Nested Loop

Loop inside another loop

Nested Loop Example

```
// in a spreadsheet

Iterate all the row {

  In each row iteration

     Iterate over each column{
       read the value of cell
     }

     }
```

Excel Sheet

Columns / Rows	0	1	2
0	12	5	4
1	23	113	32

CYBERTEK

Nested Loop

```
//a nested for loop
for (int r = 0; r < 2; r++) {

    for (int k = 0; k < 3; k++) {

      System.out.print("|Row"+ r + "-Column"+k + "| ");

    }
     System.out.println();

}
```

Nested Loop

```
//Nested for each loop
   int[] rows = {1,2,3,4};
   int[] cols = {1,2,3};
   for (int row : rows) {

     for (int col : cols) {
       System.out.print(" |Row"+row+ "-Column"+col);
     }
     System.out.println();

   }
```

Nested Loop

```
// a nested while loop
  int i = 0 ;
while(i<5) {
    i++;
    int j = 0 ;
    while(j<3) {
      j++;
      System.out.print(" |Row"+i+"-Column"+j);
    }
    System.out.println();

}
```

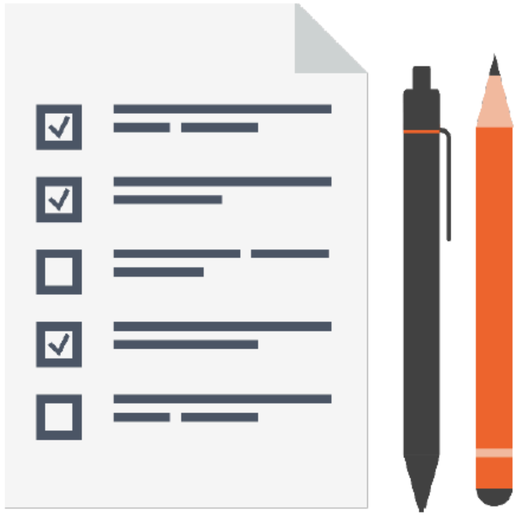
Nested Loop

```
// a nested while loop
int i1 = 0 ;
do {
  i1++;
  int j = 0 ;
     do{
       j++;
       System.out.print(" |Row"+i1+"-Column"+j);
     } while(j<3) ;

  System.out.println();

}while(i1<5);
```

Nested Loop

```
//Nested for loop and while loop
   for (int j = 0; j < 4; j++) {

     int k=0;
     while(k<3) {
       System.out.print(" |Row"+j+"-Column"+k);
       k++ ;
     }
     System.out.println();
   }
```

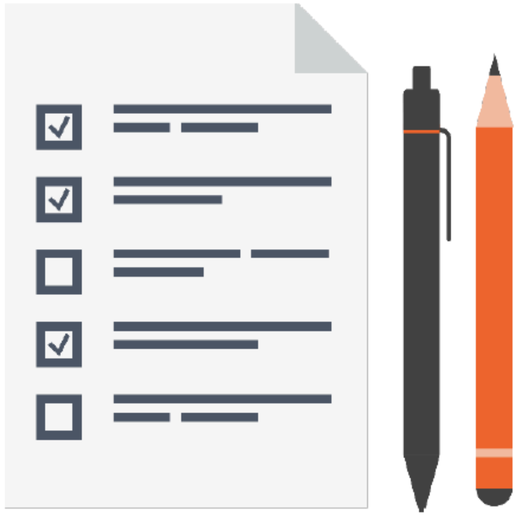
Java Programming
Agenda



- Understanding Java Arrays
- Sorting an Array
- Using binary-search
- Multi-Dimensional Array

CYBERTEK

After today's session you should be able to:



- Have better understanding of Array
- Learn Arrays class to sort an Array
- Use Arrays class to binary search
- Understand and create Multi-Dimensional Array

