Chapter 5

More Data Types and operators

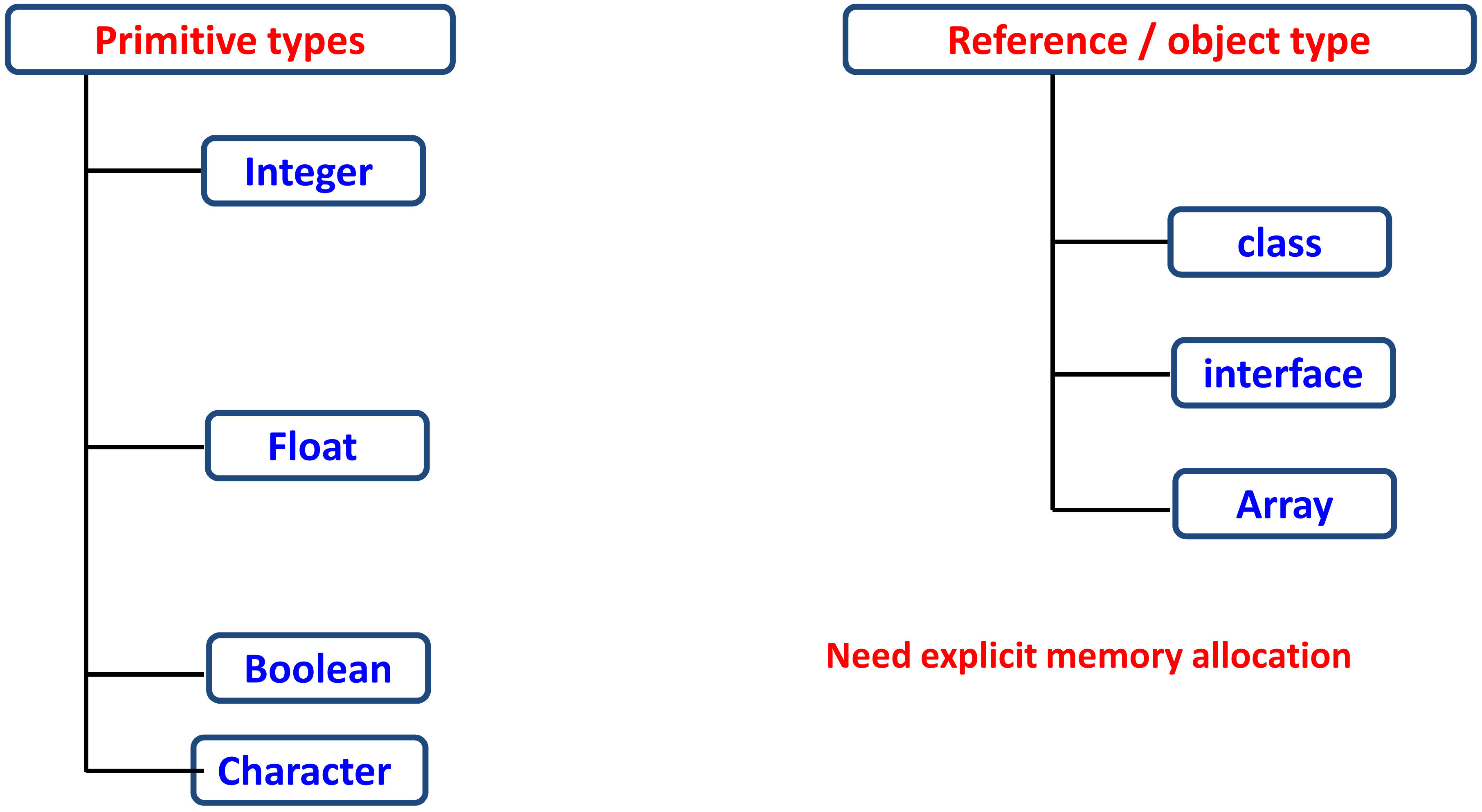
More Data Types and Operators

- Arrays
- Multidimensional Arrays
- Alternative Array Declaration Syntax
- Assigning Array References
- Using the Length Member
- The For-Each Style for Loop
- Strings
- The Bitwise operators.

