37;
    arr[2] = arr[1] / 2;
    for (int i=0; i<arr.length; i++)
      System.out.println("arr[" + i + "] = " + arr[i]);
  }
}
```

Declaring an array:
`datatype[] array_name`

Constructing an array:
`array_name = new datatype[size]`

Accessing individual array elements.

```
Length = ?
arr[0] = ?
arr[1] = ?
arr[2] = ?
```

Array in Java (2/2)

- Alternative loop syntax for accessing array elements
- Illustrate toString() method in Arrays class to print an array

TestArray2.java

```
public class TestArray2 {
   public static void main(String[] args) {
      // Construct and initialise array
      double[] arr = { 35.1, 21, 57.7, 18.3 };

      // using the length attribute
      System.out.println("Length = " + arr.length);

      for (int i=0; i<arr.length; i++) {
         System.out.print(arr[i] + " ");
      }
      System.out.println();

      // Alternative way
      for (double element: arr) {
         System.out.print(element + " ");
      }
      System.out.println();
      System.out.println(Arrays.toString(arr));
   }
}
```

```
Length = 4
35.1 21.0 57.7 18.3
35.1 21.0 57.7 18.3
[35.1, 21.0, 57.7, 18.3]
```

Syntax (enhanced for-loop):
`for (datatype e: array_name)`

Go through all elements in the array. “e” automatically refers to the array element sequentially in each iteration

Using toString() method in Arrays class

Array as a Parameter

- The reference to the array is passed into a method
  - Any modification of the elements in the method will affect the actual array

TestArray3.java

```
public class TestArray3 {
  public static void main(String[] args) {
    int[] list = { 22, 55, 33 };

    swap(list, 0, 2);

    for (int element: list)
      System.out.print(element + " ");
    System.out.println();
  }

  // To swap arr[i] with arr[j]
  public static void swap(int[] arr, int i, int j) {
    int temp = arr[i]; arr[i] = arr[j]; arr[j] = temp;
  }
}
```

Detour: String[] in main() method

- The main() method contains a parameter which is an array of String objects
- We can use this for command-line arguments

TestCommandLineArgs.java

```
public class TestCommandLineArgs {
  public static void main(String[] args) {

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

```
java TestCommandLineArgs The "Harry Potter" series has 7 books.
args[0] = The
args[1] = Harry Potter
args[2] = series
args[3] = has
args[4] = 7
args[5] = books.
```

Returning an Array

- Array can be returned from a method

TestArray4.java

```
public class TestArray4 {
  public static void main(String[] args) {
    double[] values;

    values = makeArray(5, 999.0);

    for (double value: values) {
      System.out.println(value + " ");
    }
  }

  // To create an array and return it to caller
  public static double[] makeArray(int size, double limit) {
    double[] arr = new double[size];

    for (int i=0; i < arr.length; i++)
      arr[i] = limit/(i+1);

    return arr;
  }
}
```

Return type:
`datatype[]`

```
999.0
499.5
333.0
249.75
199.8
```

Common Mistakes (1/3)

- length versus length()
  - To obtain length of a **`String`** object str, we use the **length()** method
    - Example: **`str.length()`**
  - To obtain length (size) of an array arr, we use the **length** attribute
    - Example: **`arr.length`**
- Array index out of range
  - Beware of ArrayIndexOutOfBoundsException

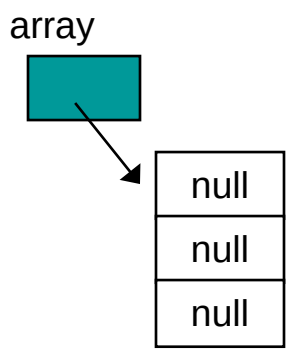
```
public static void main(String[] args) {
  int[] numbers = new int[10];
  . . .
  for (int i = 1; i <= numbers.length; i++)
    System.out.println(numbers[i]);
}
```

Common Mistakes (2/3)

- When you have an array of objects, it's very common to forget to instantiate the array's objects.
- Programmers often instantiate the array itself and then think they're done – that leads to java.lang.NullPointerException
- Example on next slide
  - It uses the **`Point`** class in the API
  - Refer to the API documentation for details

Common Mistakes (3/3)

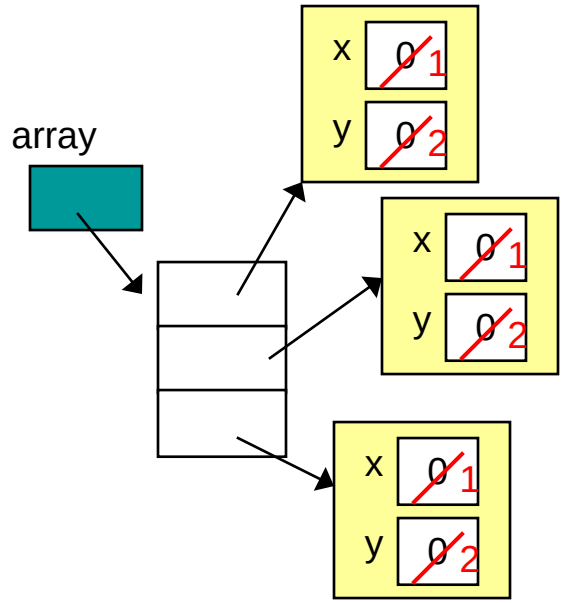
```
Point[] array = new Point[3];
for (int i=0; i<array.length; i++) {
   array[i].setLocation(1,2);
}
```



There are no objects referred to by array[0], array[1], and array[2], so how to call setLocation() on them?!