- Create :
  Simple program using Arrays

CYBERTEK

Array

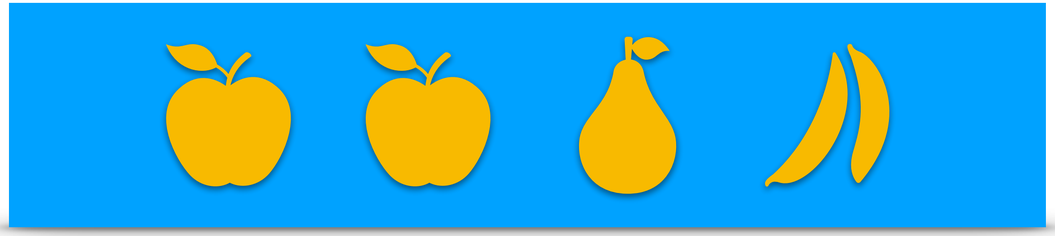
Single Object that hold multiple value of same type

Examples

Grocery List

Contacts

Prime Numbers



2, 3, 5, 7, 11, 13, 17, 19, 23, 29

Creating an Array

```
// Syntax format

dataType [ ] variableName = new  dataType[size] ;
```

Creating an Array of Primitives

```
// Syntax for creating int Array

    int[ ] numbers = new int[3];

   // or

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

numbers

Element	0	0	0
Index	0	1	2

Creating an Array of Primitives

```
// Syntax for creating int Array

int[ ] numbers = new int[3];

  numbers[0] = 2 ;
  numbers[1] = 5 ;
  numbers[2] = 7 ;
```

numbers

Element	2	5	7
Index	0	1	2

Creating an Array

```
// Syntax for creating int Array

    int[ ] numbers = new int[ ] {2, 5, 7};

// or directly as below

    int[ ] numbers =  {2, 5, 7};
```

numbers

Element	2	5	7
Index	0	1	2

String Array Example

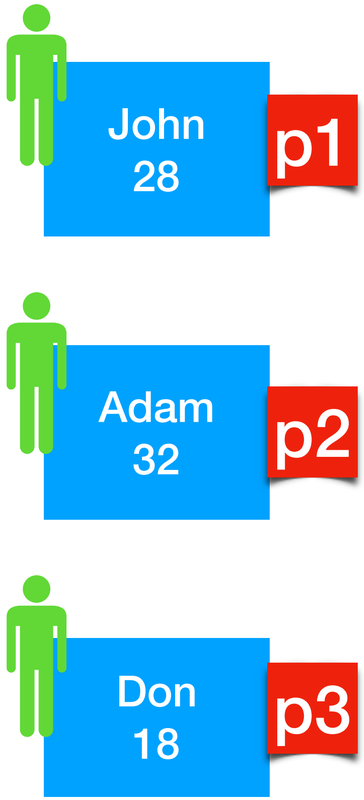
```
// Syntax for creating String Array

      String[ ] names = new String[ ] {“John”,”Adam”, “Don”};

// or directly as below


    String[ ] names = {“John”,”Adam”, “Don”};
```

names

Element	John	Adam	Don
Index	0	1	2

CYBERTEK

Creating an Array of Reference Type

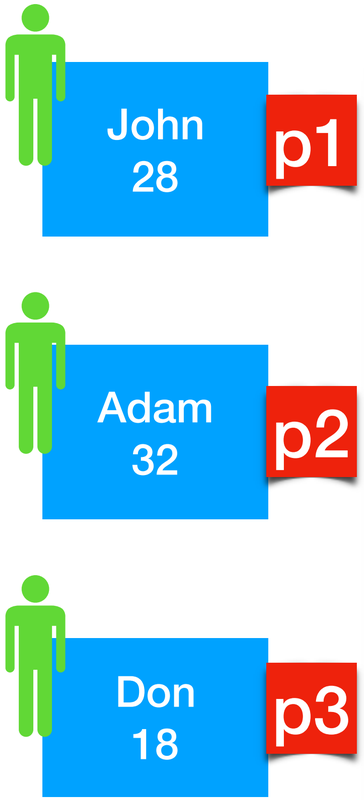



```
// Syntax for creating Person Array

Person p1 = new Person();  //assume name and age assigned already
Person[ ] people = new Person[3];

  people[0] = p1 ;
  people[1] = p2 ;
  people[2] = p3 ;
```

people

Element	p1	p2	p3
Index	0	1	2

Creating an Array of Reference Type




