


# Computer Language



Reference Type

# Agenda

---

## ■ Reference Type

- Basics

- Array

# Basics

Array

# Reference Type

## ■ Data type

### ➤ Primitive type

- Integer (byte, char, short, int, long)
- Floating-point (float, double)
- Boolean

### ➤ Reference type

- Array
- Enum
- Class (+String, Wrapper)
- Interface

# Reference Type (cont'd)

## ■ Data type

### ➤ Primitive type variable

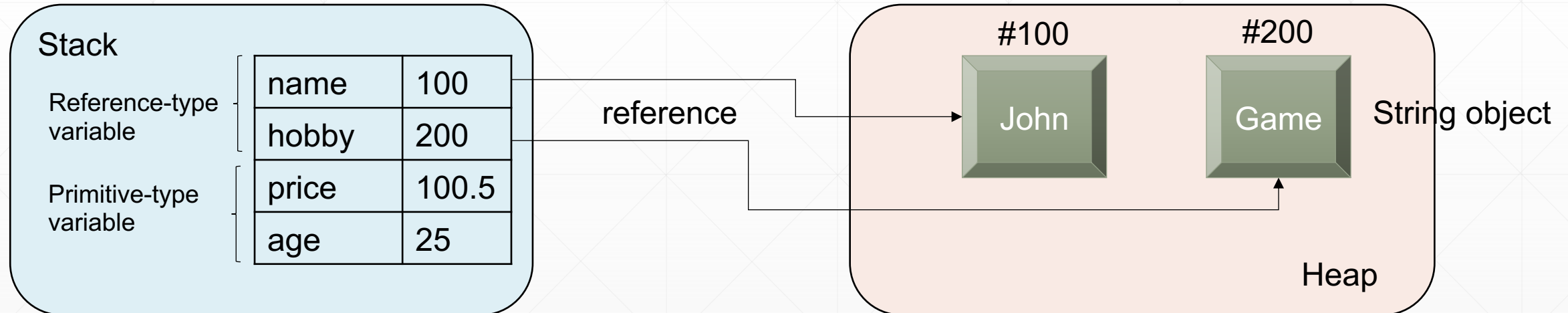
- Store a value in the variable (stack)

### ➤ Reference type variable

- Store a value in the memory (heap)
- Store an address of the memory for further reference

```
int age = 25;  
double price = 100.5;
```

```
String name = "John";  
String hobby = "Game";
```



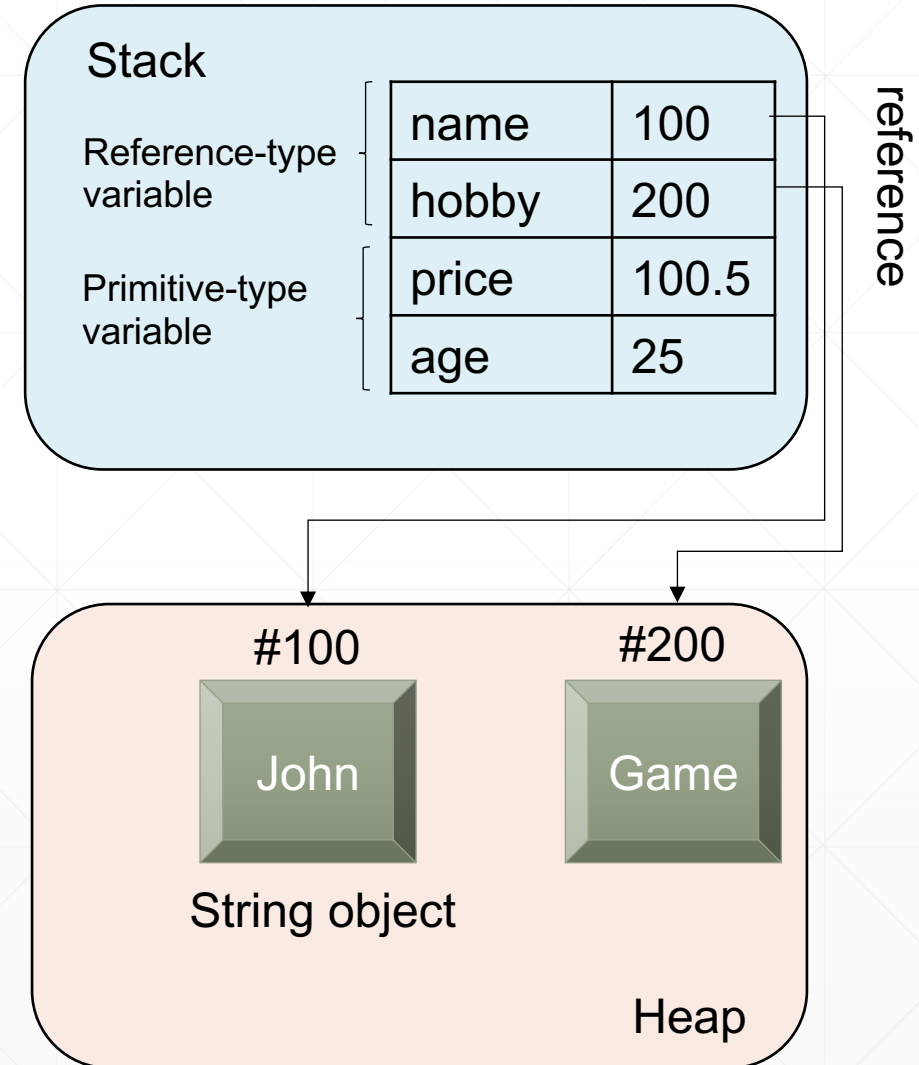
# Reference Type (cont'd)

## ■ Stack

- Allocated for each thread
- Temporary memory based on the method call
- Memory blocks for the method are stacked
  - Local variables, references, etc
  - Only accessible in the block
- After the method is finished, the memory space for the data is cleared

## ■ Heap

- Created by JVM
- Uses as long as the application is running
- All objects are stored in a heap with a global access, therefore, can be referenced from anywhere in the app



