Jagged Arrays

2D Jagged Array

Scenario: To store the marks of students in the following scenario:

Classrooms	Students
0	0-4
1	0-2
2	0-3

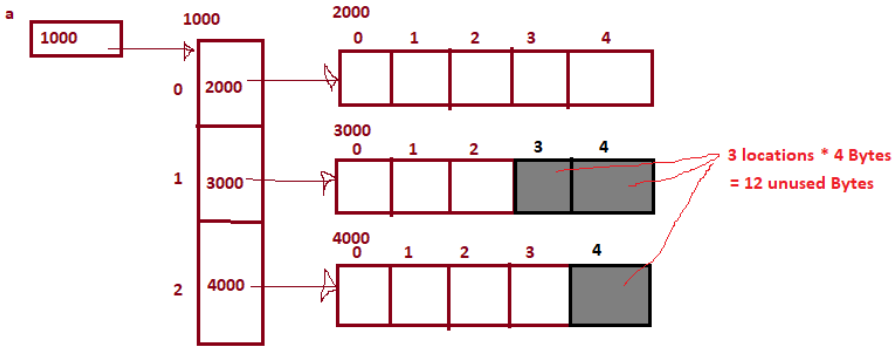
Declaration: `int[][] a = new int[3][ ];` //Classrooms/Rows

```
a[0] = new int[5];
a[1] = new int[3];
a[2] = new int[4];
```

//Students/Columns

Can we handle irregular data using regular arrays?

`int[][] a = new int[3][5];`



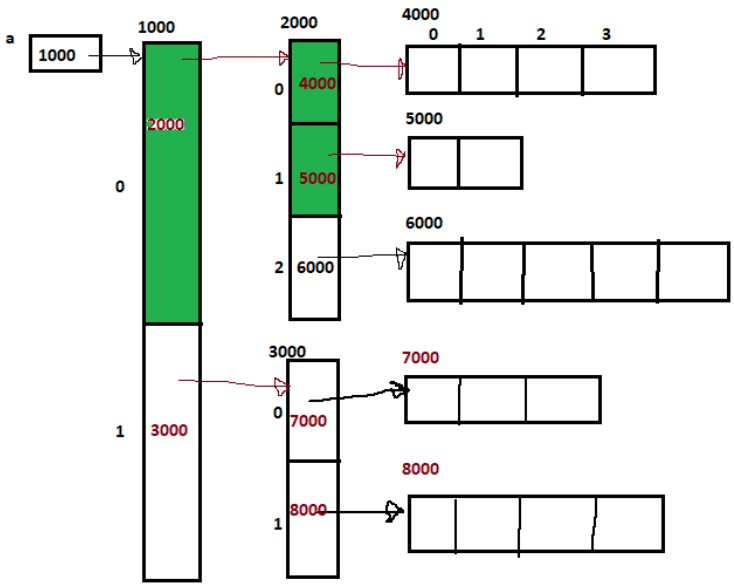
Adv:

1. Efficient Utilization of Memory
2. Can handle irregular data

3D Jagged Array

Scenario:

Schools	Classrooms	Students
0	0	0-3
	1	0-1
	2	0-4
1	0	0-2
	1	0-3



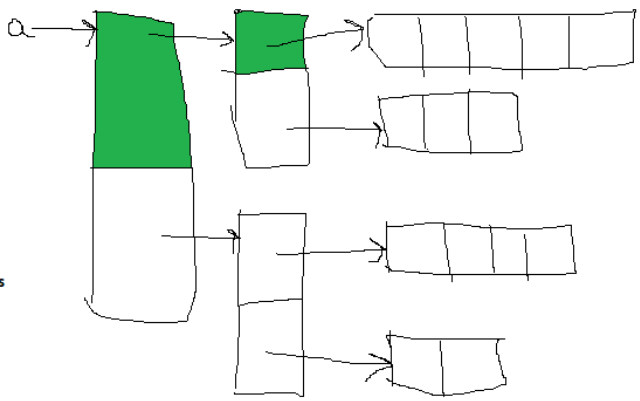
Declaration: `int[][][] a = new int[2][ ][ ];` //Schools/Blocks

```
a[0] = new int[3][ ];
a[1] = new int[2][ ];
```
//Classrooms/Rows

```
a[0][0] = new int[4];
a[0][1] = new int[2];
a[0][2] = new int[5];
a[1][0] = new int[3];
a[1][1] = new int[4];
```
//Students/Columns

Scenario:

Schools	Classrooms	Students
0	0	0-4
	1	0-2
1	0	0-3
	1	0-1

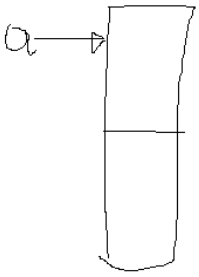


Declaration: `int[][][] a = new int[2][2][];` //Schools & Classrooms

```
a[0][0] = new int[5];
a[0][1] = new int[3];

a[1][0] = new int[4];
a[1][1] = new int[2];
```