```
// Syntax for creating Person Array

Person[ ] people = new Person[ ] {p1, p2, p3};

// or

Person[ ] people = {p1, p2, p3};
```

people

Element	p1	p2	p3
Index	0	1	2

Using an Array

```
// Syntax for accessing Array elements
 int[] intArray = {2,5,7};

 System.out.println("Numbers 1 " + numbers[0]);
 System.out.println("Numbers 2 " + numbers[1]);
 System.out.println("Numbers 3 " + numbers[2]);
```

numbers

Element	2	5	7
Index	0	1	2

Looping an Array

```
String[] names = {“John”,”Adam”, “Don”};
for (int i = 0; i < names.length(); i ++) {

    System.out.println(“names ” +names[i]);

 }
```

Looping an Array

```
String[] names = {“John”,”Adam”, “Don”};
for (String item: names) {

    System.out.println(“names ” +item);

 }
```

Arrays Class

Using **Arrays** Class from java.util package for common array operations

```
import  java.util.Arrays;

import  java.util.*;
```

Arrays Class

```
int[] targetArrayObject = {11,4,6};
 int   targetItemToSearch = 4 ;

 Arrays.sort(targetArrayObject) ; // 4,6,11
Arrays.binarySearch(targetArrayObject, targetItemToSearch); //0
// Optionally
 Arrays.toString(targetArrayObject);// [4,6,11]
```

Sorting an Array

```
 int[] intArray = {2,15,7,1,3};
// Syntax for sorting Array elements
 Arrays.sort(intArray);

 for (int item: intArray) {

     System.out.println("numbers " +item);

  }
```

Before

Element	2	15	7	1	3
Index	0	1	2	3	4

After

Element	1	2	3	7	15
Index	0	1	2	3	4

Getting string value of Array items

```
int[] ints = {2,15,7,1,3};

Arrays.sort(ints);
// Syntax for sorting Array elements
System.out.println(Arrays.toString(ints));

System.out.println(ints); //[I@7a46a697
```

Sorting a String Array

```
 String[] names = {“John”,”Adam”, “Don”};
// Syntax for sorting Array elements
 Arrays.sort(ints);
 for (String item: names) {

     System.out.println(“names ” +item);

  }
```

Before

Element	John	Adam	Don
Index	0	1	2

After

Element	Adam	Don	John
Index	0	1	2

Searching within an Array

Arrays Class from java.util package provide convenient way to search ONLY IF the array is already sorted

```
// Syntax format
Arrays.binarySearch(yourArray, element) ;
```

Binary search result

Senario	Result
Element found in sorted array	Index of match
Element not found in sorted array	Negative value showing one smaller than the negative of index, where m match needs to be instead to preserve sorted order
Unsorted Array	Unpredictable result

Using Binary Search

```
// Syntax for accessing Array elements
  int[] ints = {2,5,7,14};

