initialize
① directly give value
② use new

SWEN20003
Object Oriented Software Development

Arrays and Strings

Shanika Karunasekera
karus@unimelb.edu.au

University of Melbourne

The Road So Far

- OOP Foundations
  - A Quick Tour of Java
  - Classes and Objects

Lecture Objectives

After this lecture you will be able to:

- Understand how to use Arrays
- Understand how to use Strings

Arrays

Motivation

- Store a single integer value

```
int x;
```

- Store two integer values

```
int x1, x2;
```

- Store n integer values

```
int[] intArray;
```

Keyword

Array: A sequence of elements *of the same type* arranged in order in memory

Array Declaration

```
basetype[] varName; OR \\
basetype varName[];
```

- **Declares** an array (`[]`)
- Each *element* is of type `basetype`

```
int[] intArray;
```

How many elements does this array have?

Pitfall: Array Declaration

declare : create a reference but point to nothing

```
int[] intArray;
int x = intArray[0];
```

```
Program.java:13: error: variable intArray might not have been initialized
```

- Arrays must be initialised, just like any other variable
- Let's look at how

Array Initialization and Assignment

```
int[] intArray_1 = {0, 1, 2, 3, 4};
```

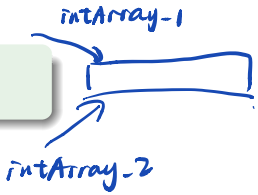
- How many elements?
- What are their values?

```
int[] intArray_2 = new int[100];
```

- How many elements?
- What are their values? all 0

```
int[] intArray_1 = new int[n];
int[] intArray_2 = intArray_1;
```

make the reference equal



- How many elements?
- What are their values?

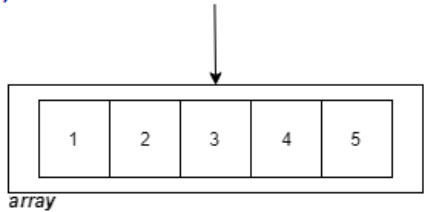
Assess Yourself

```
int[] intArray_1 = {10, 20, 30, 40};
int[] intArray_2 = intArray_1;

System.out.println(intArray_2[0]);

intArray_1[0] = 15;

System.out.println(intArray_2[0]);
```

Program Output:
10
15

Pitfall: Array Assignment

```
int[] array = {
    1, 2, 3, 4, 5
};
```



- Array is a data type, similar to data types you create by defining
- Arrays are *references*!
- Manipulating one reference affects all references

Assess Yourself

Write a Java static method, `computeDoublePowers`, that accepts an integer `n`, and returns an array of `double`s of that size. Your method should then fill that array with increasing powers of two (starting from 1.0).

```
public static double[] computeDoublePowers (int n) {
    double[] array = new double[n];
    for (int i=0; i<n; i++) {
        array[i] = Math.pow(2,i);
    }
    return array;
```

→ local variable, need to be returned

Assess Yourself

if it's return, the array won't be destroyed
memery allocate.
use a new reference to point to it

```java
public static double[] computeDoublePowers(int n) {
    double[] nums = new double[n];

    for (int i = 0; i < n; i++) {
        nums[i] = Math.pow(2, i);
    }

    // For sanity checking
    for (int i = 0; i < n; i++) {
        System.out.println(nums[i]);
    }

    return nums;
}
```

Multi-Dimensional Arrays

- Java permits "multi-dimensional" arrays
- Technically exist as "array of arrays"
- Declared just like 1D arrays

```
int[][] nums = new int[10][10]; // Square array
int[][] nums = new int[10][];   // Irregular array
```

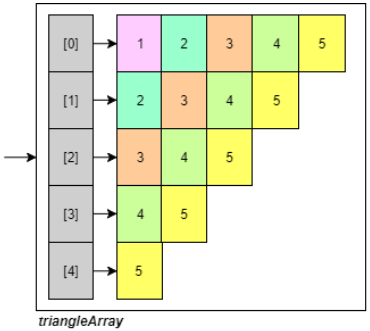
- Initialising 2D arrays slightly more complicated

```
for (int i = 0; i < nums.length; i++) {
    nums[i] = new int[<length_of_subarray>];
}
```

Assess Yourself

Write a program that can generate the following 2D array:

```
int[][] triangleArray = {
    {1, 2, 3, 4, 5},
    {2, 3, 4, 5},
    {3, 4, 5},
    {4, 5},
    {5},
};
```



triangleArray

Can you write your program with as **few assumptions as possible**?

