Java Programming

— Basics of Java Programming: Arrays —

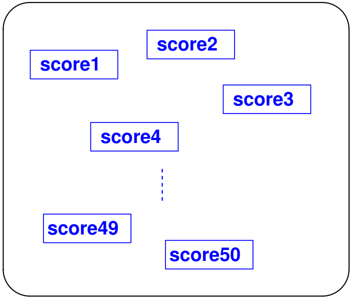
Waseda University

Problem

Example

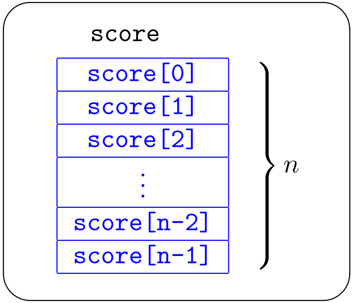
When test scores for 50 persons are given, calculate average of test scores and output it.

- We need 50 variables for data for 50 persons.
- It is troublesome to declare and manage 50 variables.
- We can regard data for 50 persons as variables that have common property "test scores".
- We can handle many variables by using **arrays** efficiently.
  Instead of declaring individual variables, such as number0, number1, . . . , and number99, you declare one array variable.



Arrays

- Java provides a data structure, the array, which stores a fixed-size sequential collection of elements of the same type.
- To use an array in a program, you must declare a variable to reference the array and specify the array 's element type.
- Arrays have data type like variables.
- To use arrays, we can manage n variables by using one variable name like right figure.



Declaration of an array (1)

Example: Declare an “int type array data”

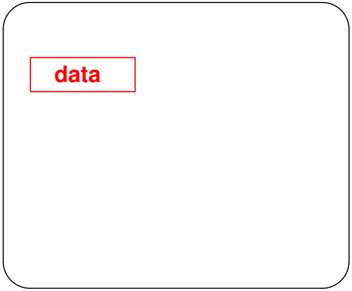
1. Declare an array variable name

   `int[] data;` — method1

   OR

   `int data[];` — method2

   - We can use both methods, throughout the lesson, we use method1.



Declaration of an array (1)

Example: Declare an "`int` type array name `data`".

1. Declare an array variable name

   `int[] data;` — method1

   OR

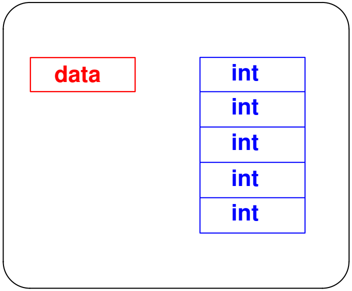
   `int data[];` — method2

   - We can use both methods, throughout the lesson, we use method1

2. Create an array

   `new int[5]`

   - Creates an array of 5 elements of `int` type.



Declaration of an array (1)

Example : Declare an "`int` type array name `data`"

1. Declare an array variable name

   `int[] data;` — method1

   OR

   `int data[];` — method2

   - We can use both methods, throughout the lesson, we use method1.

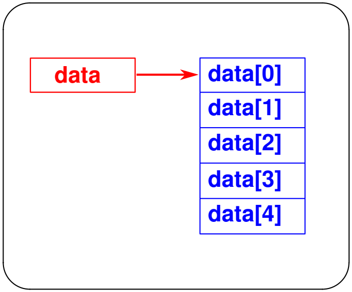
2. Create an array

   `new int[5]`

   - Creates an array of 5 elements of `int` type.

   `data = new int[5];`

   - It assigns the reference of the newly created array to the variable `data`. The array has 5 int type elements.



Declaration of an array (2)

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

- We can declare an "`int` type array `data`".
- The `data` is the array of 5 elements of `int` type.
- The elements of `data` are

  `data[0]`, `data[1]`, `data[2]`, `data[3]`, `data[4]`

  . An indexed variable can be used in the same way as a regular variable.
- Numbers in brackets `[ ]` is called the **index**. The array elements are accessed through the index. Please note that the index must be from 0. (Array indices are 0 based.)

Declaration of an array (3)

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

- We can write the declaration and creation of an array at once.

