Object Oriented Programming with Java 8
PG-DAC

Rohan Paramane

Agenda

- Array

Array Introduction

- Array, stack, queue, LinkedList are data structures.
- In Java, data structure is called collection and value stored inside collection is called element.
- Array is a sequential/linear container/collection which is used to store elements of same type in continuous memory location.

In C/C++

Static Memory allocation for array

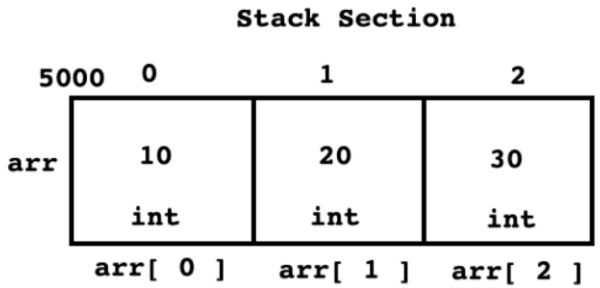
```
int arr1[ 3 ];    //OK

int size = 3;
int arr2[ size ];     //OK
```

In C/C++

Dynamic Memory allocation for array

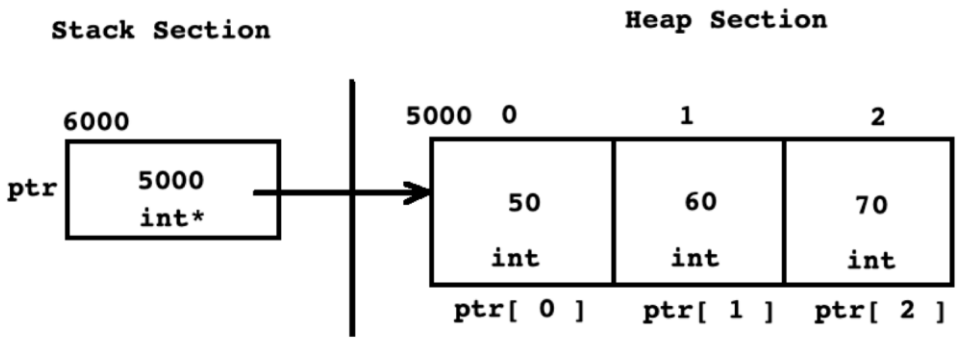
```
int *arr = ( int* )malloc( 3 * sizeof( int ));
//or
int *arr = ( int* )calloc( 3, sizeof( int ));
```

Static v/s Dynamic Memory Allocation In C/C++



[Static Memory Allocation]

```
arr[ 0 ]  =>  10
arr[ 1 ]  =>  20
arr[ 2 ]  =>  30
```



[Dynamic Memory Allocation]

```
ptr[ 0 ]  =>  50
ptr[ 1 ]  =>  60
ptr[ 2 ]  =>  70
```

Array Declaration and Initialization In C

- `int arr[ 3 ]; //OK : Declaration`
- `int arr[ 3 ] = { 10, 20, 30 }; //OK : Initialization`
- `int arr[ ] = { 10, 20, 30 }; //OK`
- `int arr[ 3 ] = { 10, 20 }; //OK : Partial Initialization`
- `int arr[ 3 ] = { }; //OK : Partial Initialization`
- `int arr[ 3 ] = { 10, 20, 30, 40, 50 }; //Not recommended`

Accessing Elements Of Array

- If we want to access elements of array then we should use integer index.
- Array index always begins with 0.

```
int arr[ 3 ] = { 10, 20, 30 };
printf("%d\n", arr[ 0 ] );
printf("%d\n", arr[ 1 ] );
printf("%d\n", arr[ 2 ] );
```

```
int arr[ 3 ] = { 10, 20, 30 };
int index;
for( index = 0; index < 3; ++ index )
    printf("%d\n", arr[ index ] );
```

Advantage and Disadvantages Of Array

- **Advantage Of Array**

  1. We can access elements of array randomly.

- Disadvantage Of Array

  1. We can not resize array at runtime.
  2. It requires continuous memory.
  3. Insertion and removal of element from array is a time consuming job.
  4. Using assignment operator, we can not copy array into another array.
  5. Compiler do not check array bounds(min and max index).