Assess Yourself

```
public class Main {
    public static void main(String[] args) {
        int HEIGHT = 5;
        int MAX_WIDTH = HEIGHT;

        int[][] triangleArray= new int[HEIGHT][];

        for (int i = 0; i < HEIGHT; i++) {
            triangleArray[i] = new int[HEIGHT - i];

            for (int j = 0; j < HEIGHT - i; j++) {
                triangleArray[i][j] = i + j + 1;
            }
        }
    }
}
```

Arrays of Objects

Arrays can be used to store objects.

Follow the steps below to create and store objects.

- Declaration of the array:

```
Circle[] circleArray;
```

- Allocation of Storage:

```
circleArray = new Circle[25];
```

  - The above statement created an array that can store references to 25 `Circle` objects.
  - Circle objects are not created, you have to create them and store them in the array - see next example.

Arrays of Objects - Example

```
// CircleArray.java
class CircleArray{
    public static void main (String[] args){
        //declare an array for Circles
        Circle[] circleArray = new Circle[3];
        // create circle objects and store in array
        for ( int i = 0;  i < circleArray.length; i++) {
            circleArray[i] = new Circle(i,i, i + 2);
        }
        for ( int i = 0;  i < circleArray.length; i++) {
            System.out.println("Circle " + i + " Radius = " +
                circleArray[i].getR());
        }
    }
}
```

Program Output:

```
Circle 0 Radius = 2.0
Circle 1 Radius = 3.0
Circle 2 Radius = 4.0
```

Array Methods

- Indexing

```
int[] intArray = new int[10];
int x = intArray[0];
int x = intArray[10];// Gives out of bounds error
int x = intArray[-1];// Gives out of bounds error
```

> C will crash but java will give error message

- Length

```
int len = intArray.length
```

- Equality

```
import java.util.Arrays;

int[] n1 = {1, 2, 3};
int[] n2 = {1, 2, 3};

Arrays.equals(n1, n2);
//true if the element values are the same, false otherwise
```

Array Methods

- Resizing - arrays are fixed length; resizing requires creating a new array.

```
int[] intArray = new int[5];
intArray = new int[intArray.length + 3];
```

thing in old array will lost

- Sorting ("ascending")

```
Arrays.sort(n1);
```

- Printing

```
System.out.println(Arrays.toString(n1));
```

Output:

```
[1, 2, 3]
```

- Full Array documentation here

For Each Loop

```
for (<type> varName : <iterable object>) {
    <block of code to execute>
}
```

- More convenient method of iteration
- No indexing required
- Useful when operating with/on the *data*, and not the array

each next Circle

```
for (Circle c : circleArray) {
    System.out.println(c.getRadius());
}
```

Strings

Strings

You have already seen `Strings` in use:

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

string constant

```
final  String STRING_CONSTANT = "Welcome to Java";
```

```
public String toString() {....}
```

```
System.out.println("arg[" + i + "]: "    + args[i]);
```

But what is a `String`?

Assess Yourself

A "String" is a(n) what?

1. Object (not technically correct) → generate from Class
2. Class ✓
3. Variable
4. Data Type ✓
5. Method
6. Privacy Modifier
7. I have literally no clue

Strings

- Strings store sequences of characters
- `String` is a Java class
- Used to represent messages, errors, and "character" related attributes like `name`
- Incredibly powerful for input and output

Keyword

String: A Java class made up of a sequence of characters.

Strings

Some examples of String variables

```
String s1 = "This is a String";
String s2 = "This is " + "also a String";
String s3 = "10";
String s4 = "s3 is still a string, even though it's a number";
```

Java Strings are almost identical to Python, except you **can't use single quotes**.