Accessing array elements (1)

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

data[0] = 10;
data[1] = data[0]*3;
```

- An indexed variable can be used in the same way as a regular variable.
- In the above, 5 elements (`data[0]`$\sim$`data[4]`) are regarded as variables.

Accessing array elements (2)

```
public class SampleArray {
  public static void main(String[] args) {
    int[] data = new int[5];
    for (int i=0; i<5; i++) {
      data[i] = 100 - i*10;
      System.out.println(data[i]);
    }
  }
}
```

```
data[0] = 100
data[1] = 90
data[2] = 80
data[3] = 70
data[4] = 60
```

- We can use variables as the index.
- When processing array elements, you will often use a for loop.

Accessing array elements (3)

Wrong access (Error)

- Be careful not to have the loop try to access elements that are outside the range of the array.
- An execution error will occur.

```
int i;
int[] data = new int[5];
for (i=0; i<=5; i++) {
  data[i] = 100 - i*10;
  System.out.println(data[i]);
}
```

- It refers to `data[5]`, but it does not exist.
- Compilation is succeed, but an execution error will occur like the following:

```
>> Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
```

Accessing array elements (4)

Length of array

- In Java, length of array (array size) is stored.
- The size of an array object is available in "array name.length".

```
int i;
int[] data = new int[5];
for (i=0; i<data.length; i++) {
  data[i] = 100 - i*10;
  System.out.println(data[i]);
}
```

- data.length is 5.

Initialization of array (1)

```
int[] data = {100, 90, 80, 70, 60};
```

- Elements of array are initialized like variables.
- Java supports braces `{ }` construct for creating an array and initializing its elements.
- The size of the array is determined by the number of values provided in the set of braces without explicitly specifying it inside the brackets.
- `data[0]`=100, `data[1]`=90, `data[2]`=80, `data[3]`=70, `data[4]`=60

```
int[] data;
data = {100, 90, 80, 70, 60};   — wrong
```

- Splitting it would cause a syntax error. Thus, the above statement is wrong.

Initialization of array (1)

```
int[] data = {100, 90, 80, 70, 60};
```

- Elements of array are initialized like variables.
- Java supports braces `{ }` construct for creating an array and initializing its elements.
- The size of the array is determined by the number of values provided in the set of braces without explicitly specifying it inside the brackets.
- `data[0]`=100, `data[1]`=90, `data[2]`=80, `data[3]`=70, `data[4]`=60

```
int[] data;
data = {100, 90, 80, 70, 60};   — wrong
```

- Splitting it would cause a syntax error. Thus, the above statement is wrong.

Initialization of array (2)

Initialization of array

- When an array is declared, its elements are assigned the default value.
- 0 for `int` type array, 0.0 for `double` type array.

For variables,

```
int data;                   — Declare only. Doesn't initialize.
System.out.println(data);   — Error
```

For arrays,

```
int i;
int[] data = new int[5];
for (i=0; i<5; i++) System.out.println(data[i]);
```

```
0
0
0
0
0
```

Initialization of array (2)

Initialization of array

- When an array is declared, its elements are assigned the default value.
- 0 for `int` type array, 0.0 for `double` type array.

For variables,

```
int data;                     — Declare only. Doesn't initialize.
System.out.println(data);     — Error
```

For arrays,

```
int i;
int[] data = new int[5];
for (i=0; i<5; i++) System.out.println(data[i]);
```

```
0
0
0
0
0
```

Initialization of array (2)

Initialization of array

- When an array is declared, its elements are assigned the default value.
- 0 for `int` type array, 0.0 for `double` type array.

For variables,

```
int data;                   — Declare only.  Doesn't initialize.
System.out.println(data);   — Error
```

For arrays,

```
int i;
int[] data = new int[5];
for (i=0; i<5; i++) System.out.println(data[i]);
```

```
0
0
0
0
0
```

Creating scores

Problem

When test scores for 50 persons are given, calculate average of test scores and output it.

In this problem, we need to prepare test scores for 50 persons.

We assume that a test score for `i`-th person is given by the following calculation formula:

```
(i*83 + 15) % 101
```

`final` variable (1)

- We can declare a constant variable as `final variable`.
- Put "`final`" before declaring a variable.
- Final variable must be initialized.
- A value of a final variable cannot be changed.
- Typically constant names are declared as all capital letters to help other programmers distinguish them from variables.