Array In Java

- Array is a reference type in Java. In other words, to create instance of array, new operator is required. It means that array instance get space on heap.
- **There are 3 types of array in Java:**
    1. Single dimensional array
    2. Multi dimensional array
    3. Ragged array
- **Types of loop in Java:**
    1. do-while loop
    2. while loop
    3. for loop
    4. for-each loop
- **To perform operations on array we can use following classes:**
    1. java.util.Arrays
    2. org.apache.commons.lang3.ArrayUtils(download .jar file)

Methods Of java.util.Arrays Class

Following are the methods of java.util Arrays class.(try javap java.util.Arrays)

- public static <T> List<T> asList(T... a)
- public static int binarySearch(int[] a, int key) //Overloaded
- public static int binarySearch(Object[] a, Object key)
- public static int[] copyOf(int[] original, int newLength)
- public static <T> T[] copyOf(T[] original, int newLength)
- public static int[] copyOfRange(int[] original, int from, int to)
- public static <T> T[] copyOfRange(T[] original, int from, int to)
- public static void fill(int[] a, int val)
- public static void fill(Object[] a, Object val)
- public static void fill(Object[] a, int fromIndex, int toIndex, Object val)
- public static void sort(int[] a) //Overloaded
- public static void sort(Object[] a)
- public static void parallelSort(int[] a)
- public static <T extends Comparable<? super T>> void parallelSort(T[] a)
- public static String toString(Object[] a) //Overloaded
- public static String deepToString(Object[] a)
- public static IntStream stream(int[] array)   //Overloaded
- public static <T> Stream<T> stream(T[] array)

Single Dimensional Array

Reference declaration

```
int arr[ ]; //OK
int [ arr ]; //NOT OK
int[ ] arr; //OK
```

Instantiation

```
int[ ] arr1 = new int[ 3 ];
//or
int size = 3;
int[ ] arr2 = new int[ size ];
```

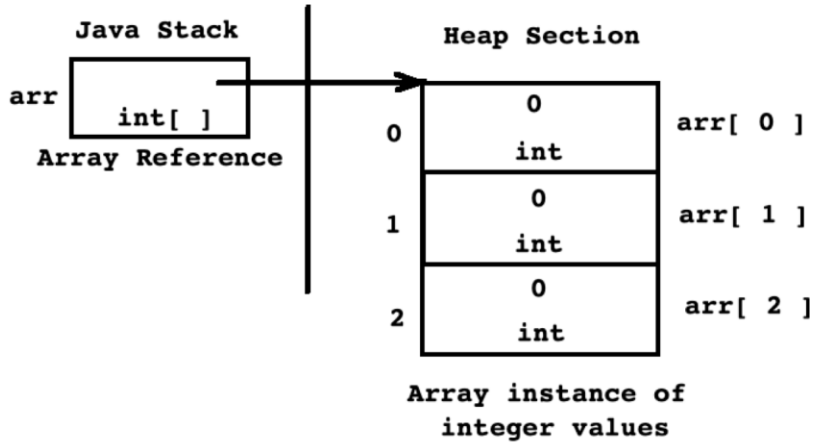
```
int[] arr1 = new int[ -3 ];  //NegativeArraySizeException
//or
int size = -3;
int[] arr2 = new int[ size ];    //NegativeArraySizeException
```

Initialization

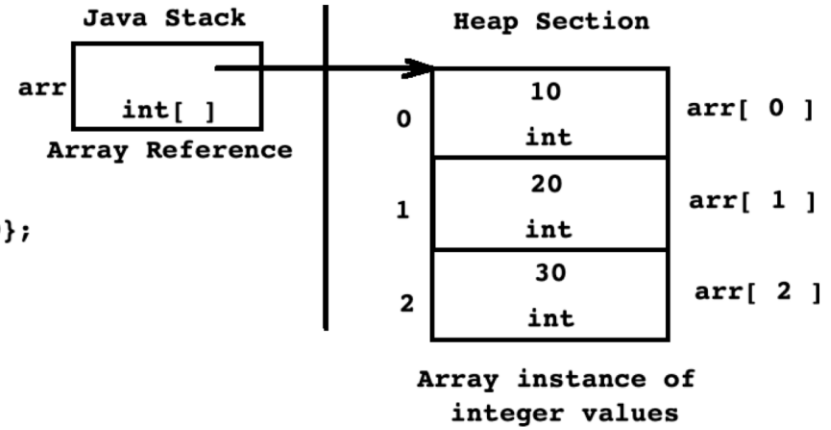
```
int[] arr = new int[ size ]{ 10, 20, 30 };  //Not OK
int[] arr = new int[  ]{ 10, 20, 30 };  //OK
int[] arr = { 10, 20, 30 };  //OK
```