Assess Yourself

What does this code output?

```
System.out.println("Game of Thrones season 8 was "good".");
```

1. “Game of Thrones season 8 was “good”.”
2. Game of Thrones season 8 was “good”.
3. Game of Thrones season 8 was good.
4. Error ✓

Special Characters *escape character*

- Some characters (like ") are "reserved"
- Mean something special to Java
- Need to "escape" them with "\" to use alternate meaning
- Examples "\n" (newline), "\t" (tab) "\"" (quotation)

```
System.out.println("Game of Thrones season 8 was \"good\".");
```

Keyword

Escaping: To include "special" characters in a string, use "\" to *escape* from that character's normal meaning.

String Operations

- You can use + (and +=) to append/concatenate two strings
  - `System.out.println("Hello " + "World");`
  - Prints `"Hello World"`
- + is clever: if either operand is a string, it will turn the other into a string
  - `System.out.println("a = " + a + ", b = " + b);`
  - If a = 1 and b = 2, this prints: `"a = 1, b = 2"`
- Why is this useful?

Assess Yourself

- `System.out.println("1 + 1 = " + 1 + 1);`
- Actually prints `"1 + 1 = 11"`
- `System.out.println("1 + 1 = " + (1 + 1));`
- Prints `"1 + 1 = 2"`

Assess Yourself

- Name some “logical” things you might do with a String
- Think about how you would do them in C and Python

String Methods

- Length
  - C: Need a helper/buddy variable
  - Python: `len("Hello")`
  - Java: `"Hello".length()`
- Upper/Lower case
  - C: `toupper(*s)`
  - Python: `s.upper()`
  - Java: `s.toUpperCase()`
- Split
  - C: Stop
  - Python: `s.split()`
  - Java: `s.split(" ")`

intArray.length
attribute.

object
method

not change the existing string
return a new String

String Methods

- Check substring presence
  - C: Why
  - Python: `"Hell" in s`
  - Java: `s.contains("Hell")`
- Find substring location
  - C: Never mind
  - Python: `s.find("Hell")`
  - Java: `s.indexOf("Hell")`
- Substring
  - C: I'm out
  - Python: `s[2:7]`
  - Java: `s.substring(2, 7)`

String Methods

- The full String class documentation can be found here.

Assess Yourself

What does this output?

```
String s = "Hello World";
s.toUpperCase();
s.replace("e", "i");
s.substring(0, 2);
s += " FIVE";
System.out.println(s);
```

not change s

s = s + "FIVE" change s

```
"Hello World FIVE"
```

Immutability

- `Strings` are *immutable*; once created, they can't be modified, only replaced
- This means that every `String` operation **returns** a new `String`
- We'll look at immutability in more detail soon...
- Let's fix up that code

Assess Yourself

What does this output?

```
String s = "Hello World";
s = s.toUpperCase();
s = s.replace("e", "i");
s = s.substring(0, 2);
s += " FIVE";
System.out.println(s);
```

"HE FIVE"

Assess Yourself

What does this output?

```
System.out.println("Hello" == "Hello");
```

`true`

same reference

Assess Yourself

What does this output?

```
String s = "Hello";
System.out.println(s == "Hello");
```

true

Assess Yourself

What does this output?

```
String s = "Hello";
String s2 = "Hello";
System.out.println(s == s2);
```

`true`

Assess Yourself

What does this output?

```
String s = "Hello";
String s2 = new String("Hello");
System.out.println(s == s2);
```

new reference is created

`false`

Equality

- In the previous example `s` and `s2` are references to *objects*.
- To check equality between two objects we must use the `equals` method.
  - Remember the `equals` method from our previous topic - a standard method every class should have

```
String s = "Hello";
String s2 = new String("Hello");
System.out.println(s.equals(s2));
```

```
true
```

Keyword

.equals: A method used to check two *objects* for equality

Lecture Objectives

Upon completion of this topic you will be able to:

- Use Arrays
- Use Strings