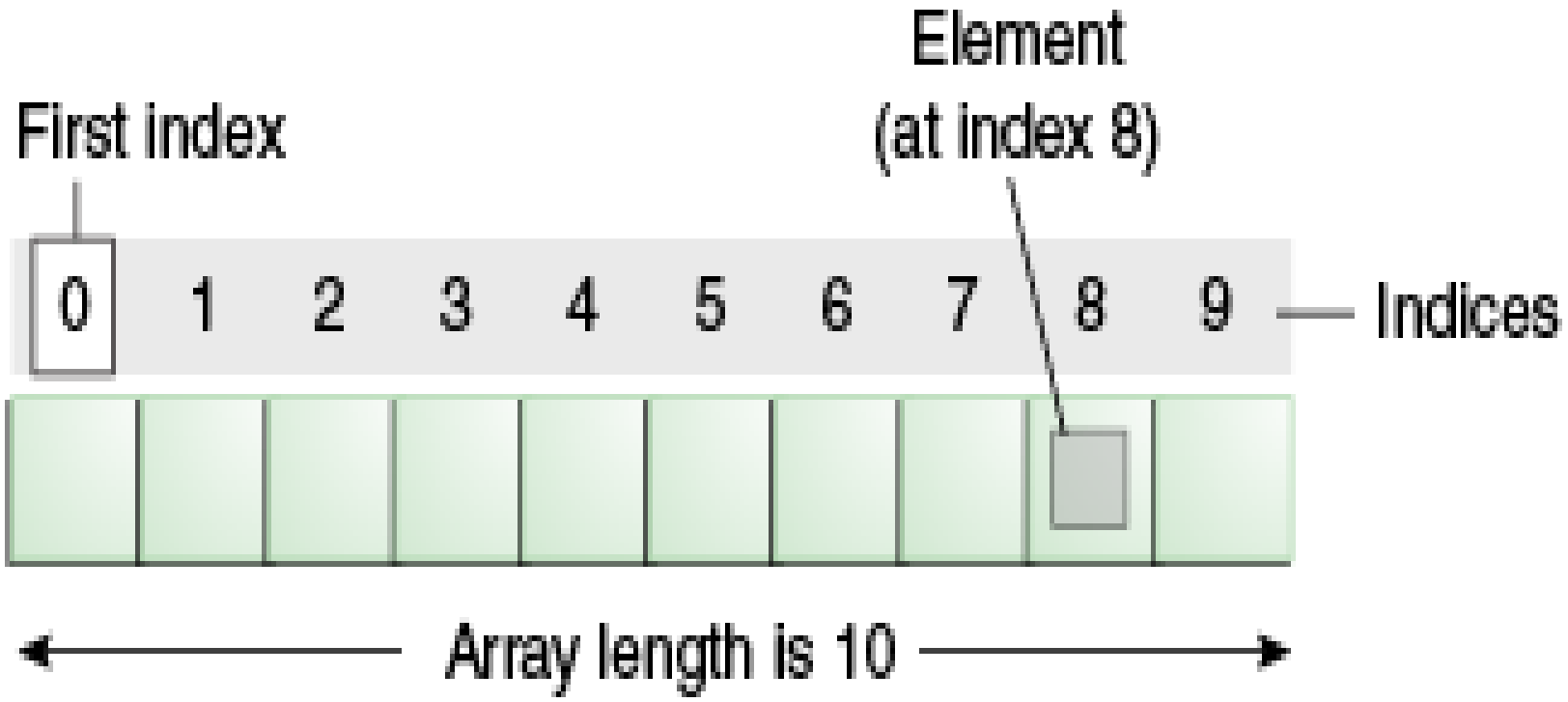
ARRAYs

Arrays


Element
(at index 8)
First index
0
1
2
3
4
5
6
7
8
9
Indices
Array length is 10

Arrays

One dimensional array

To create an array we need to perform two steps

1) declare the array

2)allocate space for the elements

1)type varName[];

ex: int a[];

2) varName = new type[size];

a = new int[10];

These two steps may be combined into one java statement as

```
type    varName[]  =    new  type[size];
int      a[]  = new int [10];
```

Accessing array: a[2] = 10;

Starts from zeroth position.