Single Dimensional Array

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




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



Using length Field

```
public class Program {
    public static void printRecord( int[] arr ) {
        for( int index = 0; index < arr.length; ++ index )
            System.out.print( arr[ index ] +"    ");
        System.out.println();
    }
    public static void main(String[] args) {
        int[] arr1 = new int[   ] { 10, 20, 30 };
        Program.printRecord(arr1);

        int[] arr2 = new int[   ] { 10, 20, 30, 40, 50 };
        Program.printRecord(arr2);

        int[] arr3 = new int[   ] { 10, 20, 30, 40, 50, 60, 70   };
        Program.printRecord(arr3);
    }
}
```

ArrayIndexOutOfBoundsException

- Using illegal index, if we try to access elements of array then JVM throws ArrayIndexOutOfBoundsException. Consider following code:

```
public static void main(String[] args) {
    int[] arr = new int[  ] { 10, 20, 30, 40, 50 };
    //int element = arr[ -1 ]; //ArrayIndexOutOfBoundsException
    //int element = arr[ arr.length ]; //ArrayIndexOutOfBoundsException
    //int element = arr[ 7 ]; //ArrayIndexOutOfBoundsException
}
```

ArrayStoreException

- If we try to store incorrect type of object into array then JVM throws ArrayStoreException.
- Consider the following code:

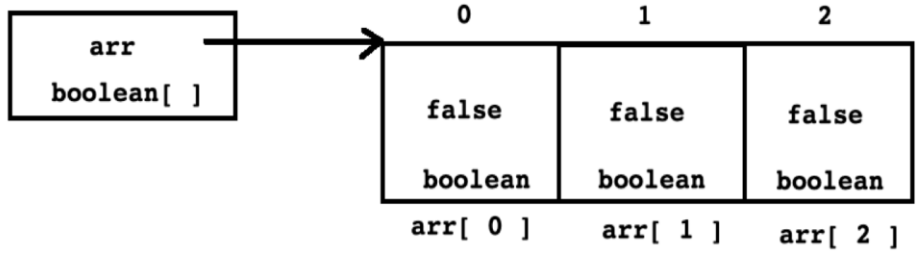
```
public class Program {
    public static void main(String[] args) {
        Object[] arr = new String[ 3 ];
        arr[ 0 ] = new String("DAC");    //OK
        arr[ 1 ] = "DMC";    //OK
        arr[ 2 ] = new Integer(123);    //Not OK : ArrayStoreException
    }
}
```

Array Of Primitive Values

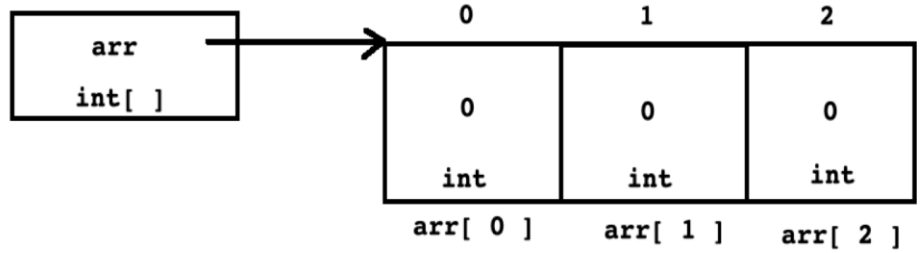
```
public class Program {
    public static void main(String[] args) {
        boolean[] arr = new boolean[ 3 ];   //contains all false
        int[] arr = new int[ 3 ];   //contains all 0
        double[] arr = new double[ 3 ]; //contains all 0.0
    }
}
```