INTRODUCTION

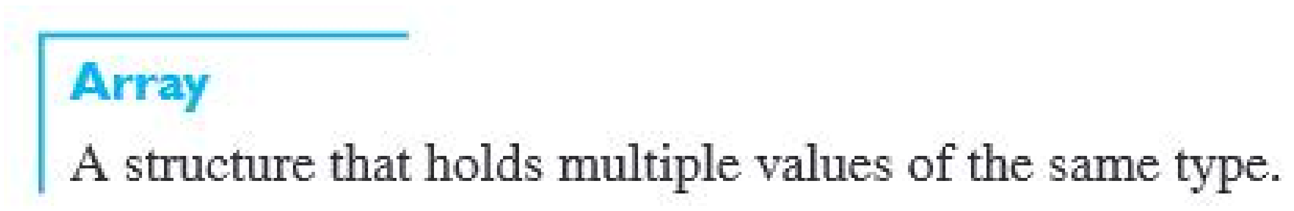
- Arrays
    1. Syntax, memory allocation
    2. one-dimentional and Multi-dimentional array
    3. Using the length member
    4. for-each loop
    5. Jagged Arrays

Data types revisited



Need explicit memory allocation

1. Syntax



Array

A structure that holds multiple values of the same type.

- Features
  - Array is an object

1. Syntax

- Creating reference of array

```
<Datatype> [ ]  <referenceName> ;
```

❑ **Example:**

```
int [ ]  price;          // java style
float  salary [ ] ;      // C style
```

Has No Memory

1. Syntax

- Allocating Memory to array
    - new keyword allocates memory for array

❑ **Example:**

```
int [ ]  price  =  new  int [10];       // Initializing memory

float  [ ] salary ;                     // No memory
salary = new  float[6];                 // Assigning memory
```

Exceptions in Array

- Legal indexes:

  between 0 and the array's length – 1

- Example

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

System.out.println( data[0] );      // okay
System.out.println( data[9] );      // okay
System.out.println( data[-1] );     // exception
System.out.println( data[10] );     // exception
```

ArrayIndexOutOfBoundsException

for loop with Array

- Initialization

```
int [ ] arr = { 10, 20, 30 };
```

- *for loop*

```
for ( int m=0; m<3; m++ )
  System.out.println(arr[m]);
```

for loop with Array

- Initialization

```
int [ ] arr = { 10, 20, 30 };
```

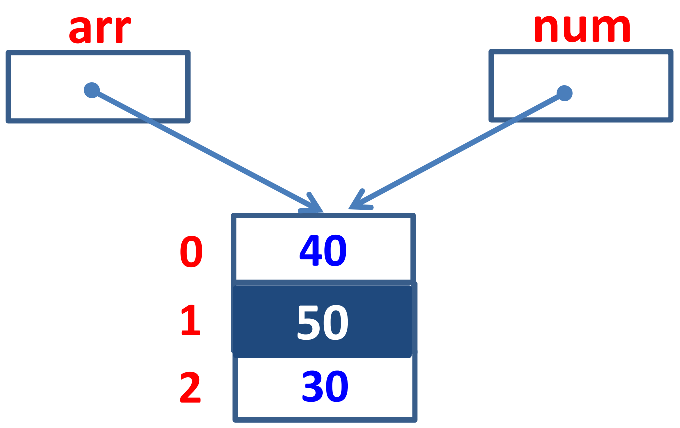
- *Using Array member: length*

```
for ( int m=0; m<arr.length; m++ )
 System.out.println( arr[m] );
```

Array copy…

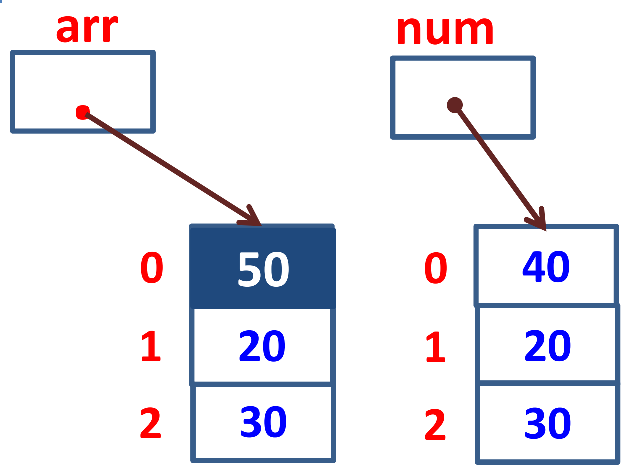
```
int  arr [] = {40, 20, 30};
int  num[] = arr;
arr[1] = 50;
System.out.println( “arr[1] =”  + arr[1]  );
System.out.println( “num[1] =”  + num[1]  );
```

- Example



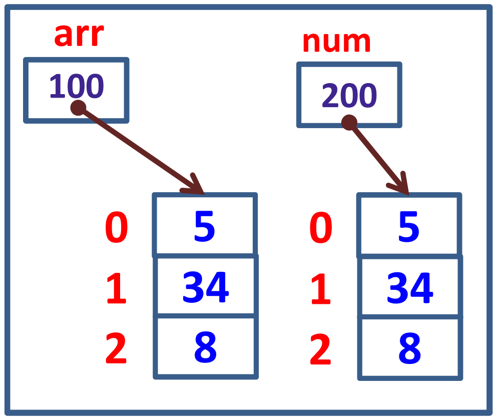
Using clone() method of array