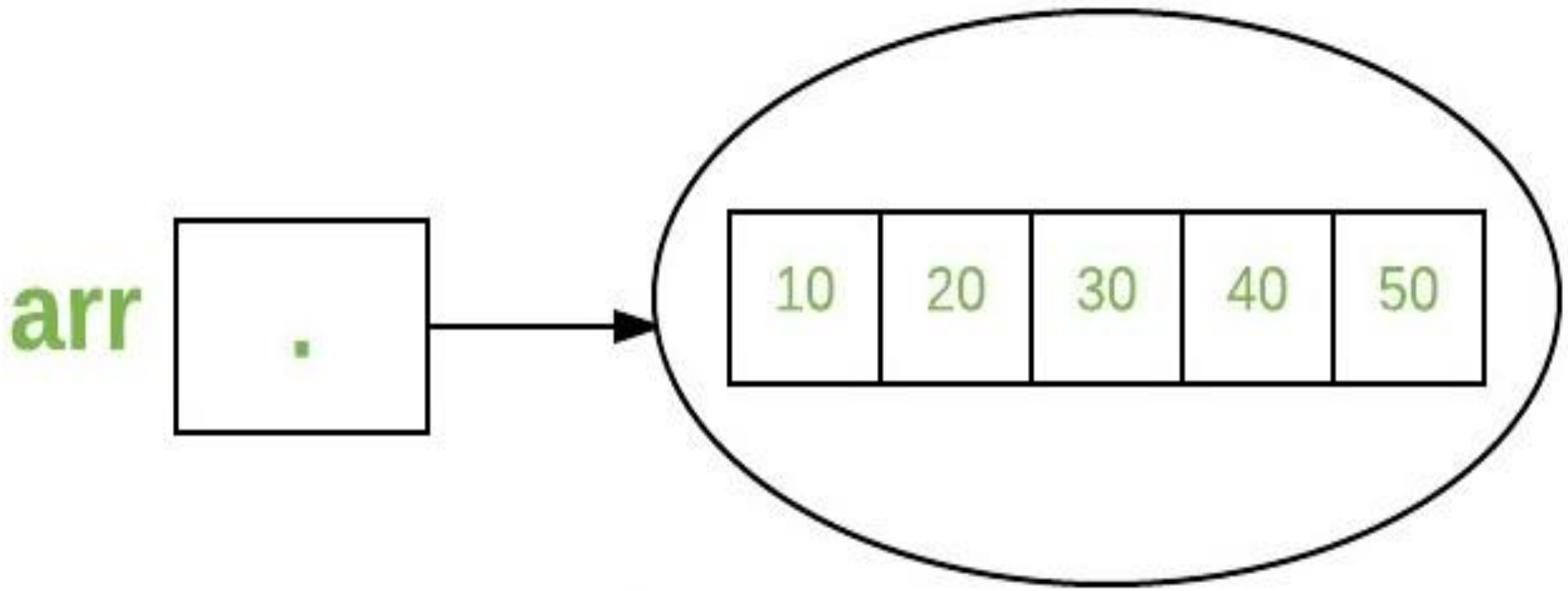
Size of the array can be obtained via the following expression.

varName.length

Abbreviated syntax to declare, allocate and initialize the elements of an array.

type varName[] = { e0, e1, ……en };

Here new is not used and sufficient memory is automatically provided.



One-Dimensional Array

We may assign array variable to another however, array itself is not physically copied.

Instead the variable on the left side of the assignment will simply refer to the same Array.

```
int j[] = { 0,1,2,3,4,5};
int  k[];
k=j;
```

```
class   FirstProgram
{
    public static void main(String args[])
    {
        int  a[]= {10,20,30};
        System.out.println(a[0]);
        System.out.println(a.length);

    }
}
```

```
class   FirstProgram
{
    public static void main(String args[])
    {
        int b[] = new int[10];
         b[0]=5;
         b[1]=6;
         System.out.println(b[0]);
         System.out.println(b.length);
    }
}
```

