LECTURE 5
ARRAYS, CLASSES AND OBJECTS

Course: Object Oriented Programming (CS F213)

Prof. Anita Agrawal,

BITS Pilani-K.K.Birla Goa campus

PREDICT THE OUTPUT OF THIS CODE

```
short num1=20000;
short num2=16000;
Short num3=40000;
byte A1=110;
byte A2=120;
byte sum_byte = A1+A2;
int sum_ short = num1+num2 +num3;
System.out.println("sum_short: "+sum_short+"
sum_byte: "+sum_byte);
```

Output:
Sum_short: ??? Sum_byte: ???

EXAMPLE

```
short num1=20000;
short num2=16000;
Short num3=40000;
byte A1=110;
byte A2=120;
byte sum_byte = A1+A2; //causes an error
int sum_ short = num1+num2 +num3; //
Automatically cast to int
System.out.println("sum_short: "+sum_short+"
sum_byte: "+sum_byte);
```

EXAMPLE

```
int i=2000790000;
int j=1108800000;
long sum = i+j;
System.out.println("sum: "+sum);
```

Sum will be -1185377296 (out of range)

EXAMPLE

```
class CharExample{
public static void main(String arg[]){
char a='A';
char b=66;
int i=67;
char c=(char)i;
int j=a;
a=a+1;
a=(char)(a+1);
a+=1;
a++; System.out.println("a: "+a+" b: "+b+" c: "+c+" j: "+j);
}
}
```

```
class CharExample{
public static void main(String arg[]){
char a='A';
char b=66; // No error: Automatic type conversion of literal to the destination type
int i=67;
char c=(char)i; // Explicit casting is required
int j=a; // Automatic type conversion
// a=a+1; // Error: possible lossy conversion from int to char
a=(char)(a+1); // No error due to explicit cast
a+=1; // No error: Internal casting
a++; // No error: Internal casting
System.out.println("a: "+a+" b: "+b+" c: "+c+" j: "+j);
}
}
```

ARRAYS

- An *array* is a container object that holds a fixed number of values of a single type.
- It allows to store and access large number of values conveniently.
- Syntax: *datatype[] identifier*;
  - Example: int [] age;
  - Alternate form: int age [];
- // this form is discouraged

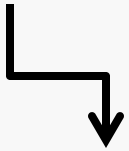
- *dataType* can be a primitive data type like: int, char, Double, byte etc. or an object.

- Identifier (variable name) can be anything provided that it follows the rules and conventions of identifiers.
- int [] age; **[value of age is null]**
- does not actually create an array;
- it simply tells the compiler that this variable will hold an array of the specified type.

CREATING THE ARRAY

- To create an array, use the new operator.

- Syntax: *var-name = new type[size];*
  - Example: age = new int [5];



allocates memory for 5 integer elements and assigns the array to the age variable.

  - Default value: 0 for numeric
  - false for boolean, etc..

ALTERNATE WAY

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

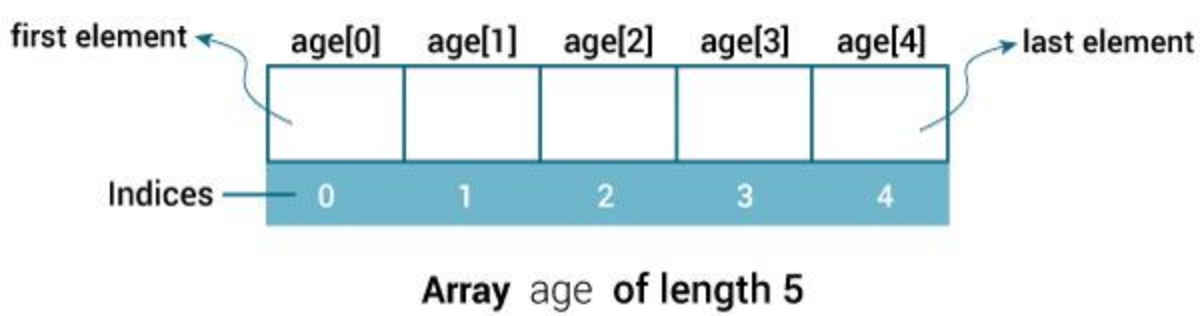
}

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

Imp: *Once the length of the array is defined, it cannot be changed in the program.*

ACCESSING THE ARRAY

- In case of primitive data types, the actual values are stored in contiguous memory locations.
- Each element is accessed by its numerical *index*.
- Numbering begins with 0.
  - The 5th element, for example, would therefore be accessed at index 4.



INITIALISING THE ARRAY

- int[] age = new int[5];

- age[0] = 10; // initialize first element
- age [1] = 5; // initialize second element
- age [2] = 30; // and so forth

ALTERNATE SYNTAX

- Shortcut syntax to create and initialize an array:
  - int[] age = { 10, 5, 30, 25,40 };
- Here the length of the array is determined by the number of values provided between braces and separated by commas.