```
int  arr [] = {40, 20, 30};
int  num[] = arr.clone();
arr[0] = 50;
System.out.println( “arr[0] =”  + arr[0]  );
System.out.println( “num[0] =”  + num[0]  );
```



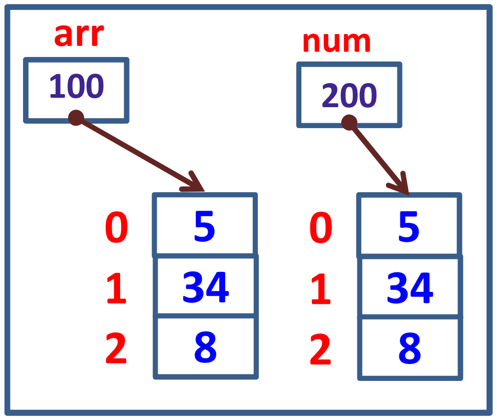
Testing for equality..

```
class CheckEquality {
            public static void main (String [] arg) {
                int  []  arr = { 5, 34, 8 } ;
                int  []  num = { 5, 34, 8 } ;

                if (  arr == num )
                            System.out.println( “ Both Are Equal ”) ;
                else
                            System.out.println(“ We are Different ”) ;
      }
}
```



Using Arrays class..

```
import  java.util.Arrays;
class CheckEquality {
            public static void main (String [] arg) {
                int  []  arr = { 5, 34, 8 } ;
                int  []  num= { 5, 34, 8 } ;

                if ( Arrays.equals(arr, num) )
                        System.out.println( “ Both Are Equal ”) ;
                else
                        System.out.println(“ We are Different ”) ;
    }
}
```



Default Initial Values

- When an array is instantiated, default values of single-dimensional arrays are

Array data type	**Default value**
byte, short, int, long	0
float, double	0.0
char	space
boolean	*false*
Any object reference (for example, a *String*)	*null*

Example

- Find biggest element in an array..

```
large= arr[i]
for(i=0; i<arr.length; i++)
{
        if(large<arr[i])
        {
                large=arr[i];
        }
}
```

Using for-each..

- ❑ Consider the following array

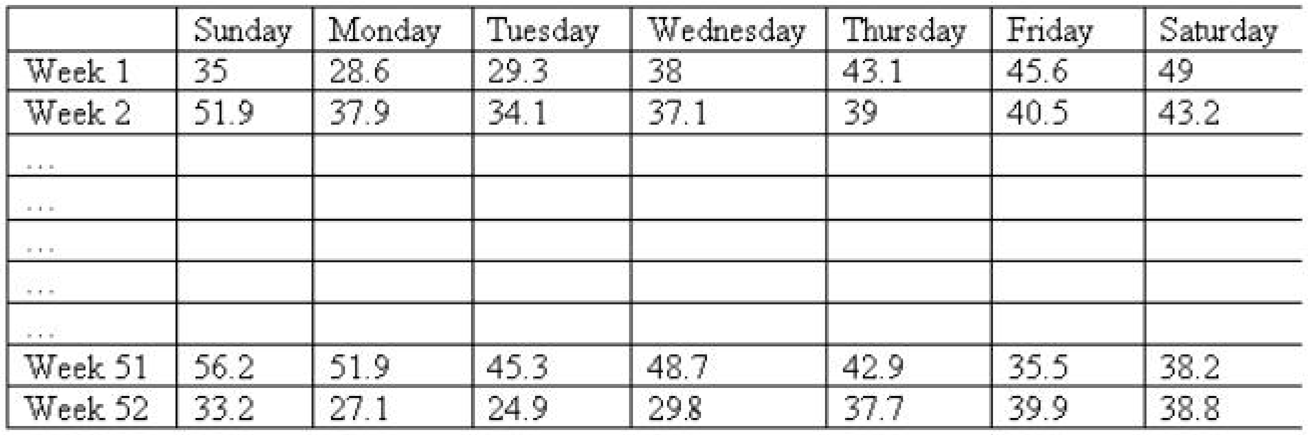
```
double [ ]   price = { 40, 20, 30 };
```

- ❑ *Using normal for loop*