Scenario:

Schools	Classrooms	Students
0	0	0-4
	1	0-4
	2	0-4
1	0	0-4
	1	0-4



Declaration: `int[][][] a = new int[2][ ][ ];` //Schools

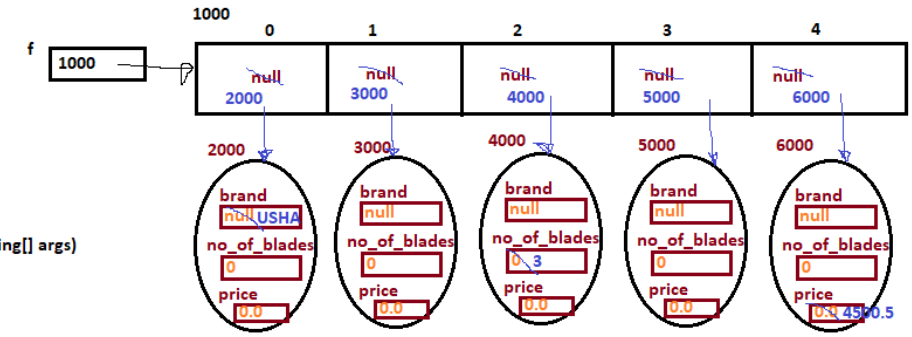
`a[0] = new int[3][5];`
`a[1] = new int[2][5];`
} classrooms & students

```
int[] a = new int[5];
float[] a = new float[5];
char[] a = new char[5];
boolean[] a = new boolean[5];
```

Create an array to store 5 Fan Objects?

```
class Fan
{
   String brand;
   int no_of_blades;
   float price;
}

class Launch
{
   public static void main(String[] args)
   {
         Fan[] f = new Fan[5];

   f[0] = new Fan();
   f[1] = new Fan();
   f[2] = new Fan();   <OR>
   f[3] = new Fan();
   f[4] = new Fan();

    f[0].brand = "USHA";
    f[2].no_of_blades = 3;
    f[3].price = 4500.5f;
   }
}
```

```
for(int i=0; i<=f.length-1; ++i)
{
     f[i] = new Fan();
}
```



Literal Initialization

```
int[] a = new int[5];
a[0] = 10;
a[1] = 20;
a[2] = 30;
a[3] = 40;
a[4] = 50;
```

<OR>

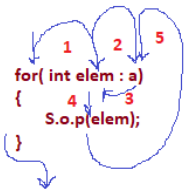
```
int[] a = {10, 20, 30, 40, 50};
```

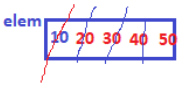
```
String[] s = {"PW", "Java", "DSA", "C++"};
```

```
for(int i=0; i<=a.length-1; ++i)
{
    int elem = a[i];
    S.o.p(elem);
}
```

<OR>

```
for( int elem : a)
{
    S.o.p(elem);
}
```






```
for( String elem : s)
{
    S.o.p(elem);
}
```

Output:

10 20 30 40 50

Assignment

Design a for-each loop to iterate over -

i. 2D Array

iii. 3D Array

```
class Arrays
{
   static copyOf()
   static fill()



}
```

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

Arrays.fill(a, 5);
```

```
int[] a = {40, 20, 10, 50, 30};

Arrays.sort(a);    //In-place sorting

for(int elem: a)
{
  S.o.p(elem);
}
```

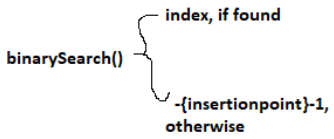
Output: 10 20 30 40 50

Pre-requisite: SORTED

```
                0  1  2  3  4
      int[] a = {10, 20, 30, 40, 50};

int index = Arrays.binarySearch(a, 30);  //2

int index =  Arrays.binarySearch(a, 25);  //-3 = -2-1

int index =  Arrays.binarySearch(a, 45); //-4-1 = -5
```

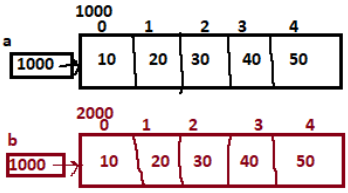
```
public static int[] copyOf(int[] x, int len)
{
        int[] b = new int[len];

        for(int i=0; i<=len-1 && i<=x.length-1; ++i)
        {
           b[i] = x[i];
        }

        return b;
}
```

```
int[] a = {10, 20, 30, 40, 50};
int[] b = {10, 20, 30, 40, 50};

boolean res = Arrays.equals(a, b);

if(res )
{
   S.o.p("Both arrays are equal");
}
else
{
   S.o.p("Both arrays are not equal");
}
```



```
class Arrays
{
static  copyOf()
```

```
int[] a = {10, 20, 30, 40, 50};
int[] b = Arrays.copyOf( a, 5);
```




```
int[] b = Arrays.copyOf(a, 3);
```

//Truncation

```
int[] b = Arrays.copyOf(a, 7);
```

//Padding

```
}
```