OBJECT-ORIENTED SYSTEMS DESIGN
[Exercise]: Arrays

2022/04/20

Department of Computer Science,
Hanyang University

Lecturer: Taeuk Kim
TA: Taejun Yoon, Seong Hoon Lim, Young Hyun Yoo

Today's Plan

1. **Chapter Review: 10 min.**

2. **Practice : 40 min.**

Creating and Accessing Arrays

- **An *array* is a data structure used to process a collection of data that is all of the same type.**
  - An array behaves like a numbered list of variables with a uniform naming mechanism.
  - It has a part that **does not change**: the name of the array.
  - It has a part that can **change**: an integer in square brackets.

Define the array identifier **`arr.`**

Each of the integers is initialized to 0 by default.

In Java, we are not limited to fixed-size array declaration.

```
public static void main(String[] args) {
    int[] arr1 ;                //java style
    int arr2[];                 //C style

    int[] arr3;                 //declare the array variable
    arr3 = new int[10];         //create the array ;assign to array variable

    int[] arr4 =  new int[10];

    Scanner sc = new Scanner(System.in);
    int size;
    double[] arr5;

    size = sc.nextInt();
    arr5 = new double[size];
```

Arrays with a Class Base Type

- The base type of an array can be a class type.

```
Date[] holidayList = new Date[20];

if (holidayList[0] == null)
    System.out.println("holidayList[0] == Null");
```

```
holidayList[0] == Null
```

- It does not create 20 objects of the class **`Date`**.
- Each of these indexed variables are automatically initialized to **`null`**.
- Each of the indexed variables can now be referenced since each holds the memory address of a **`Date`** object.

- Initializing an array by using a **`for`** loop:

```
for (int index =0 ; index < holidayList.length ; index++){
    holidayList[index] = new Date();
}

if (holidayList[0] != null)
    System.out.println("holidayList[0] != Null");
```

```
holidayList[0] != Null
```

- Like any other object, each of the indexed variables requires a separate invocation of a constructor using **`new`** (perhaps using a **`for`** loop) to create an object to reference.

Privacy Leaks with Array Instance Variables

- **Arrays are objects; If an accessor method does return the contents of an array, special care must be taken.**

```
class Array_ex{
    private int[] arr ={1,2,3};

    public int[] getArr(){
        return arr;
    }

    @Override
    public String toString() {
        return "Array_ex{" +
                "arr=" + Arrays.toString(arr) +
                '}';
    }
}

public class Array_demo2 {
    public static void main(String[] args) {
        Array_ex array_ex = new Array_ex();
        int [] arr =  array_ex.getArr();
        System.out.println(array_ex.toString());
        arr[0] = 3;
        System.out.println(array_ex.toString());
    }
}
```

```
Array_ex{arr=[1, 2, 3]}
Array_ex{arr=[3, 2, 3]}
```

Can change private value **`arr`** without mutator method: ***privacy leak!!!!***

The accessor method should return a reference to a *deep copy* of the private array object.

```
public int[] getArr(){
    int[] temp = new int[this.arr.length];
    for(int i = 0; i < this.arr.length; i++)
        temp[i] = this.arr[i];
    return temp;
}
```

Multidimensional Arrays

```
int row = 10;
int col = 5;

int[][] arr2d = new int[row][col];
double[][][] arr3d = new double[4][5][6];
char[][] a = new char[5][12];
```

Multidimensional arrays are declared and created in basically the same way as one-dimensional arrays.

The instance variable **`length`** does not give the total number of indexed variables in a two-dimensional array.

```
System.out.println(arr2d.length);
System.out.println(arr2d[0].length);
```

```
10
5
```

`Row length = arr2d.length`
`Column length = arr2d[0].length`

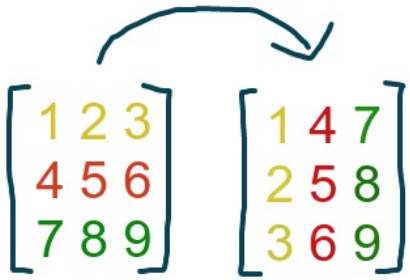
```
public static void printArr(int[][] arr){
    int row,col;
    for(row = 0; row < arr.length; row++){
        for(col = 0; col < arr[0].length; col++)
            System.out.print(arr[row][col]);
        System.out.println();
    }
}
```

A nested **`for`** loop can be used to process a two-dimensional array.

Dealing with Multidimensional Arrays: Transpose

```java
public class Transpose {
    public static void main(String[] args) {
        int[][] arr = {{1,2,3,4},{5,6,7,8},{9,8,7,6}};
        printArr(arr);
        System.out.println();

        int[][] arr_T = transpose(arr);
        printArr(arr_T);
    }
    public static int[][] transpose(int[][] arr){
        int[][] temp = new int[arr[0].length][arr.length];
        for(int i = 0; i< temp.length; i++){
            for (int j = 0; j<temp[0].length; j++){
                temp[i][j]=arr[j][i];
            }
        }
        return temp;
    }
    public static void printArr(int[][] arr){
        int row,col;
        for(row = 0; row < arr.length; row++){
            for(col = 0; col < arr[0].length; col++)
                System.out.print(arr[row][col]);
            System.out.println();
        }
    }
}
```

- **Transpose**



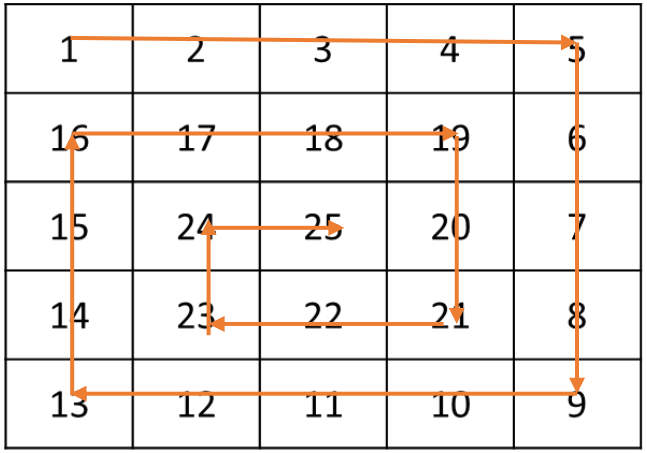

```
1234
5678
9876

159
268
377
486
```

Practice

Exercise/WeekN/practice1.java
/practice2.java

Practice 1: DrawSnail

- **Define a static method `drawSnail`.**
  - `public static int[][] drawSnail(int n)`
  - Given an integer `n`, return an `n` X `n` array whose elements are structured like the right example: Starting from 1, the outermost parts of the array are first filled one by one in a clockwise manner, and then the same thing is recursively done with the inner parts.



- **`Main` method**
  - Enter a matrix size `n`.
  - Construct the resulting snail array using `drawSnail`.
  - Print out the array.

Practice 2: Sorting Elements in an Array

- **Import the `Person` class we defined last week.**
  - `import week7.*;`

- **`Main` method**
  - Create an array of size 10 whose base type is the `Person` class.
    - You should initialize its elements with random `Person` objects.
  - Sort the array in ascending order according to the age of each person.
    - **Hint:** You can use the `AgeCalculator.isOlder()` method to derive who is older between two people.
    - If two people are of the same age, their order doesn't matter; In this case, you can **arbitrarily** determine the order.
    - Implement any sorting algorithm (e.g., selection sort studied in our theory session) to sort the array.
    - **Warning:** You are not allowed to exploit the `Array.sort()` method.

Time for Practice

Get it started, and ask TAs if you are in a trouble.