```
for ( int m=0; m<price.length; m++ )
    System.out.println ( price[m] );
```

Multi Dimension Array..

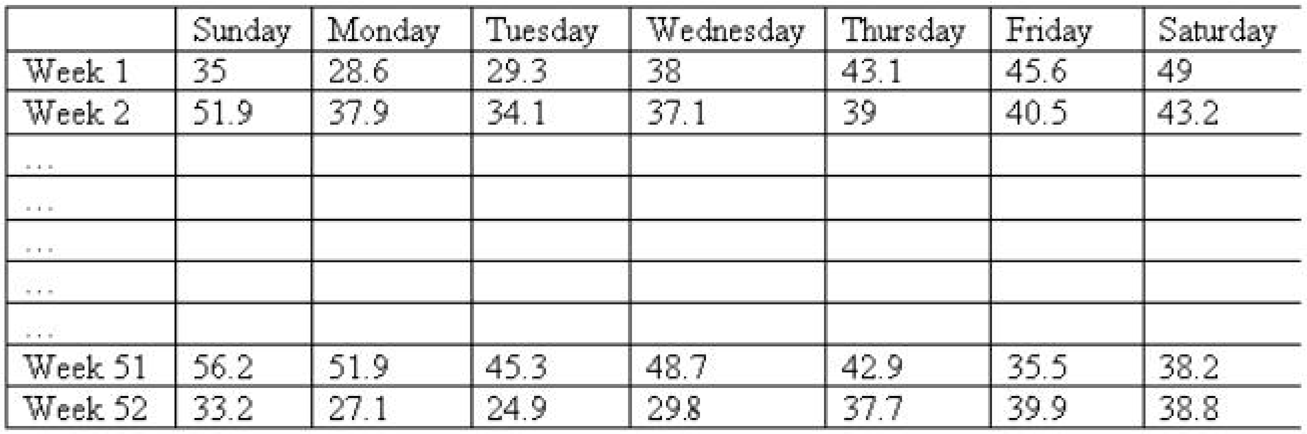
- 2-D Array
  - allow organization of data in rows and columns in a table-like representation.



	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Week 1	35	28.6	29.3	38	43.1	45.6	49
Week 2	51.9	37.9	34.1	37.1	39	40.5	43.2
...							
...							
...							
...							
...							
Week 51	56.2	51.9	45.3	48.7	42.9	35.5	38.2
Week 52	33.2	27.1	24.9	29.8	37.7	39.9	38.8

Multi Dimension Array..

- 2-D Array
  - allow organization of data in rows and columns in a table-like representation.



	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Week 1	35	28.6	29.3	38	43.1	45.6	49
Week 2	51.9	37.9	34.1	37.1	39	40.5	43.2
...							
...							
...							
...							
...							
Week 51	56.2	51.9	45.3	48.7	42.9	35.5	38.2
Week 52	33.2	27.1	24.9	29.8	37.7	39.9	38.8

2-Dimension Array

An Array of Arrays

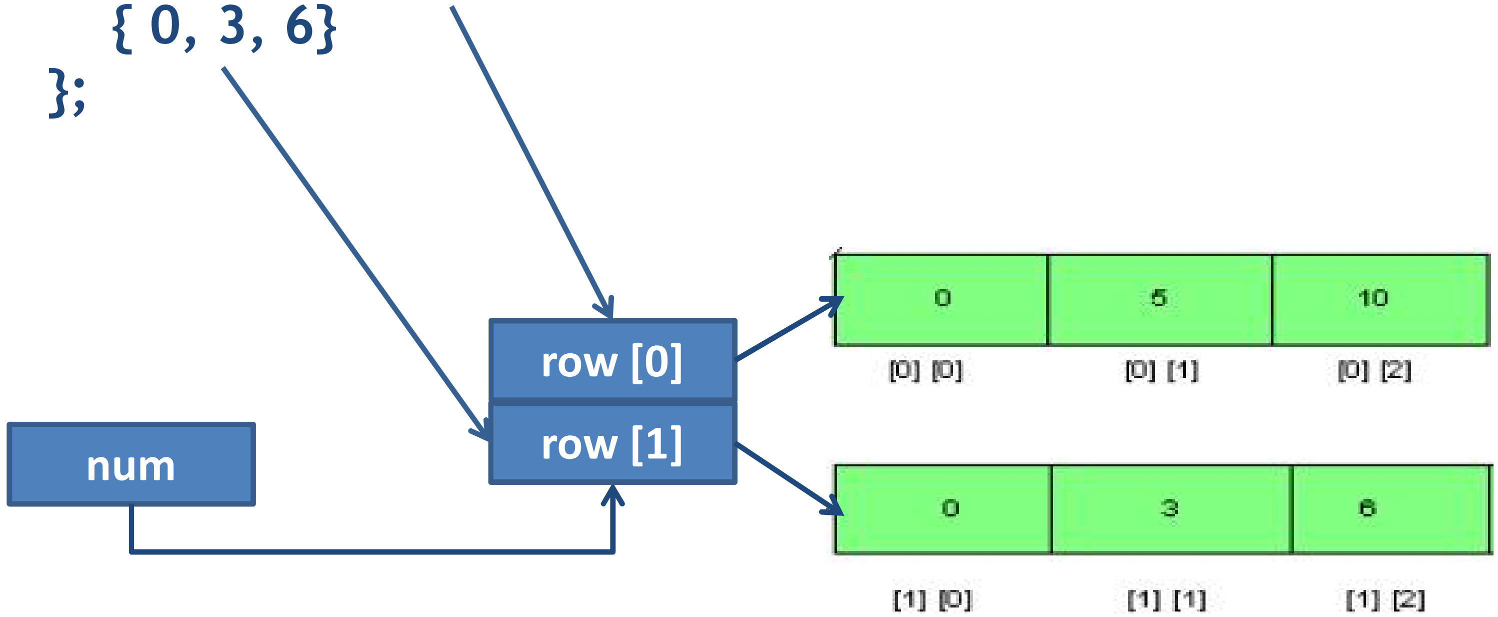
- Declaring a 2-dimensional array
  - datatype [][] arrayName;

    or
  - datatype [][] *arrayName1, arrayName2, ...;*
- Instantiating 2-dimensional array: example
  - int [][] matrix = new int[3][3];
  - float [][] price = new float[2][4];

Initializing 2-D Array:

An Array of Arrays