Corrected code:

```
Point[] array = new Point[3];
for (int i=0; i<array.length; i++) {
   array[i] = new Point();
   array[i].setLocation(1,2);
}
```

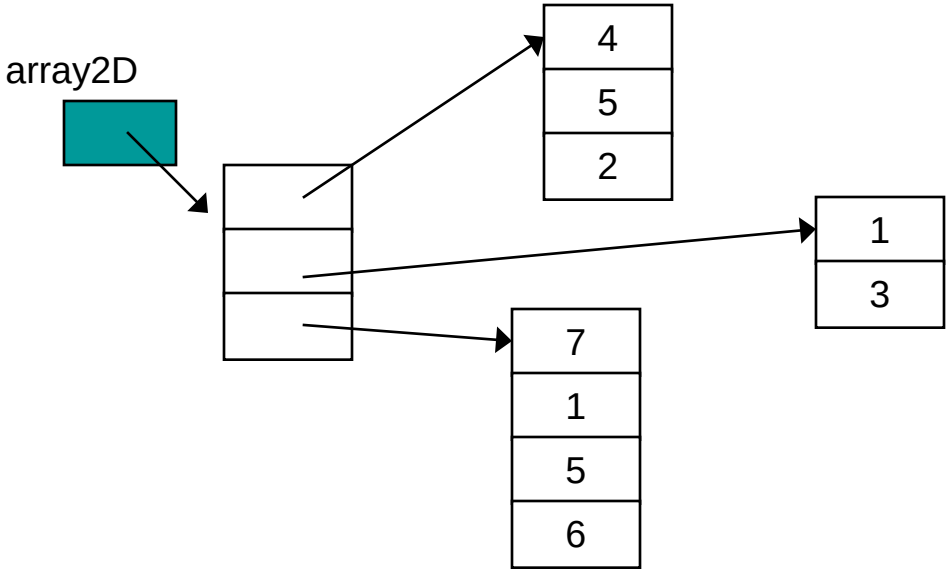


2D Array (1/2)

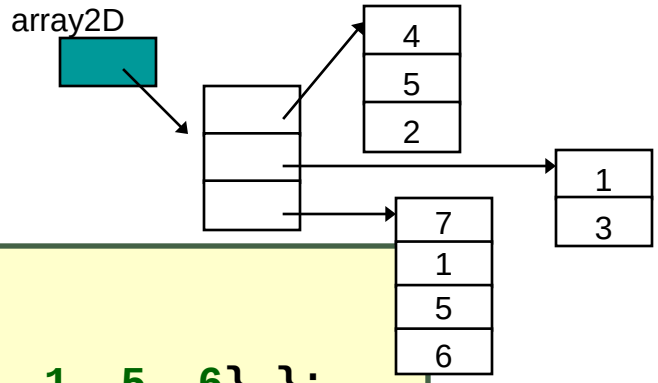
- A two-dimensional (2D) array is an array of array.
- This allows for rows of different lengths.

```
// an array of 12 arrays of int
int[][] products = new int[12][];
```

```
int[][] array2D = { {4,5,2}, {1,3},
                    {7,1,5,6} };
```



2D Array (2/2)




```
public class Test2DArray {
  public static void main(String[] args) {
    int[][] array2D = { {4, 5, 2}, {1, 3}, {7, 1, 5, 6} };

    System.out.println("array2D.length = " + array2D.length);
    for (int i = 0; i < array2D.length; i++)
      System.out.println("array2D[" + i + "].length = "
                         + array2D[i].length);

    for (int row = 0; row < array2D.length; row++) {
      for (int col = 0; col < array2D[row].length; col++)
        System.out.print(array2D[row][col] + " ");
      System.out.println();
    }
  }
}
```

Test2DArray.java

```
array2D.length = 3
array2D[0].length = ?
array2D[1].length = ?
array2D[2].length = ?
?
?
?
```

Drawback

- Array has one major drawback:
  - Once initialized, the array size is fixed
  - Reconstruction is required if the array size changes
  - To overcome such limitation, we can use some classes related to array
- Java has an **Array** class
  - Check API documentation and explore it yourself
- However, we will not be using this Array class much; we will be using some other classes such as **Vector** or **ArrayList** (to be covered later)
- Before doing Vector/ArrayList, we will introduce another concept later called **Generics**

Practice Exercises

- These practice exercises are mounted on CodeCrunch this week
  - #04: Basic Statistics
    - Computing mean and standard deviation
  - #05: Missing Digits
    - Using array of primitive type
  - #06: Row and Column Sums
    - Two-dimensional array
- The files are also available on the CS1020 website: http://www.comp.nus.edu.sg/~cs1020/4_misc/practice.html
- You are urged to work on these exercise as they are important for you to cement your basic understanding of the topics that are covered so far

Pract. Ex. #05

Missing Digits (1/2)

- *[This is adapted from a CS1010 exercise in C]*
  Write a program MissingDigits.java to read in a positive integer and list out all the digits that do not appear in the input number. (Assume input value has no leading zeroes.)
- You are to use primitive array, not Vector, ArrayList or any other related classes.
- You should use a boolean array.
- Sample run:

```
Enter a number: 73015
Missing digits in 73015: 2 4 6 8 9
```

Missing Digits (2/2)

- What is the boolean array for? Idea?

End of file