# Reference Type (cont'd)

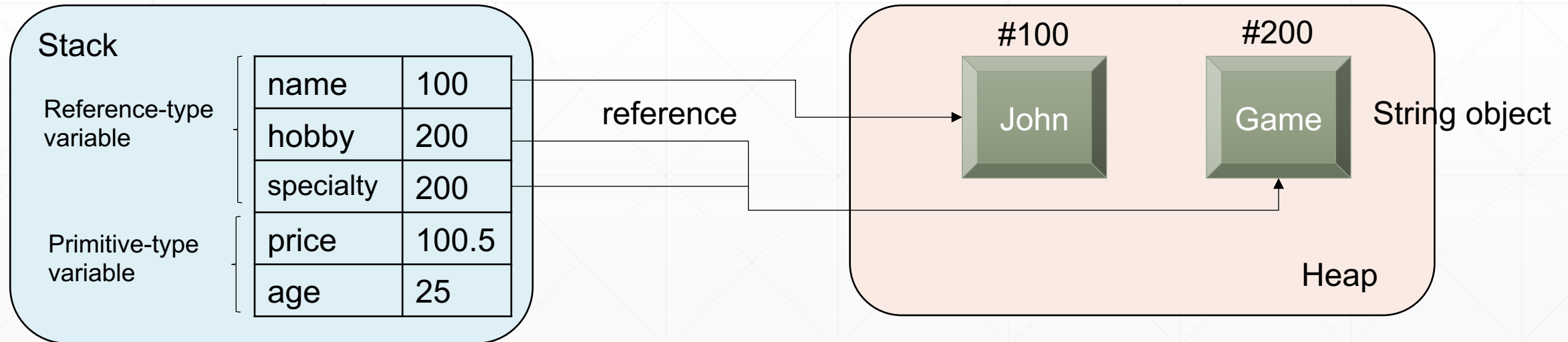
## ■ ==, != operations

### ➤ Primitive type variable

- Check if two values are same or not

### ➤ Reference type variable

- Check if two variables are pointing the same reference or not

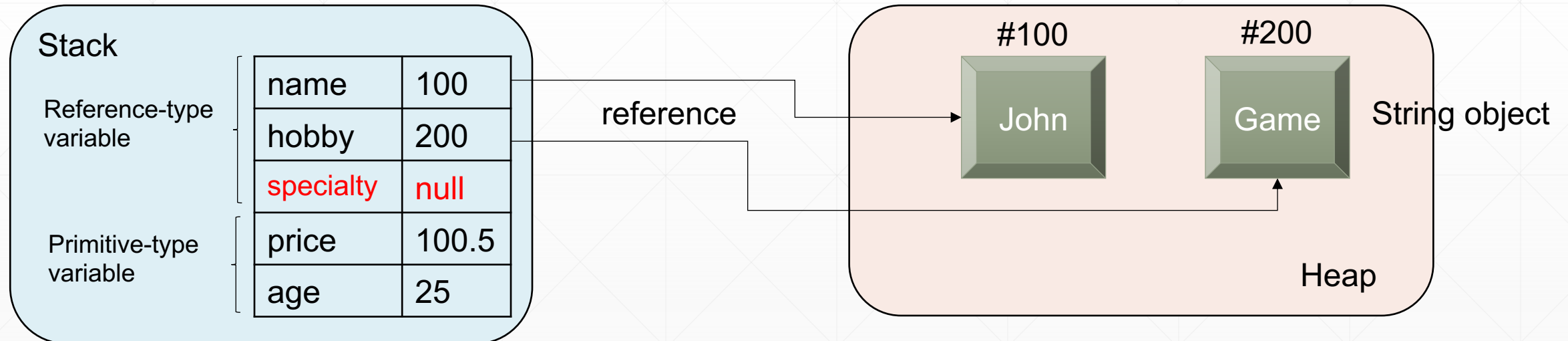


# Reference Type (cont'd)

## ■ null

- Can be used for a reference type
- Can be used as a default value when a reference-type variable does not point anything
- !=, == operations can be used for null type

name == null (false)  
specialty == null (true)





# Reference Type (cont'd)

## ■ NullPointerException (NPE)

- One of Exceptions (will be discussed later)
  - Program error
- When if we try to use variables/methods of null

```
int[] intArray = null;  
intArray[0] = 10;      //NullPointerException
```

```
String str = null;  
System.out.println("  Length: " + str.length());  //NullPointerException
```

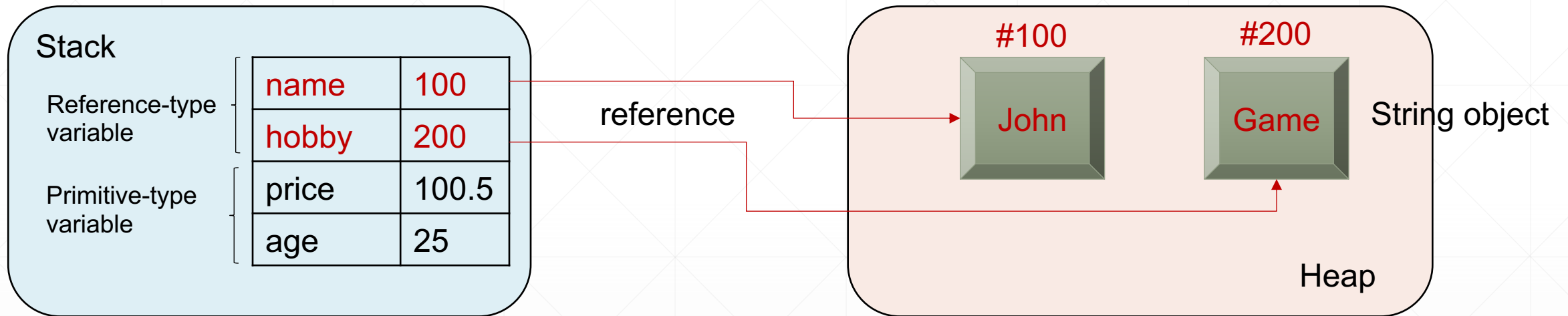
# Reference Type (cont'd)

## ■ String

- Class to store a string value

```
int age = 25;  
double price = 100.5;
```

```
String name = "John";  
String hobby = "Game";
```

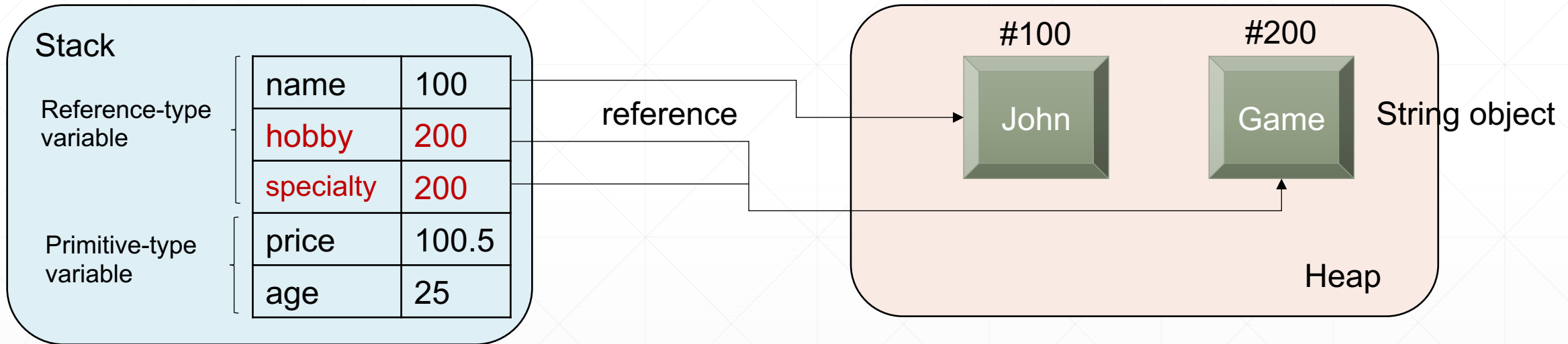


- Generation of String object using “new” keyword
  - Generate a new String object in Heap area
  - The address of the memory is returned