```
int num [] [] = {          {0, 5, 10},
                   { 0, 3, 6}
                 };
```

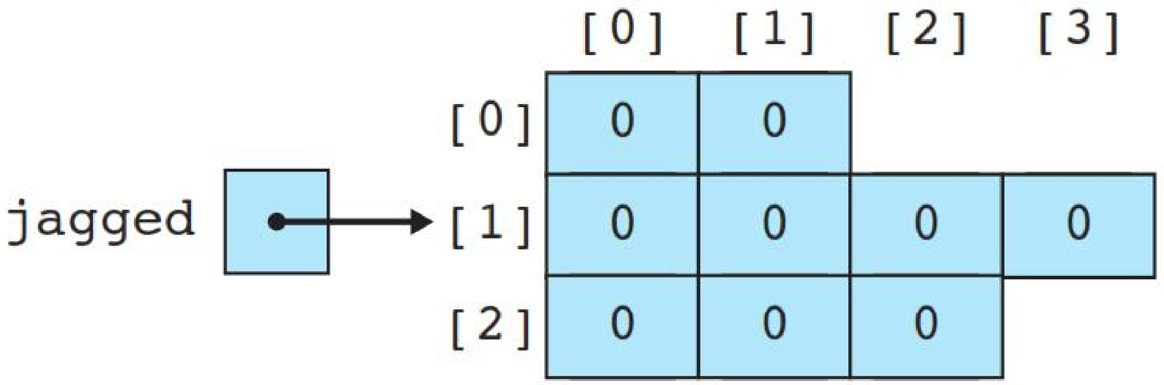


Finding biggest element using for each..

```
class Big {
        public static void main (String [] arg) {
                int  [] myArr = {  45, 5, 34, 8 } ;
                int  big = myArr[0];

                for( int num : myArr)
                        if (  big<num )
                                big = num;