System.out.println(Arrays.binarySearch(ints,2));  0
System.out.println(Arrays.binarySearch(ints,7));  2
System.out.println(Arrays.binarySearch(ints,9));  -4
System.out.println(Arrays.binarySearch(ints,15)); -5
```

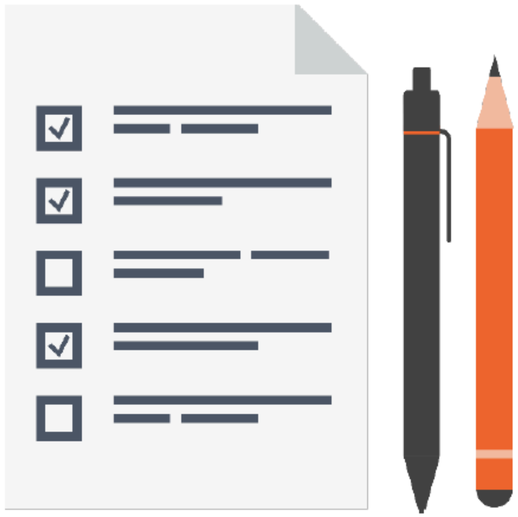
Element	2	5	7	14
Index	0	1	2	3

Using Varargs

Varargs is short for variable arguments where you can pass a variable number of argument with same type.

It can be seen same as array when used as method parameters