ACCESSING THE ARRAY

```
class ArrayExample
{
   public static void main(String[] args)
    {
      int[] age = {12, 4, 5, 2, 5};
      for (int i = 0; i < 5; ++i)
       {
         System.out.println("Element at index " + i
           +": " + age[i]);
       }
     }
 }
```

OUTPUT

- Element at index 0: 12
- Element at index 1: 4
- Element at index 2: 5
- Element at index 3: 2
- Element at index 4: 5

ACCESSING THE ARRAY…..CONTD

```
//Access and alter array elements
class ArrayExample {
   public static void main(String[] args) {
      int[] age = new int[5];
      // insert 14 to third element
      age[2] = 14;
      // insert 34 to first element
      age[0] = 34;

      for (int i = 0; i < 5; ++i) {
         System.out.println("Element at index " + i +": " + age[i]);
      }
   }
}
```

OUTPUT OF THE PROGRAM

```
Element at index 0: 34
Element at index 1: 0
Element at index 2: 14
Element at index 3: 0
Element at index 4: 0
```

MULTIDIMENSIONAL ARRAYS

- An array of arrays/multidimensional array.
- An array whose components are themselves arrays, unlike C or Fortran.
- The rows are allowed to vary in length.

Right index:
column

Left index:
row

	Column 1	Column 2	Column 3	Column 4
Row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]

MULTIDIMENSIONAL ARRAYS (CONTD…)

Example:

```
String[][] names = { {"Mr. ", "Mrs. ", "Ms. "},
{“Manish", “Meeta"}
```

- We can also make use of nested for loops for the initialisation of multidimensional arrays.

MULTIDIMENSIONAL ARRAYS (CONTD…)

Print:

Mr. Manish

Ms. Meeta

```
System.out.println(names[0][0] +names[1][0]);
    // Mr Manish
 System.out.println(names[0][2] + names[1][1]);
   //Ms. Meeta
```

MULTIDIMENSIONAL ARRAYS (CONTD…)

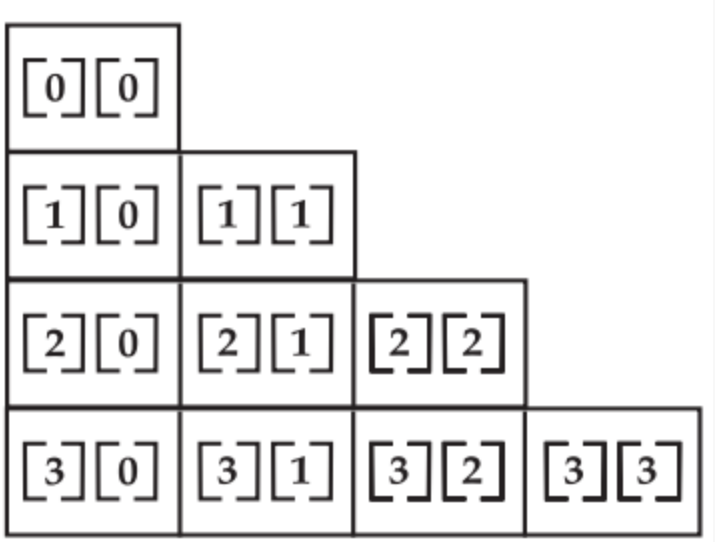
```
class MultiDimArrayDemo
{
    public static void main(String[] args)
     {
         String[][] names =
         {
             {"Mr. ", "Mrs. ", "Ms. "},
             {“Manish", “Meeta"}
         };
        // Mr. Manish
        System.out.println(names[0][0] + names[1][0]);
        // Ms. Meeta
        System.out.println(names[0][2] + names[1][1]);
    }
}
```

MULTIDIMENSIONAL ARRAYS (CONTD…)

- For a multidimensional array, we only need to specify the size for the first (leftmost) dimension
- *int [][] twoD = new int[4][];*
- *twoD[0] = new int[5];*
- *twoD[1] = new int[5];*
- *twoD[2] = new int[5];*
- *twoD[3] = new int[5];*
- *Also possible to write: twoD[0]={10,6,12,13,5};*

MULTIDIMENSIONAL ARRAYS (CONTD…)

- No need to allocate the same number of elements for each dimension
- *int twoD[][] = new int[4][];*
- *twoD[0] = new int[1];*
- *twoD[1] = new int[2];*
- *twoD[2] = new int[3];*
- *twoD[3] = new int[4];*



ALTERNATE DECLARATION

int[] a1, a2, a3;

or

inta1[], a2[], a3[];

Next……classes and objects