Array Of Primitive Values

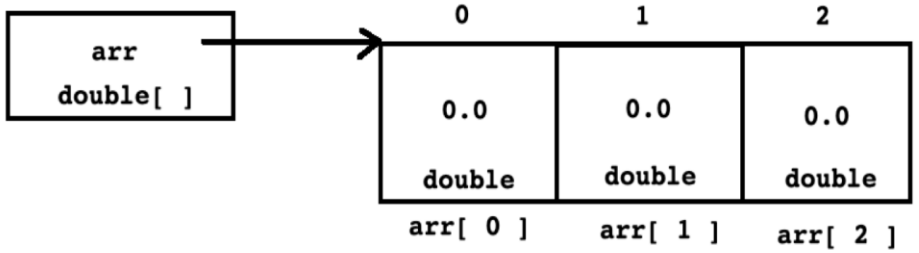
```
boolean[] arr = new boolean[3];
```




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




```
double[] arr = new double[3];
```



If we create array of primitive values then it's default value depends of default value of data type.

Array Of References

```
public class Program {
    public static void main(String[] args) {
        Date[] arr = new Date[ 3 ]; //Contains all null
    }
}
```

Array Of References and Instances

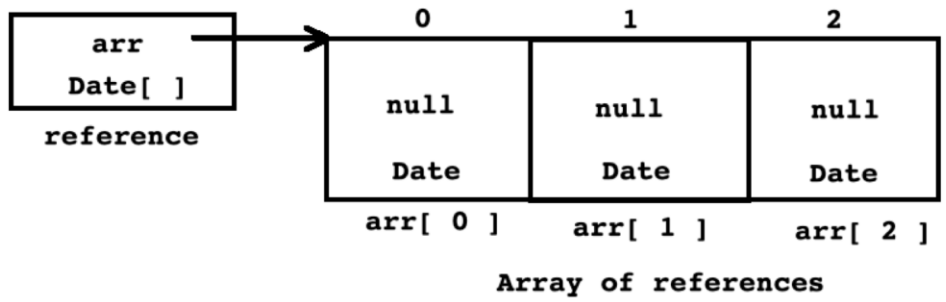
```
public class Program {
    public static void main(String[] args) {
        Date[] arr = new Date[ 3 ]; //Contains all null
    }
}
```

- Let us see how to create array of instances of non primitive type

```
public class Program {
    public static void main(String[] args) {
        Date[] arr = new Date[ 3 ];
        arr[ 0 ] = new Date( );
        arr[ 1 ] = new Date( );
        arr[ 2 ] = new Date( );
    }
    //or
    public static void main(String[] args) {
        Date[] arr = new Date[ 3 ];
        for( int index = 0; index < arr.length; ++ index )
        arr[ index ] = new Date( );
    }
}
```

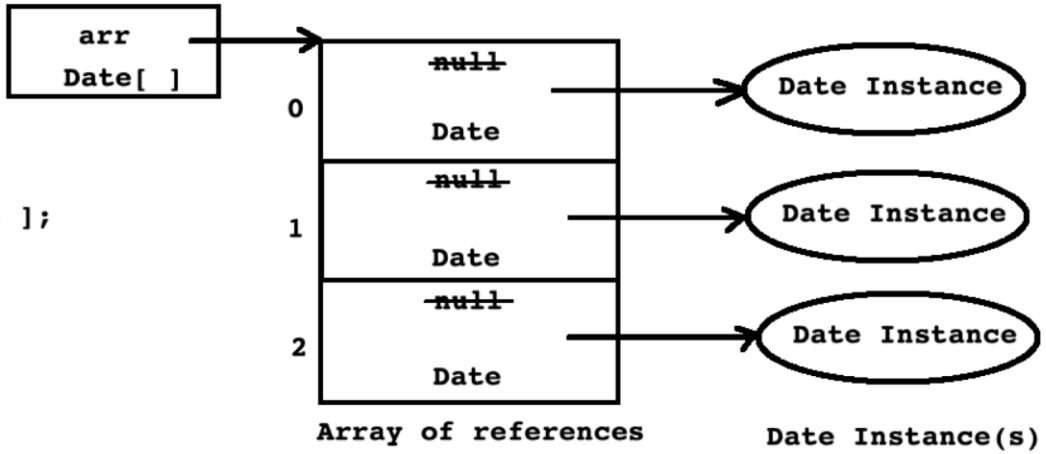
Array Of References and Instances

```
Date[ ] arr = new Date[ 3 ];
```