# Reference Type (cont'd)

## ■ String

- String object can be shared if string literals are same



```
String hobby = "Game";  
String specialty = "Game";
```

# Reference Type (cont'd)

## ■ Enumeration

- Special data type to store a set of constants
- Common example
  - Representing compass directions: {NORTH, SOUTH, EAST, WEST}
  - Representing the days of a week: {SUNDAY, MONDAY, TUESDAY, ..., SATURDAY}
- Enum-type variable must be equal to one of the values that have been predefined for it
- Declaration 

`public enum Enumtype { ...(a set of enum constants) }`

  - Need to be declared in the java file with the same Enumtype name
  - Enum constant should be CAPITAL (naming convention)

```
public enum Week { MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, ... }
```

```
public enum LoginResult { LOGIN_SUCCESS, LOGIN_FAILED }
```

# Reference Type (cont'd)

## ■ Enumeration

- Declaration of Enum type variable

```
Enumtype variableName;
```

```
Week today;
```

```
Week reservationDay;
```

- Assigning a value to Enum type variable

- Value must be equal to one of the values that have been predefined for it

```
Enumtype variableName = Enumtype.constant;
```

```
Week today = Week.SUNDAY;
```

- Enum type is a kind of reference type

- Enum-type variable can use null literal

```
Week birthday = null;
```

# Reference Type (cont'd)

## ■ Example)

Hello.java

```
public class Hello {  
    public static void main(String[] args) {  
  
        Week myDay = Week.FRIDAY;  
  
        switch (myDay) {  
            case MONDAY:  
                System.out.println("Mondays are bad.");  
                break;  
            case FRIDAY:  
                System.out.println("Fridays are better.");  
                break;  
            case SATURDAY: case SUNDAY:  
                System.out.println("Weekends are best.");  
                break;  
            default:  
                System.out.println("Midweek days are so-so.");  
                break;  
        }  
    }  
}
```

Week.java

```
public enum Week {  
    MONDAY,  
    TUESDAY,  
    WEDNESDAY,  
    THURSDAY,  
    FRIDAY,  
    SATURDAY,  
    SUNDAY  
}
```



# Basics **Array**

# When We need an Array?

---

- We want to calculate the grades for 5 students

- Store each student's score, then
- Calculate mean, min, max of all students' scores!

- We need five variables to do this!

- score1
- score2
- score3
- score4
- score5



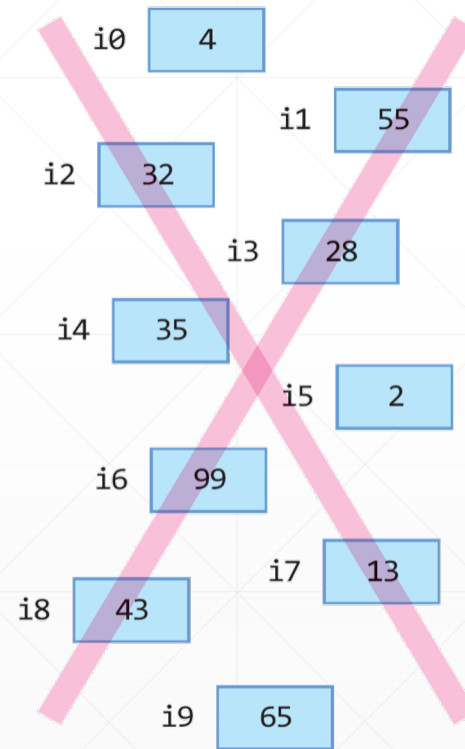
# When We need an Array? (cont'd)

- What about 50 students
  - Store each student's score, then
  - Calculate mean, min, max of all students' scores!
- We need fifty variables to do this...?
  - score1, score2, score3, score4, score5 ... score50?
  - Ok..
- What about 500 students?
  - We need five hundred variables then?
- So, we need a data structure to store a list of elements! (like, array!)

# Array: Characteristics

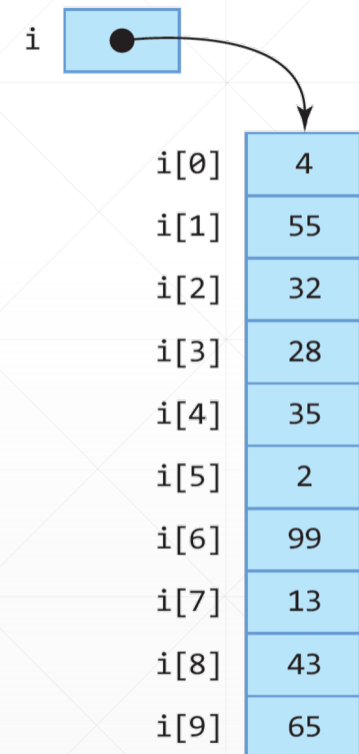
- Data structure used to store multiple values in a single variable, instead of declaring separate variables for each value
- A container object that holds a fixed number of values of a single type

```
int i0, i1, i2, i3, i4, i5, i6, i7, i8, i9;
```



```
sum = i0+i1+i2+i3+i4+i5+i6+i7+i8+i9;
```

```
int i[] = new int[10];
```