Java Programming
Agenda

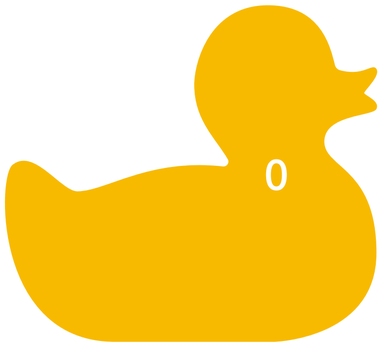


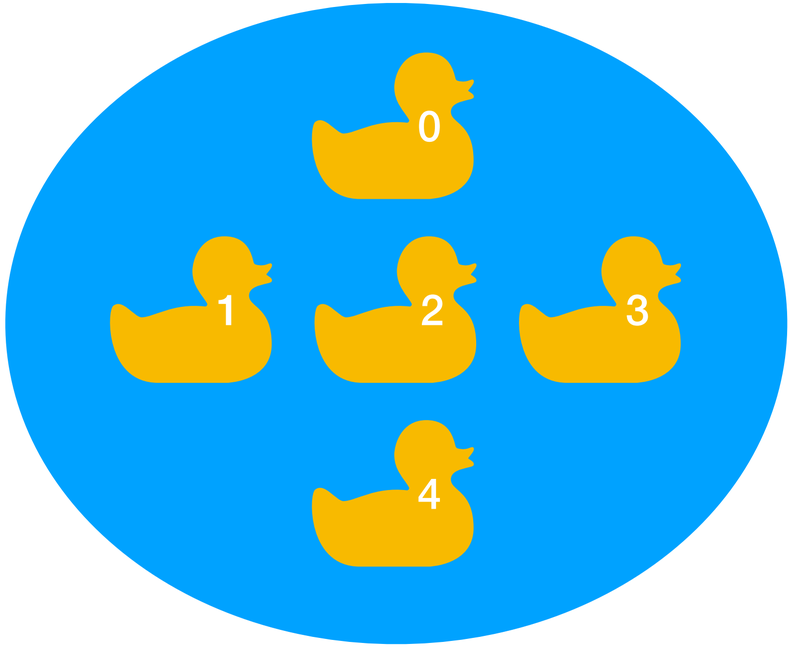
- Understanding Multi-Dimensional Array
- Declaring , Constructing , Initializing Multi-Dimensional Array
- Accessing multi-dimensional Array items using nested loop

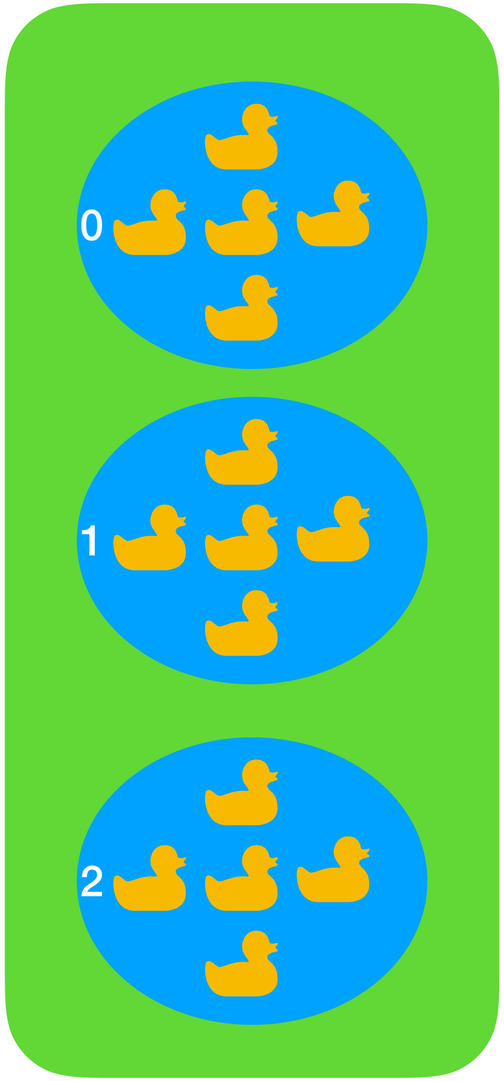
CYBERTEK

Multi Dimensional Array

An Array Object that store other arrays



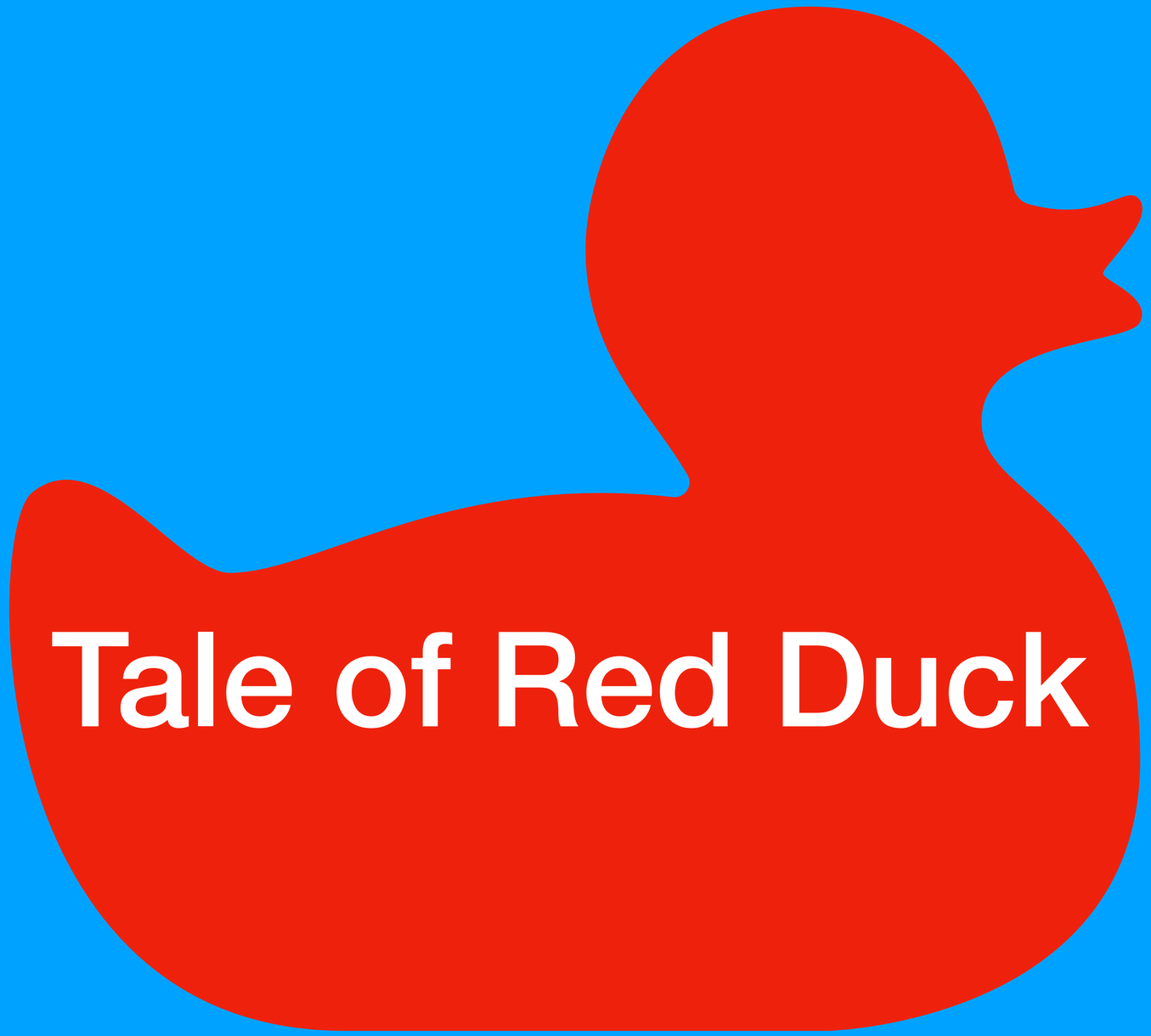