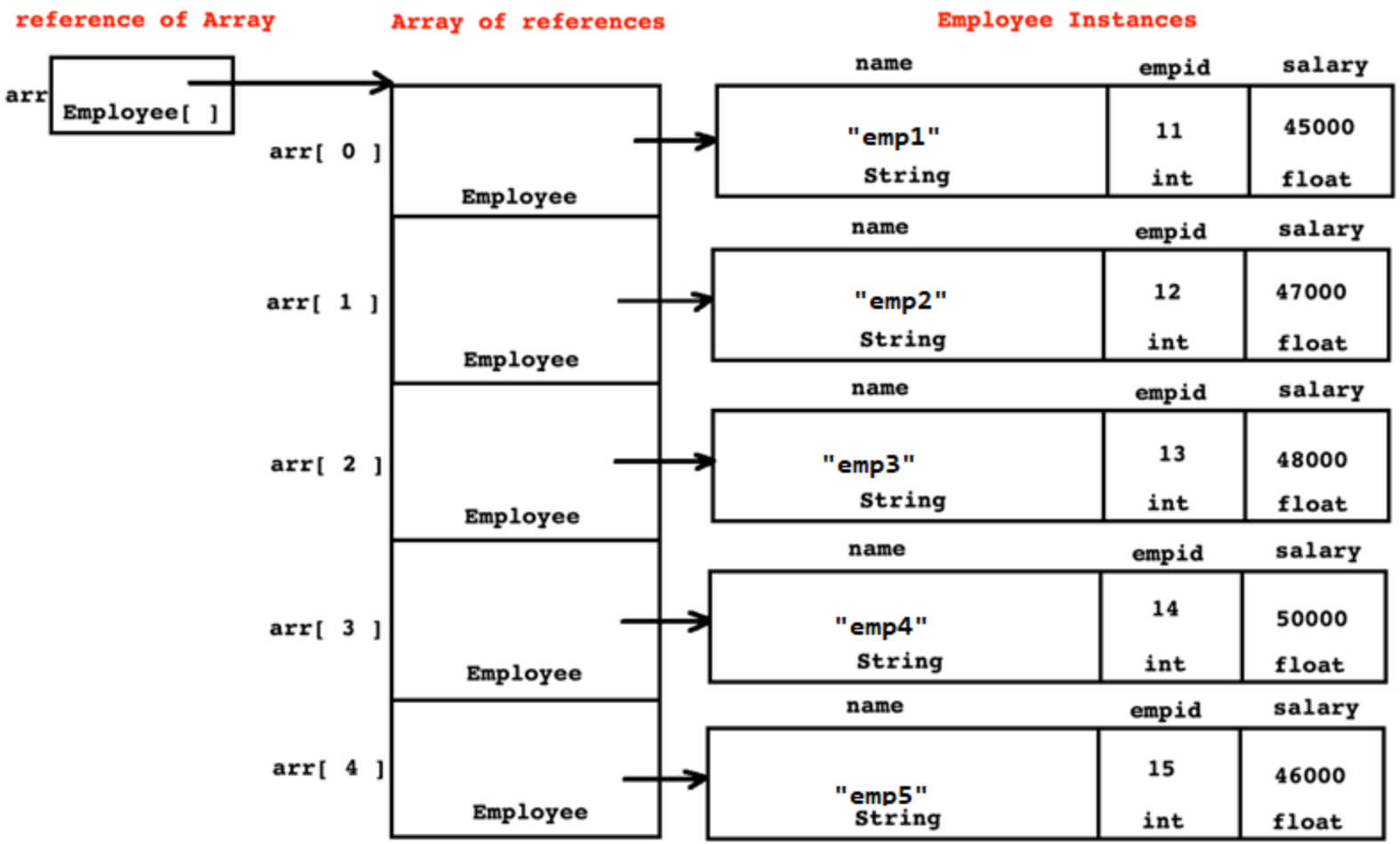
If we create an array of references then by default it contains null.

```
Date[] arr = new Date[ 3 ];
arr[ 0 ] = new Date( );
arr[ 1 ] = new Date( );
arr[ 2 ] = new Date( );
```



[Array Of Instances]

Array Of Instances



Multi Dimensional Array

- Array of elements where each element is array of same column size is called as multi dimensional array.