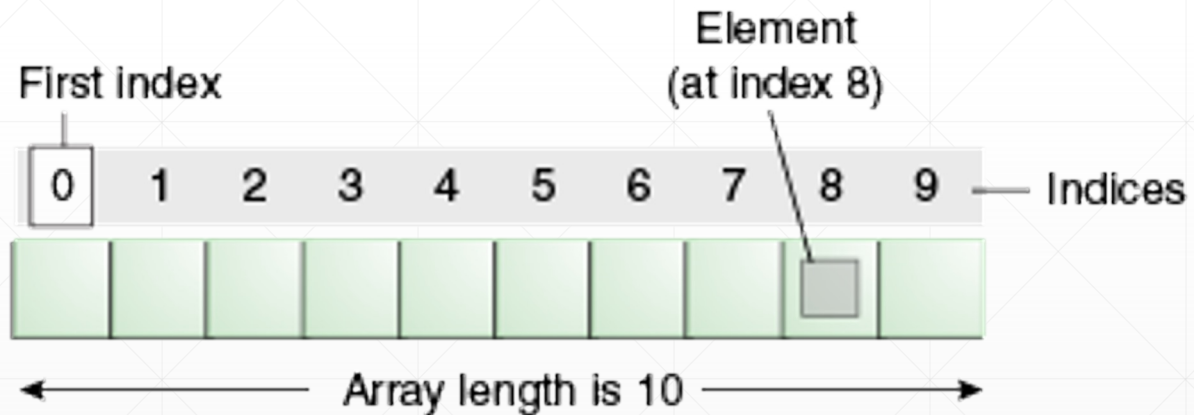


```
for(sum=0, n=0; n<10; n++)  
    sum += i[n];
```

# Array: Index

## ■ Element

- Each item in an array
- Accessed by its numerical index
- Index number begins with 0



# Array: Declaration and Creation

## ■ Declaration

type[] variable;

```
int[] intArray;  
double[] doubleArray;  
String[] strArray;
```

type variable[];

```
int intArray[];  
double doubleArray[];  
String strArray[];
```

## ■ Array variable can be null

- If null, accessing array elements (i.e., array[index]) is not possible
  - NullPointerException NPE occurs!

# Array: Declaration and Creation (cont'd)

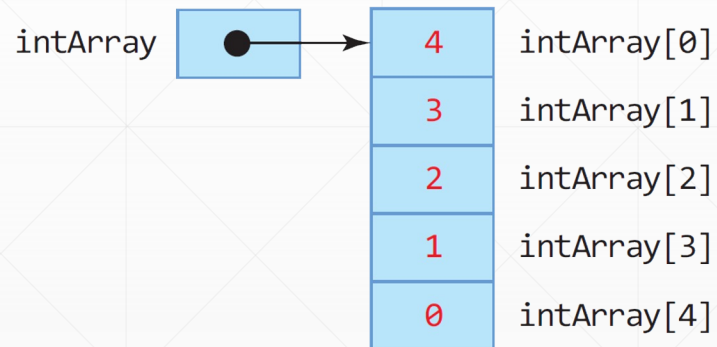
## ■ Declaration with initialization

`type[] variable = { value0, value1, value2, ...};`

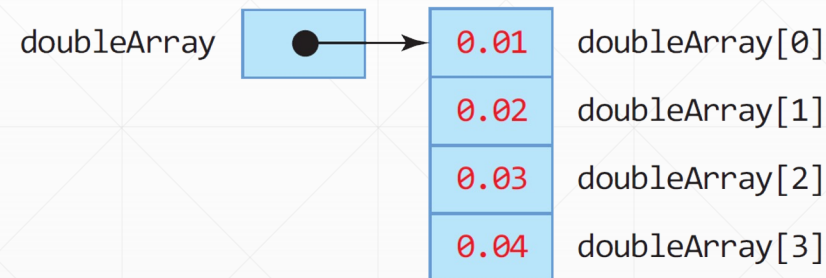
### ➤ Example)

```
int intArray[] = {4,3,2,1,0};  
double doubleArray[] = {0.01, 0.02, 0.03, 0.04};
```

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



`double doubleArray[] = {0.01, 0.02, 0.03, 0.04};`

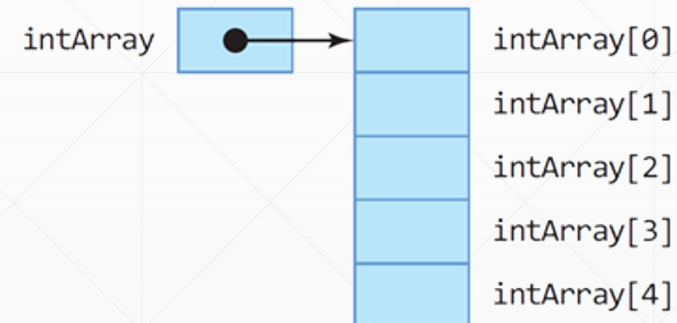
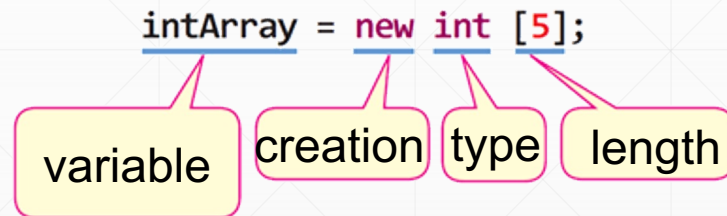
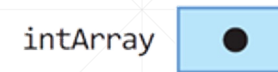
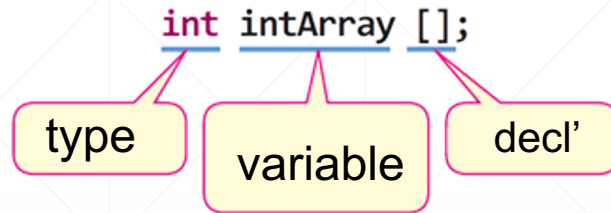