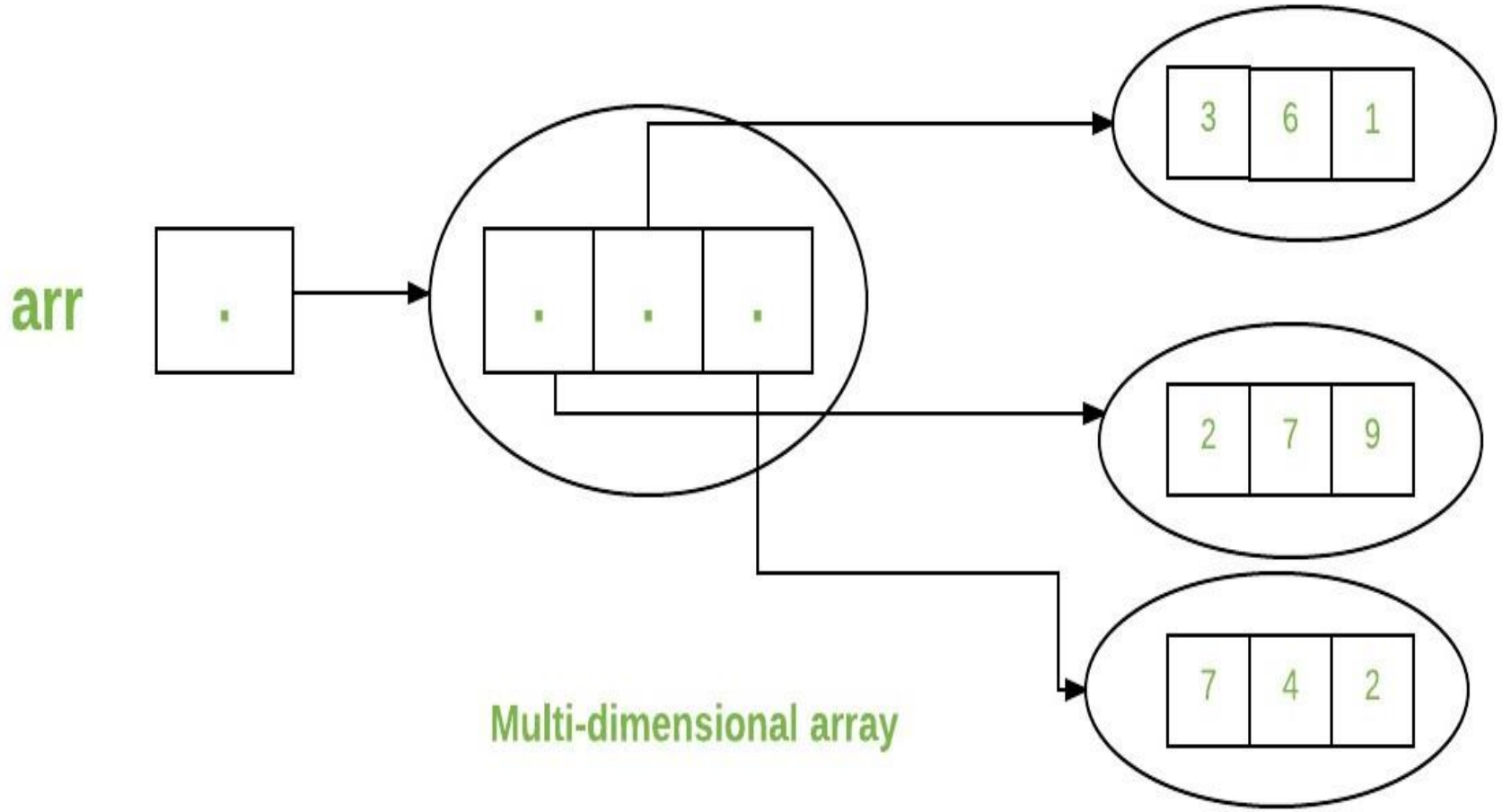
```
class   FirstProgram
{
    public static void main(String args[])
    {
         int b[] = new int[10];
          b[0]=5;
          b[1]=6;
          System.out.println(b[0]);
          System.out.println(b.length);

    }
}
```

Output: 5, 10

```
class   FirstProgram
{     public static void main(String args[])
    {
          int  a[]= {10,20,30};
          int b[];
          b= a;

          System.out.println("array a = "+a[0]+ " "+a[1]+"  "+ a[2]);
          System.out.println("array b=  " + b[0]+ " "+b[1]+"  " +b[2]);

          b[1] = 50;


           System.out.println("array a = "+a[0]+ " "+a[1]+"  "+ a[2]);
          System.out.println("array b=  " + b[0]+ " "+b[1]+"  " +b[2]);
    } }
```

Write a java program to find largest element in an aray

Multi Dimensional Arrays



Multidimensional array

In java multidimensional arrays are implemented as arrays of arrays

1) Declaration

type varName[][];

ex: int a[][];

2)Allocating space

varName = new type[size1][size2];

ex: a = new int[10][10];