Reference declaration:

```
int arr[ ][ ];   //OK
int [ ]arr[ ]    //OK
int[ ][ ] arr;   //OK
```

Array Creation:

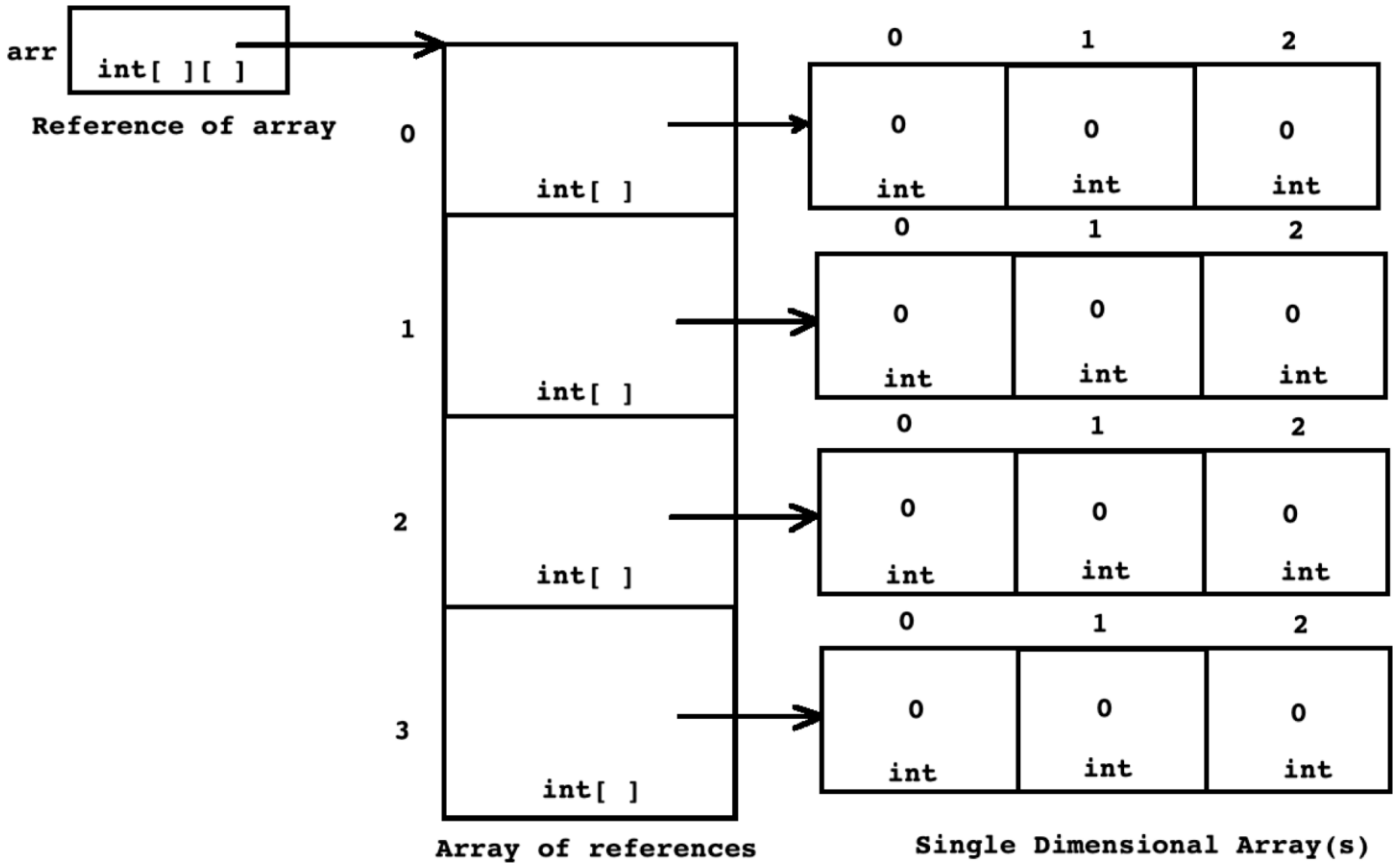
```
int[][] arr = new int[ 2 ][ 3 ];
```

Initialization

```
int[][] arr = new int[ ][ ]{{10,20,30},{40,50,60}}; //OK
int[][] arr = { {10,20,30}, {40,50,60} };    //OK
```

Multi Dimensional Array

+ Multi Dimensional Array



Ragged Array

- A multidimensional array where column size of every array is different.