# Array: Declaration and Creation (cont'd)

## ■ Creation after Declaration

`type[] variable; // array declaration`

`variable = new type[length]; // array creation (memory allocation)`



# Array: Access

## ■ Accessing array element

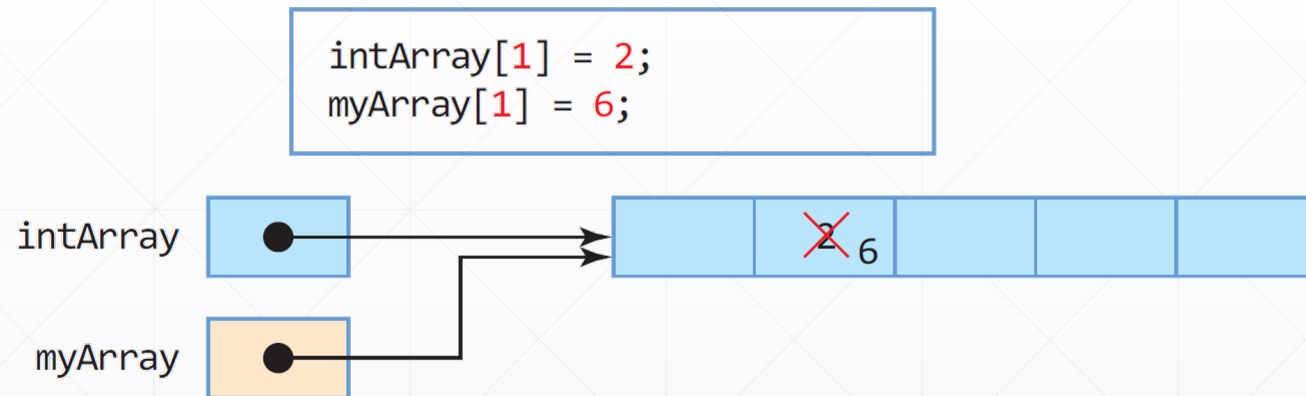
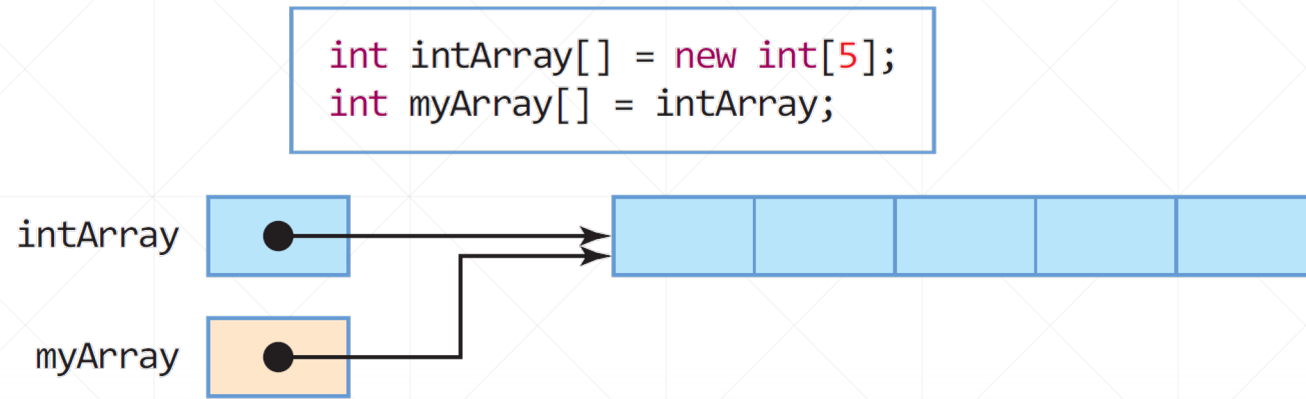
`ArrVariable[index]`

- Get the element located in the position of *index* in the array
- Index begins with 0
- The index of the last element in an array is (the length of the array -1)

```
int intArray [] = new int[5]; // create an array with size of 5 (index: 0~4)
intArray[0] = 5; // store 5 to index 0
intArray[3] = 6; // store 6 to index 3
int n = intArray[3]; // access index 3 of intArray and the assign the value to variable n
n = intArray[-2]; // error!
n = intArray[5]; // error!
```

# Array: Access (cont'd)

- A single array can be shared with multiple references





# Array: Access (cont'd)

- Example) take 5 positive integers from the user, store them in an array, and print the max value!

```
Scanner scanner = new Scanner(System.in);

int intArray[] = new int[5];    // create an array

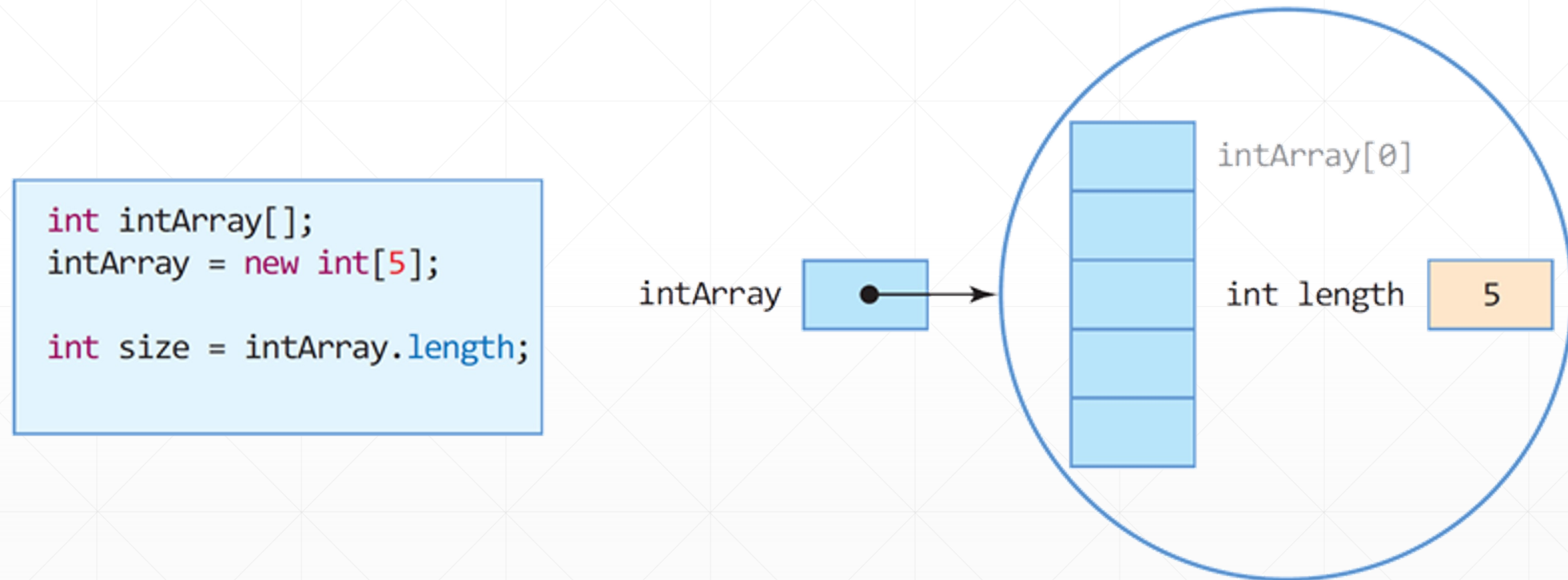
int max = 0;    // current max value
System.out.println("Input 5 Positive Numbers.");
for (int i = 0; i < 5; i++) {
    intArray[i] = scanner.nextInt();    // store the input value to the array
    if (intArray[i] > max)    // if intArray[i] is greater than the current max
        max = intArray[i];    // then set intArray[i] as current max
}
System.out.print("The maximum value is " + max + ".");

scanner.close();
```