```
final int SIZE = 50;
SIZE = 100;                — Cannot change
```

- We declare `SIZE` as a constant for 50.

`final` variable (1)

- We can declare a constant variable as `final variable`.
- Put "`final`" before declaring a variable.
- Final variable must be initialized.
- A value of a final variable cannot be changed.
- Typically constant names are declared as all capital letters to help other programmers distinguish them from variables.

```
final int SIZE = 50;
SIZE = 100;              — Cannot change
```

- We declare `SIZE` as a constant for 50.

`final` **variable (2)**

In example,....

- We declare `SIZE` as a constant for 50 (person's data) by using `final` keyword.
- Calculate test scores by `(i*83 + 15) % 101`.

$$\Downarrow$$

Create test scores for 50 persons:

```
int i;
final int SIZE = 50;
int[] score = new int[SIZE];

for (i=0; i<SIZE; i++) {
  score[i] = (i*83 + 15) % 101;
}
```

Casting (1)

For computing an average,

- Calculation of an average is (summation of scores) / (the number of people)
- Declare int type variable `sum` as a summation of scores. Declare double type variable `ave` as an average of scores.

$$\Downarrow$$

```
sum = 0;
for (i=0; i<SIZE; i++) {
  sum += score[i];
}
ave = sum / SIZE;
```

- Both `sum` and `SIZE` are `int` type.
- "`sum / SIZE`" is int type division, the fractional part is truncated.
- In such a case, `Casting` is useful.

Casting (2)

Typecast operators

- The syntax for casting a type is to specify the target type in parentheses, followed by the variable ’s name or the value to be cast. For example, the following statement:
- `(type)` formula
- Casted data type becomes a given a data type.

In the part of calculation of an average,

```
sum = 0;
for (i=0; i<SIZE; i++) {
  sum += score[i];
}
ave = (double) sum / SIZE;
```

Casting (2)

Typecast operators

- The syntax for casting a type is to specify the target type in parentheses, followed by the variable 's name or the value to be cast. For example, the following statement:
- `(type)` formula
- Casted data type becomes a given a data type.

In the part of calculation of an average,

```
sum = 0;
for (i=0; i<SIZE; i++) {
  sum += score[i];
}
ave = (double) sum / SIZE;
```

Example program

Ave.java

```
public class Ave{
  public static void main(String[] args) {
    int i,sum;
    double ave;
    final int SIZE = 50;
    int[] score = new int[SIZE];

    for(i=0;i<SIZE;i++){
      score[i] = (i*83 + 15) % 101;
    }

    sum = 0;
    for(i=0;i<SIZE;i++){
      sum += score[i];
    }

    ave = (double) sum / SIZE;
    System.out.println("ave : " + ave);
  }
}
```

Two-dimensional array (1)

Example: Declare a two-dimensional array `data`.

```
int[][] data;
data = new int[4][3];
```

data[0][0]	data[0][1]	data[0][2]
data[1][0]	data[1][1]	data[1][2]
data[2][0]	data[2][1]	data[2][2]
data[3][0]	data[3][1]	data[3][2]

- There will be four rows and three columns in the table.
- To declare a two-dimensional array, two sets of brackets are required.
- Each elements `data[i][j]` can be used as `int` type variables.

Two-dimensional array (1)

Example: Declare a two-dimensional array `data`.

```
int[][] data;
data = new int[4][3];
```

data[0][0]	data[0][1]	data[0][2]
data[1][0]	data[1][1]	data[1][2]
data[2][0]	data[2][1]	data[2][2]
data[3][0]	data[3][1]	data[3][2]

- There will be four rows and three columns in the table.
- To declare a two-dimensional array, two sets of brackets are required.
- Each elements `data[i][j]` can be used as `int` type variables.

Two-dimensional array (1)

Example: Declare a two-dimensional array `data`.

```
int[][] data;
data = new int[4][3];
```

data[0][0]	data[0][1]	data[0][2]
data[1][0]	data[1][1]	data[1][2]
data[2][0]	data[2][1]	data[2][2]
data[3][0]	data[3][1]	data[3][2]

- There will be four rows and three columns in the table.
- To declare a two-dimensional array, two sets of brackets are required.
- Each elements `data[i][j]` can be used as `int` type variables.

Two-dimensional array (2)

```
int[][] data = new int[4][3];
```

- We can write the declaration and creation of a two-dimensional array at once similar to one-dimensional array..