Reference declaration

```
int arr[][];
int []arr[];
int[][] arr;
```

Array creation

```
int[][] arr = new int[3][];
arr[ 0 ] = new int[ 2 ];
arr[ 1 ] = new int[ 3 ];
arr[ 2 ] = new int[ 5 ];
```

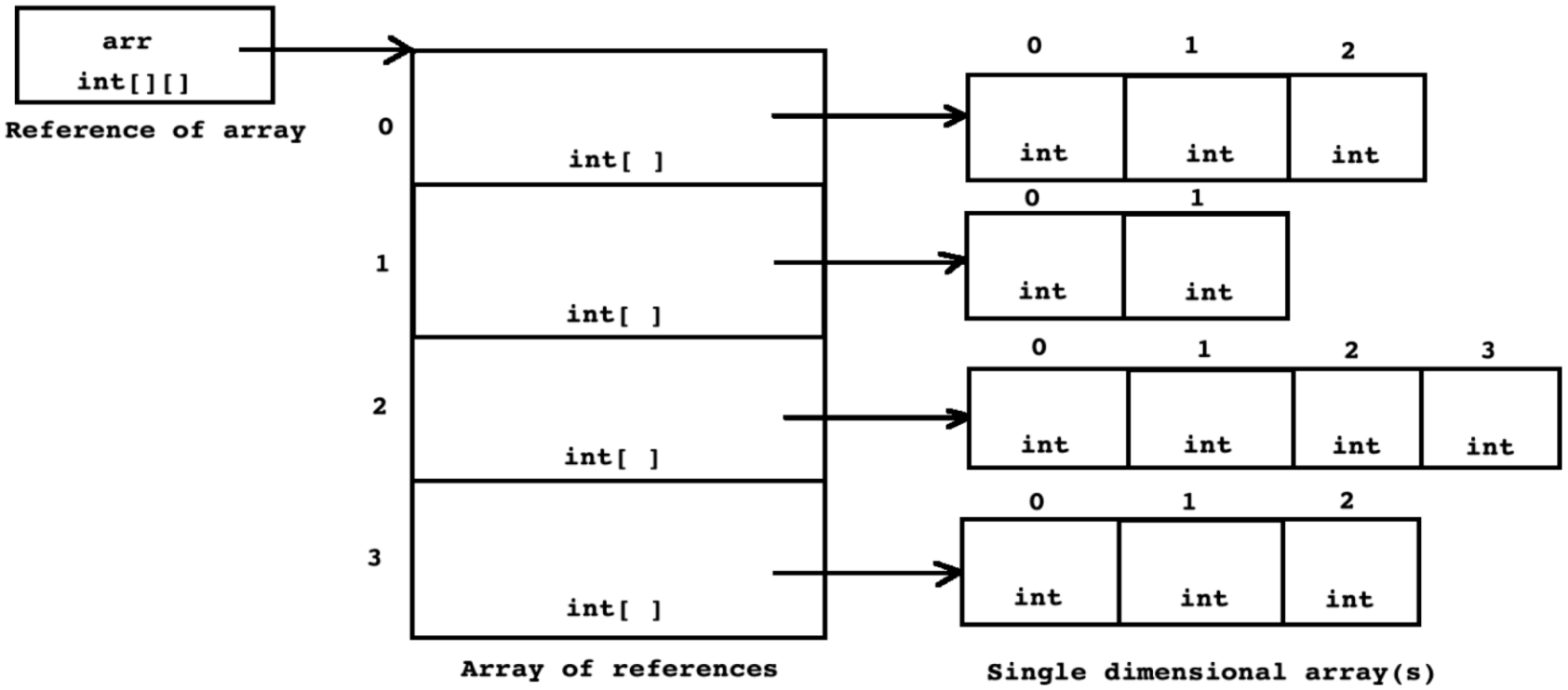
Array Initialization

```
int[][] arr = new int[3][];
arr[ 0 ] = new int[  ]{ 10, 20 };
arr[ 1 ] = new int[  ]{ 10, 20, 30 };
arr[ 2 ] = new int[  ]{ 10, 20, 30, 40, 50 };
```

```
int[][] arr = { { 1, 2 }, { 1, 2, 3 }, {1,2,3,4,5}};
```

Ragged Array

+ Ragged Array



Thank you.

Rohan.paramane@sunbeaminfo.com