# Array: Length

■ Array is a kind of Object in Java

➤ Length field of an array represents the size of the array



# Array: Length

## ■ Example)

- Take a set of integers from the user to fill out the array
- Use the length of an array to determine how many times a user needs to type the number!

```
int intArray[] = new int[5];
int sum = 0;

Scanner scanner = new Scanner(System.in);
System.out.println("Input " + intArray.length + " numbers: ");
for (int i = 0; i < intArray.length; i++)
    intArray[i] = scanner.nextInt(); // store an integer value to the array

for (int i = 0; i < intArray.length; i++)
    sum += intArray[i]; // sum up all the values in the array using for statement

System.out.print("Average is " + (double) sum / intArray.length);
scanner.close();
```

# Array: For-Each

- Advanced for statement to iterate each element in the array or enum

- Do not need to check the length of an array in the loop!

for (type variable : array)

```
int[] num = { 1,2,3,4,5 };  
int sum = 0;  
for (int k : num) // for each iteration, k is set to num[0], num[1], ..., num[4]  
    sum += k;  
System.out.println("Sum: " + sum);
```

```
String names[] = { "apple", "pear", "banana", "cherry", "strawberry", "grape" } ;  
for (String s : names)  
    System.out.print(s + " ");
```

```
enum Week { MON, TUE, WED, THU, FRI, SAT, SUN }  
for (Week day : Week.values())  
    System.out.print(day + " ");
```

# Array: 2D-array

## ■ Declaration

```
int    intArray[][];  
char   charArray[][];  
double doubleArray[][];
```

```
int[][] intArray;  
char[][] charArray;  
double[][] doubleArray;
```

## ■ Creation

```
intArray = new int[2][5];  
charArray = new char[5][5];  
doubleArray = new double[5][2];
```

```
int    intArray[][] = new int[2][5];  
char   charArray[][] = new char[5][5];  
double doubleArray[][] = new double[5][2];
```

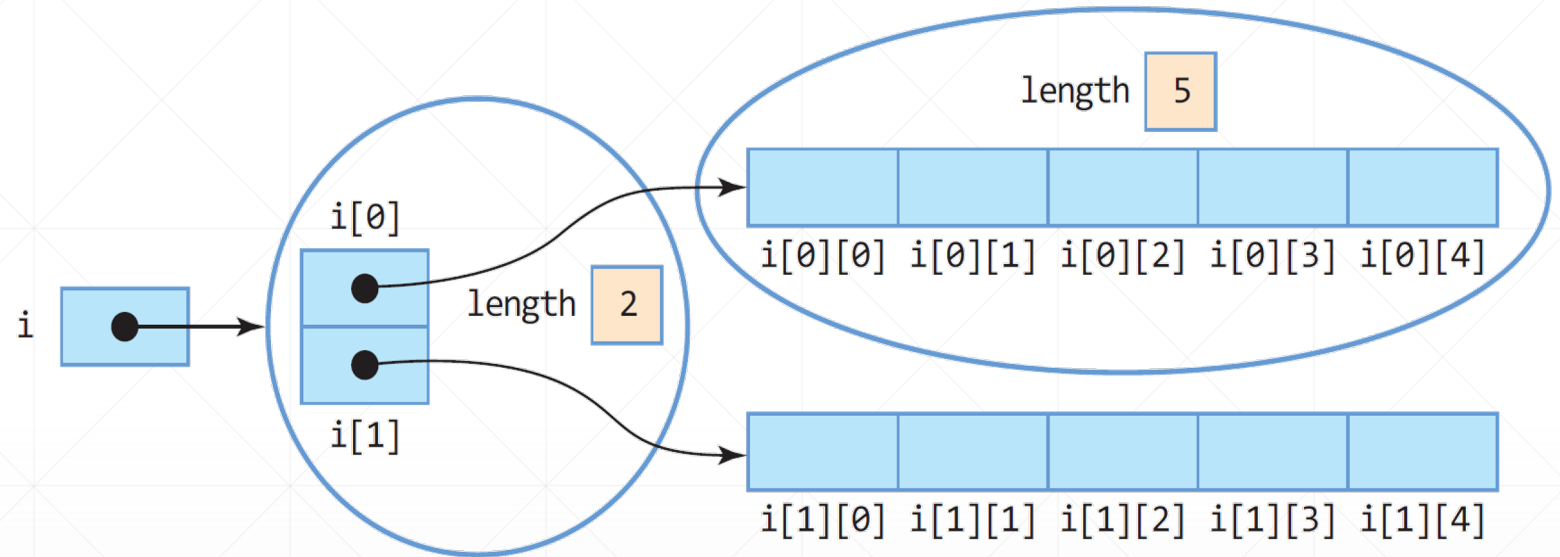
## ■ Declaration with initialization

```
int intArray[][] = {{0,1,2},{3,4,5},{6,7,8}};  
char charArray[][] = {{'a', 'b', 'c'},{'d', 'e', 'f'}};  
double doubleArray[][] = {{0.01, 0.02}, {0.03, 0.04}};
```

# Array: 2D-array (cont'd)

## ■ Conceptual view

```
int i[][] = new int[2][5];  
int size1 = i.length; // 2  
int size2 = i[0].length; // 5  
int size3 = i[1].length; // 5
```



## ■ The length of 2D-array

- `i.length = 2` (the number of row)
  - `i[0].length = 5` (length of n-th 1D-array)
  - `i[1].length = 5` (length of n-th 1D-array)