                System.out.println(“Bigest = ” + big) ;
   }
}
```

Jagged Array..

- Consider following structure of array

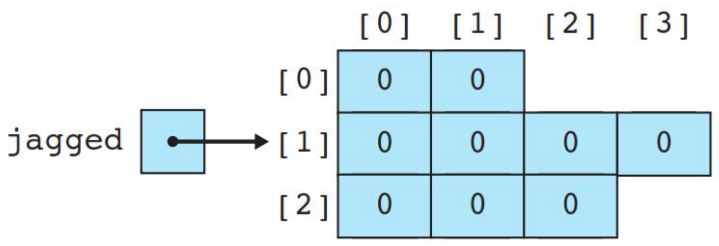


- Each row can have different number of columns

Jagged Array..

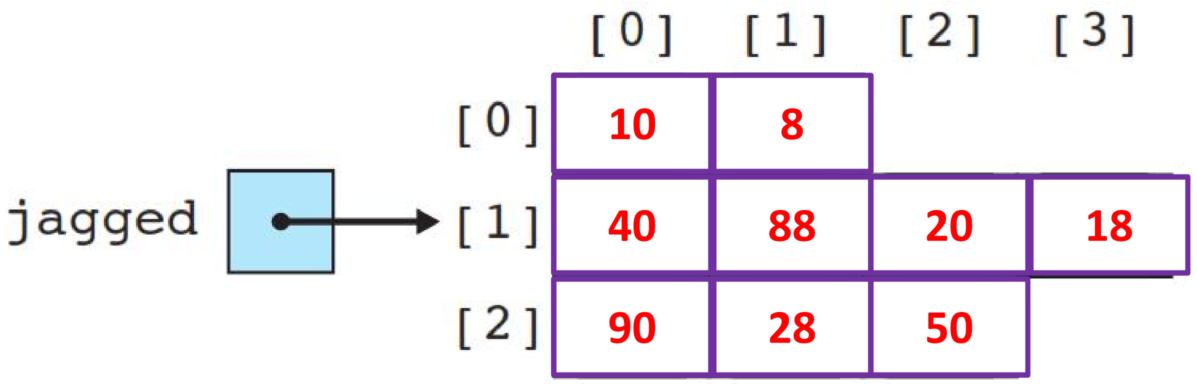
- Creating jagged array

```
int  [][] jagged = new int [3] [ ] ;

jagged[0] = new int [2] ;

jagged[1] = new int [4] ;

jagged[2] = new int [3] ;
```



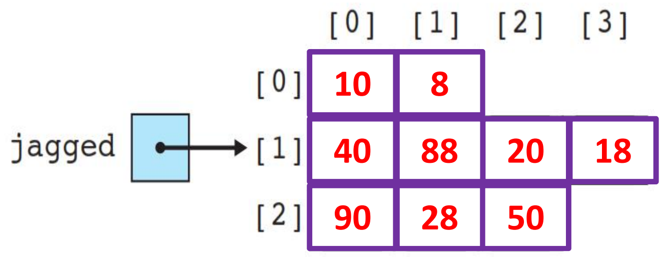
Jagged Array..

- Application of Jagged Arrays
  - Pascal's Triangle
- Write a program to initialize jagged array with following values.
- Also find the biggest element in this array.



Finding biggest in jagged array..

```
class CheckEquality {

    public static void main (String [] arg) {
        int  [][]  jagged =   {  { 10, 8 }, { 40, 88, 20, 18 }, { 90, 28, 50 }   } ;

        int  big = jagged[0][0];

            for (  int  row = 0 ; row < jagged.length ; row++)

                for (  int  col = 0 ; col < jagged[row].length ; col++)

                    if (  big< jagged[row][col] )

                        big  = jagged[row][col];

        System.out.println("Biggest = " + big) ;

    }
}
```



Bitwise Operators

- Converting uppercase ad lowercase using bitwise

```
class Lowercase{
		public static void main(String[] args){
				char ch;
		for(int i= 0 ; i < 10 ; i++){
		ch=(char) (‘A’ + i);
		System.out.println(ch);

		ch = (char) ((int) ch | 32);
		System.out.println(ch + “ “);
		}
	}
}
```

```
Class ArrayDemo{
    public static void main(String [] args){
    int [] arr = new int[10];
    int i;
    for(i=0; i<10; i++){
          arr[i] =i;
          System.out.println(“Array contains ” +arr[i]);
          }
    }
}
```

```
class JavaArrayLengthTest
{
    public static void main(String[] args) {
        String[] testArray = { "Apple", "Banana", "Carrots" };
    int arrayLength = testArray.length; System.out.println("The length of the array is: " + arrayLength); } }
```

```
class UpCase {
 public static void main(String args[]) {
   char ch;
     for(int i=0; i < 10; i++)
     {
               ch = (char) ('a' + i);
               System.out.print(ch);
               ch = (char) ((int) ch & 65503);
               System.out.print(ch + " ");
      }
 }
```