Arrays

Java Arrays

- Arrays group objects or primitives of the same type together.
- In Java, an array is an object.
- Memory for the array reference is allocated on the stack.
- Memory for the array object is allocated dynamically on the heap.

Declaring Arrays References

- An array reference is declared as follows:

  element_type arr_ref_name [ ] ;

OR: *element_type[] arr_ref_name;*

- Examples:

```
char c_arr [ ] ;
Point p_arr [ ];          //p_arr is a null reference to an array
                                //of references to objects of
                                //class point.
Box boxArray [ ] ;
```

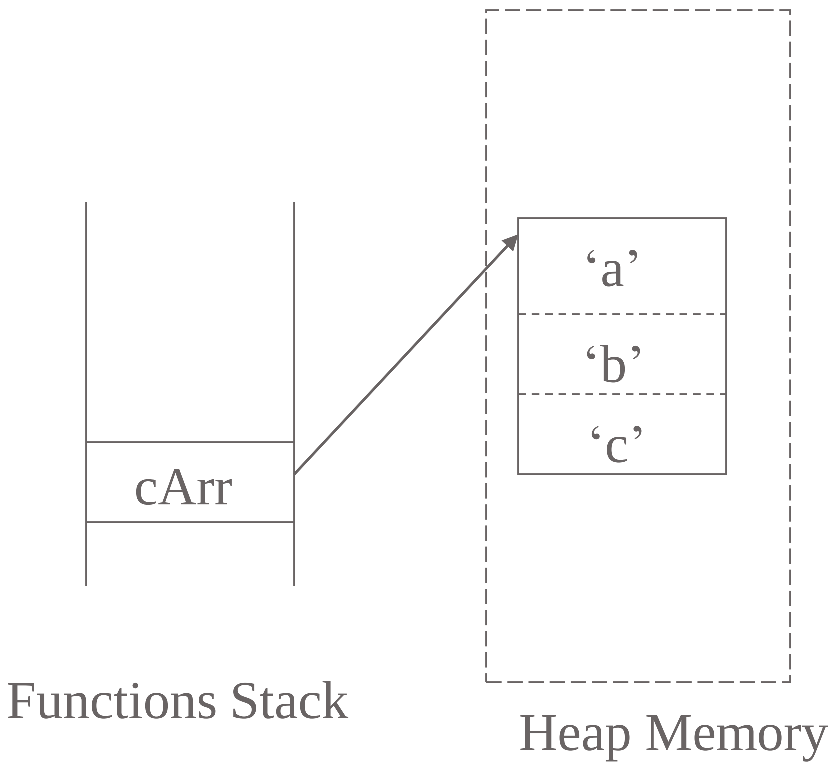
Creating Arrays

- An array, like any other object, is instantiated using the *new* keyword.
- Examples:

```
char c_arr[ ] = new char[100]; //This will create an (array)
                               //object that holds 100 chars.
int i_arr[ ] ;
…
i_arr = new int [MAX];
Point pArr[ ] = new Point [ 200 ]; //This will create an (array)
                                   //object that holds 200 null
                                   //references to objects of
                                   //class Point.
```