# Array: 2D-array (cont'd)

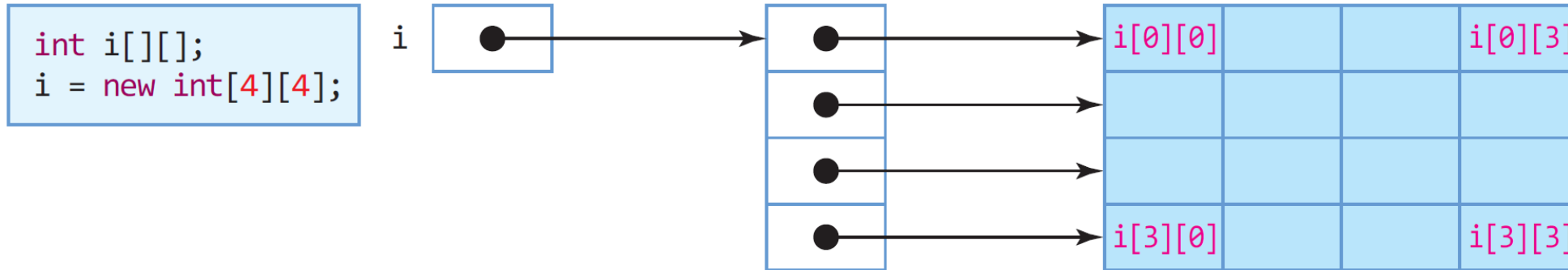
- Store GPA scores for each year/semester in a 2d array, and then calculate their average

```
double score[][] = {{3.3, 3.4},      // GPA for the 1st year
                   {3.5, 3.6},      // GPA for the 2nd year
                   {3.7, 4.0},      // GPA for the 3rd year
                   {4.1, 4.2}};    // GPA for the 4th year
double sum = 0;
for (int year = 0; year < score.length; year++) // for each year
    for (int term = 0; term < score[year].length; term++) // for each semester
        sum += score[year][term]; // sum-up!

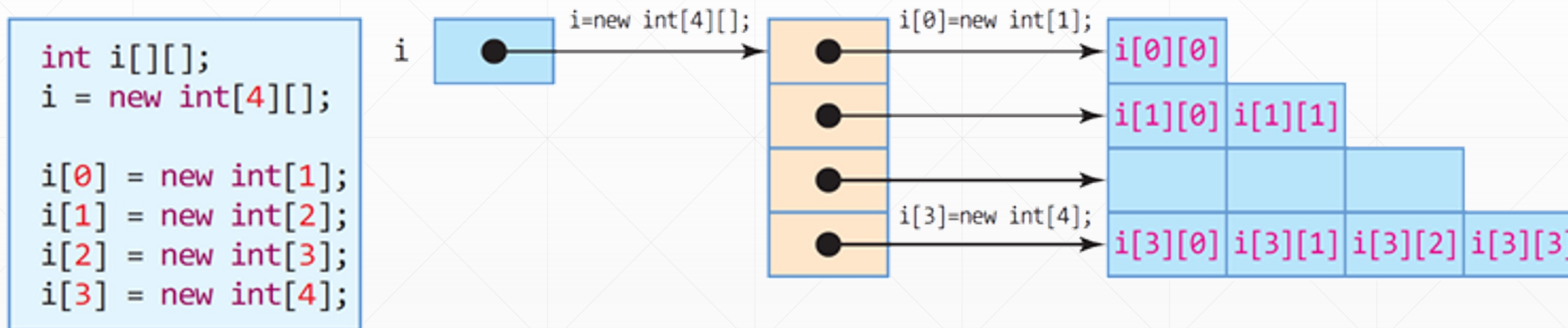
int n = score.length; // the number of rows
int m = score[0].length; // the number of columns
System.out.println("Total GPA is " + sum / (n * m));
```

# Array: 2D-array (cont'd)

## ■ Square array



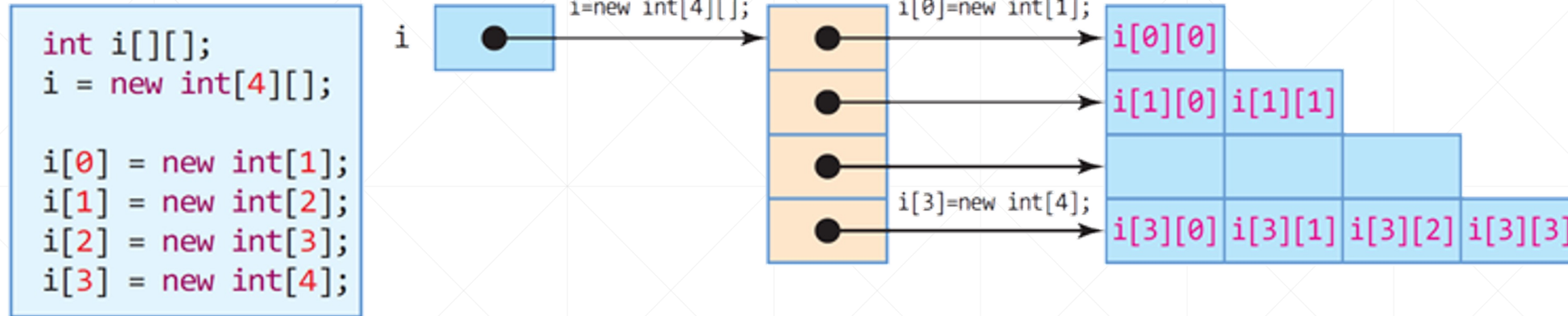
## ■ Non-square array





# Array: 2D-array (cont'd)

## ■ Non-square array



## ■ The length of a non-square 2D-array

- `i.length = 4` (the number of row)
  - `i[0].length = 1` (length of n-th 1D-array)
  - `i[1].length = 2` (length of n-th 1D-array)
  - `i[2].length = 3` (length of n-th 1D-array)
  - `i[3].length = 4` (length of n-th 1D-array)

# Array: 2D-array (cont'd)

- Example) Create a non-square array as shown in the following figure, initialize the array, and print it.

10	11	12
20	21	
30	31	32
40	41	

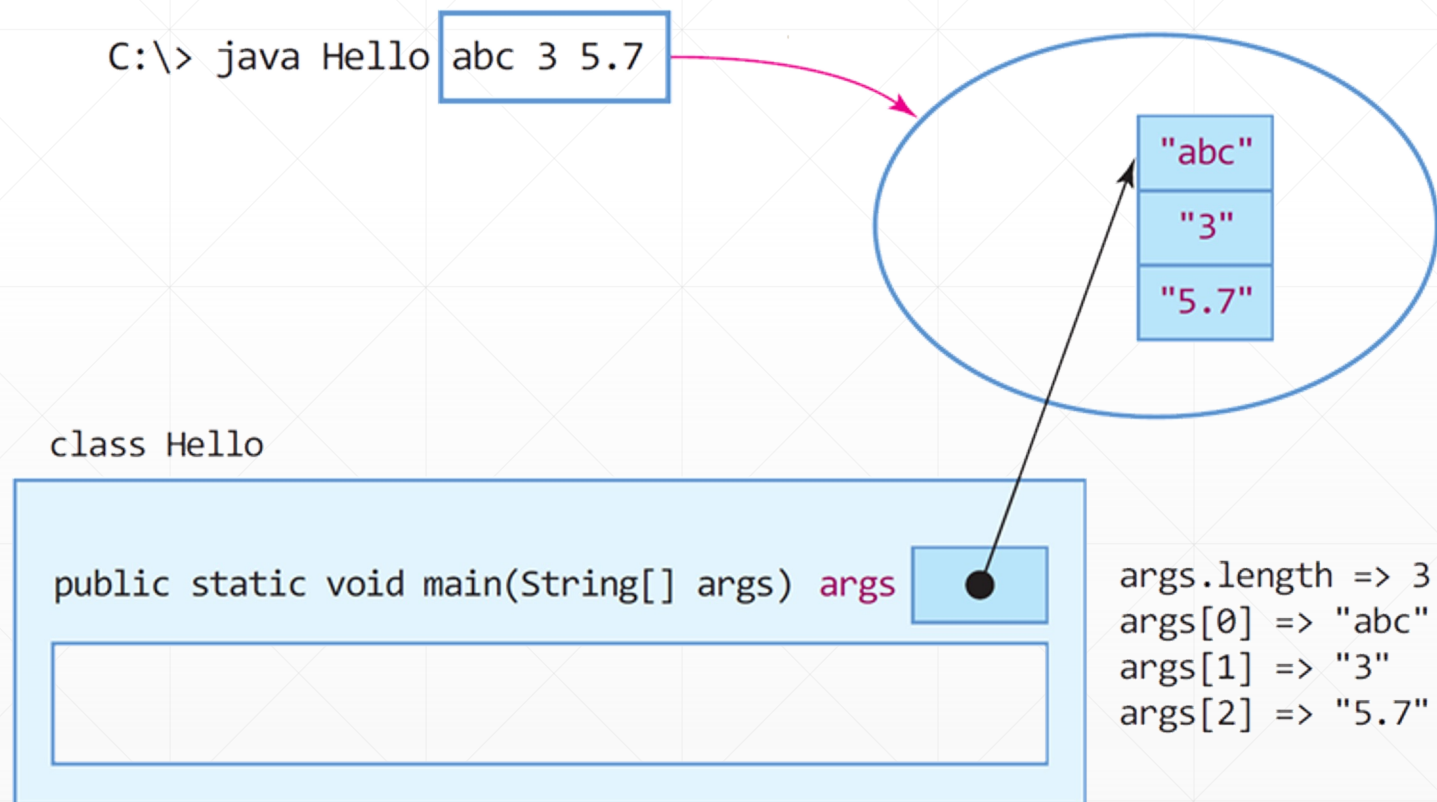
```
int intArray[][] = new int[4][];  
intArray[0] = new int[3];  
intArray[1] = new int[2];  
intArray[2] = new int[3];  
intArray[3] = new int[2];
```

```
for (int i = 0; i < intArray.length; i++)  
    for (int j = 0; j < intArray[i].length; j++)  
        intArray[i][j] = (i + 1) * 10 + j;
```