OR

type varName = new type [size1][size2];

int a[][] = new int[10][10];

The size of the multidimensional array can be obtained by

```
varName.length;
```

To obtain the length of individual array

```
varName[index].length
```

Declaration, allocation & initialization

```
type  varName[][] = { { e00, e01,..e0x}, {e10,e11,..e1y},….{e20,e21,…e2z}};

int    a[][] = { {10,20,30}, { 4,5}, {1,2,3,4}};
```

```
class   FirstProgram
{
    public static void main(String args[])
    {
         int  a[][]={{10,20,30,40}, { 3,4}, {5,6,7,8,9}};

        System.out.println(a[1][1]);
         System.out.println(a.length);
         System.out.println(a[0].length);
         System.out.println(a[1].length);
         System.out.println(a[2].length);

    }
}
```

```
System.out.println("Enter row and column");
m = sc.nextInt();
n = sc.nextInt();
```

Type casting

- Process of converting one data type to another is called casting.
- 4 integer types can be cast into any other type except boolean.
- Float and double can be cast to any other type except boolean.
- Casting a floating point value to an integer will result in a loss of the fractional part.

Below shows which are guaranteed to result in no loss of information.

From	To
byte	short, char, int, long, float, double
short	int, long, float, double
char	int, long, float, double
int	long, float, double
long	float, double
float	double

Automatic conversion : assigning the value of one type to a variable of different type without a cast.

```
byte  b = 75;
int a = b;
```

The process of assigning a smaller type to a larger one is known as **widening** or **promotion.**

Explicit conversion(narrowing)

Assigning a larger type to a smaller one is known as **narrowing**.

type 1 var1 = (type2) var2

Ex: int a;

double b =3.4;

a= (int) b;

Type conversions in expressions

- During evaluation of the expression, if the operands are of different types, lower type is automatically converted to higher type before the operation proceeds.

- Whenever a **char, byte** or **short** is used in an expression, its value is automatically converted to i**nt** during the evaluation of that expression called **integral promotion**

- After the automatic integral promotions are applied, the java compiler converts all operands “up” to the type of the largest operand. This is called type promotion.

It is on an operation-by-operation basis as described in the following type-conversion algorithm.

I**F** either operand is a double

THEN the other operand is converted to double

ELSE IF either operand is float

THEN the other operand is converted to float

ELSE IF either operand is long

T**HEN** the other operand is converted to long.

- The final result of an expression is converted to the type of the variable on the left of the assignment sign before assigning the value to it.

- The following changes are introduced during the final assignment.

  1 float to int causes truncation of fractional part.
  2 double to float causes rounding of digits.
  3 long to int causes dropping of the excess higher order bits.

```
class Promote
{       public static void main(String args[])
        {
            byte b = 42;       char c = 'a';
          short s = 1024;    int i = 50000;
          float f = 5.67f;    double d = .1234;

          double result = (f * b) + (i / c) - (d * s);
          System.out.println("result = " + result);
      }}
```

- First subexpression, f * b, b is promoted to a float and the result of the subexpression is float.
- In the subexpression i / c, c is promoted to int, and the result is of type int.
- Then, in d * s, the value of s is promoted to double, and the type of the subexpression is double.
- Finally, these three intermediate values, float, int, and double, are considered.
- The outcome of float plus an int is a float. Then the resultant float minus the last double
- is promoted to double, which is the type for the final result of the expression.

```
byte a=2, d=3
short c;
c= a+d;
```

Error: found: int required short.

Correct:

```
byte a=2, d=3
short c;

c= (short)(a+d)
```

```
byte b;
int i=258; //1 0000 0010

b = (byte)i;
System.out.println(b);
```

OUTPUT: 2

```
float f=23.99f
int i=(int)f;
System.out.println(i);
```

```
float f=23.99f
int i=(int)f;
System.out.println(i);
```

OUTPUT: 23

```
 byte b1 =1;
byte  b2 = 2;
byte b3 = b1 * b2;
System.out.println(b3);
```

ERROR

```
byte b=50;
short  t;
   t = b
```

NO ERROR

```
byte b=50;
short t;
   t = b - 1
```

ERROR : found : int
required : short

```
int  n=386; // 1 1000 0010
byte b;
b = (byte)n;
System.out.println(b);
```

```
int  n=386; //Output : -126

byte b;

b = (byte)n;
System.out.println(b);
```