Primitives Arrays – Memory Allocation

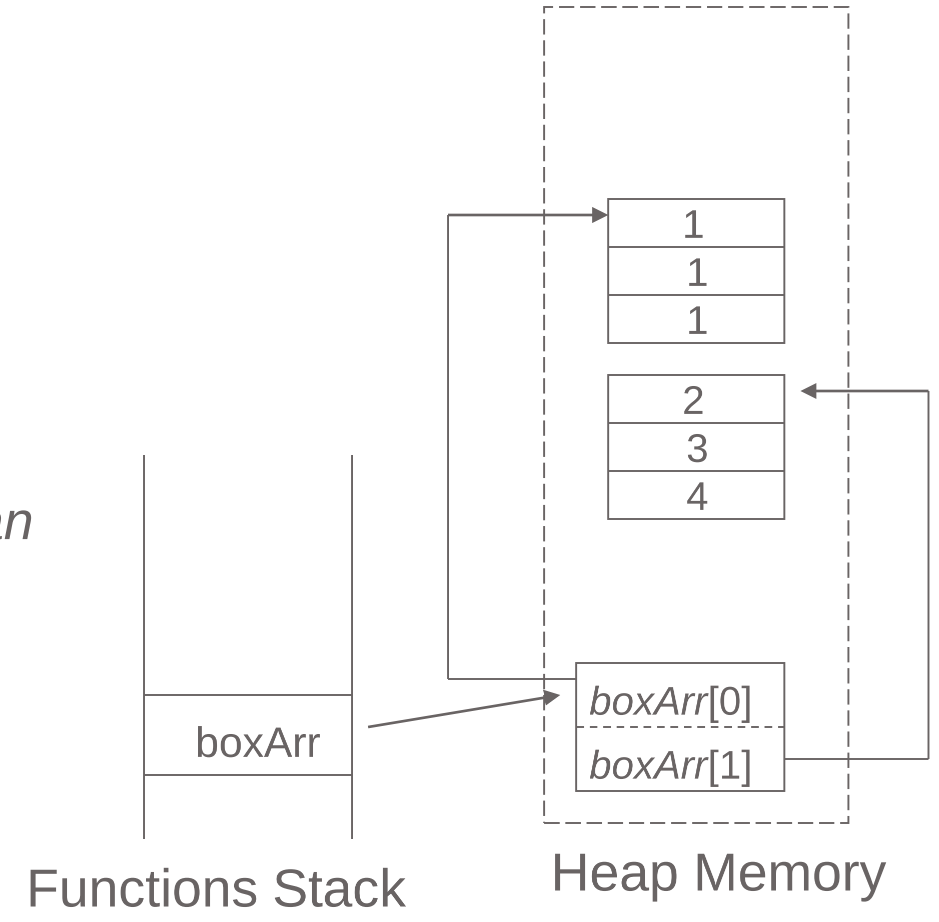
```
char cArr [ ] = new char [3] ;
cArr[0]=‘a’;
cArr[1]=‘b’;
cArr[2]=‘c’;
```



Arrays of Objects – Memory Allocation

Box boxArr [ ] = new Box [2] ;
boxArr[0]= new Box(1,1,1);
boxArr[1]= new Box(2,3,4);

Note: boxArr is a reference to an array of 2 references, each pointing at another Box object.



Arrays Initialization

Arrays may be initialized during declaration or assigned values after it.

```
String names[ ] =new String[3];
names[0]=new String ("John");
names[1]=new String("Bryce");
names[2]=new String("Levy");

String names[ ] ={
  “John”,
  “Bryce”,
 “Levy”
};
```

```
Box boxes[ ] =new Box[3];
boxes[0]=new Box (10,20,10);
boxes[1]=new Box (3,5,13);
boxes[2]=new Box (8,6,11);

Box boxes[ ] ={
 new Box (10,20,10),
 new Box (3,5,13),
 new Box (8,6,11)
};
```

MultiDimensional Arrays

- Array of arrays.
- Example:

```
short twoDim[ ][ ];
twoDim = new short[4][ ];  //twoDim is a reference to
                           //array of four elements (each
                           //element is of type array of short).
twoDim[0]=new short[9];
twoDim[1]=new short[3];

short twoDim[ ][ ] = new short [ ] [9] ;            //illegal
twoDim = new short[4][5];                           //legal
```

hi-tech college

JOHN BRYCE
Leading in IT Education
a matrix company

Arrays Bounds

- Arrays subscripts begin at 0 and may have a max value of the size of the array minus 1.
- Example:

```
int arr[ ]=new int [10];
for (int i=0;i<arr.length; i++)  {
   System.out.println(arr[0]);
}
```

Arrays Bounds

```
short twoDim[ ][ ];
twoDim = new short[4][ ];
twoDim[0]=new short[9];
twoDim[1]=new short[3];
twoDim[2]=new short[6];
twoDim[3]=new short[2];

int x=twoDim.length;          //4
int y=twoDim[1].length;       //3
```

Array Resizing

- An array can not be resized.
- The same array reference may be reinitialized to another array.
- Example:

```
   int arr[ ]=new int [10];
…
arr = new int [4];                //unless another reference to the
                                  //first array exist elsewhere, the first
                                  //array is lost and may be garbage
                                  //collected.
```

Copy an Array

- Use System.arraycopy(. . .) method to copy arrays.
- Syntax:

  System.arraycopy(sourceArr,src_starting_ind, target, target_starting_ind, sourceArr.length);

Note: System.arraycopy( ) copies primitives or references, not objects.

Copy an Array - Example

```
int source[ ] = {1,2,3,4,5,6};
int target[ ] = {2,54,67,87,87,87,87,4,3,4,65};
System.arraycopy(source, 0, target, 4, 2);
int i;
for (i=0;i < target.length; i++)
    System.out.println(target[i]);

//output is:
// 2,54,67,87,1,2,87,4,3,4,65
```