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

# Array: Misc.

- What is String[] args in the main method?
- main() is the entry point of Java application
  - Arguments for starting Java application is passed through args String array



# Array: Misc. (cont'd)

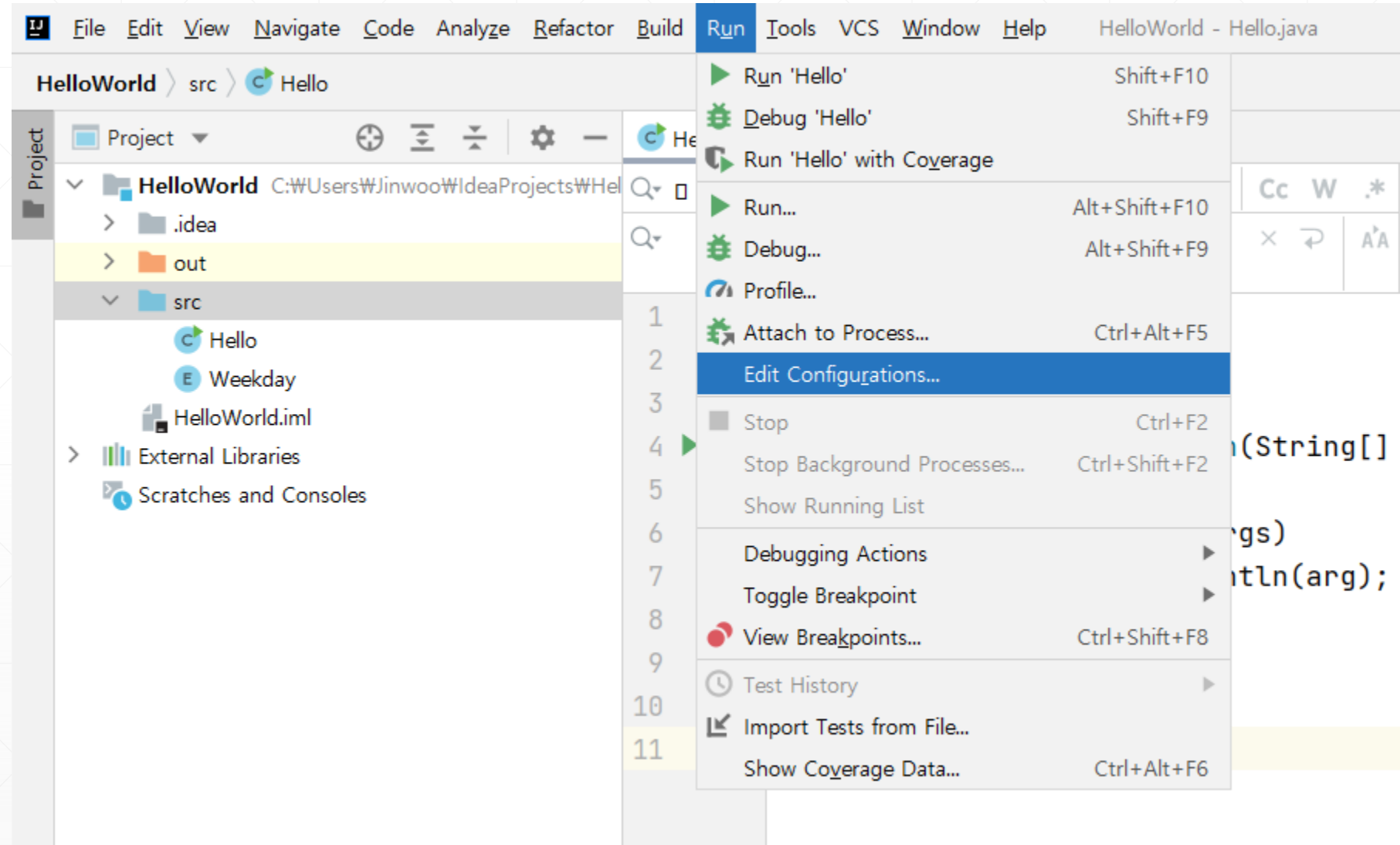
- Example) print out the contents of args array in the main method

```
public class Hello {  
    public static void main(String[] args) {  
  
        for(String arg : args)  
            System.out.println(arg);  
  
    }  
}
```

- If no arguments are set, then nothing is printed

# Array: Misc. (cont'd)

## ■ Edit Configuration



# Array: Misc. (cont'd)

## ■ Edit Configuration: passing arguments

Name:

---

**Build and run**



Name:

---

**Build and run**

CLI arguments to your application. Alt+R

What happens?

# Q&A

---

- Next week (eClass video)
  - OOD/P: Class and Methods