```
int singleDuck ;
```

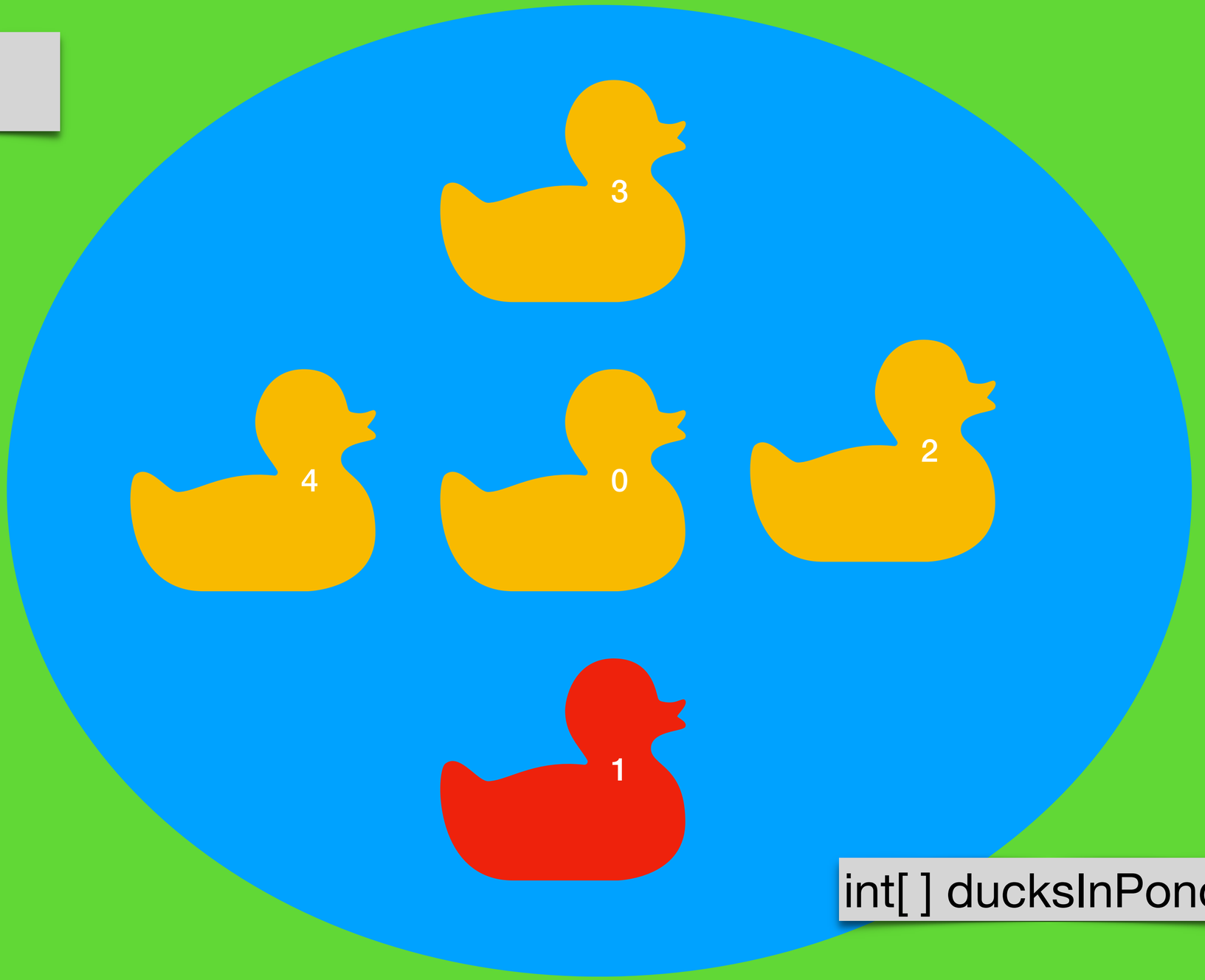
```
int[ ] ducks ;
```

```
int[ ][ ] groupsOfDucks;
```



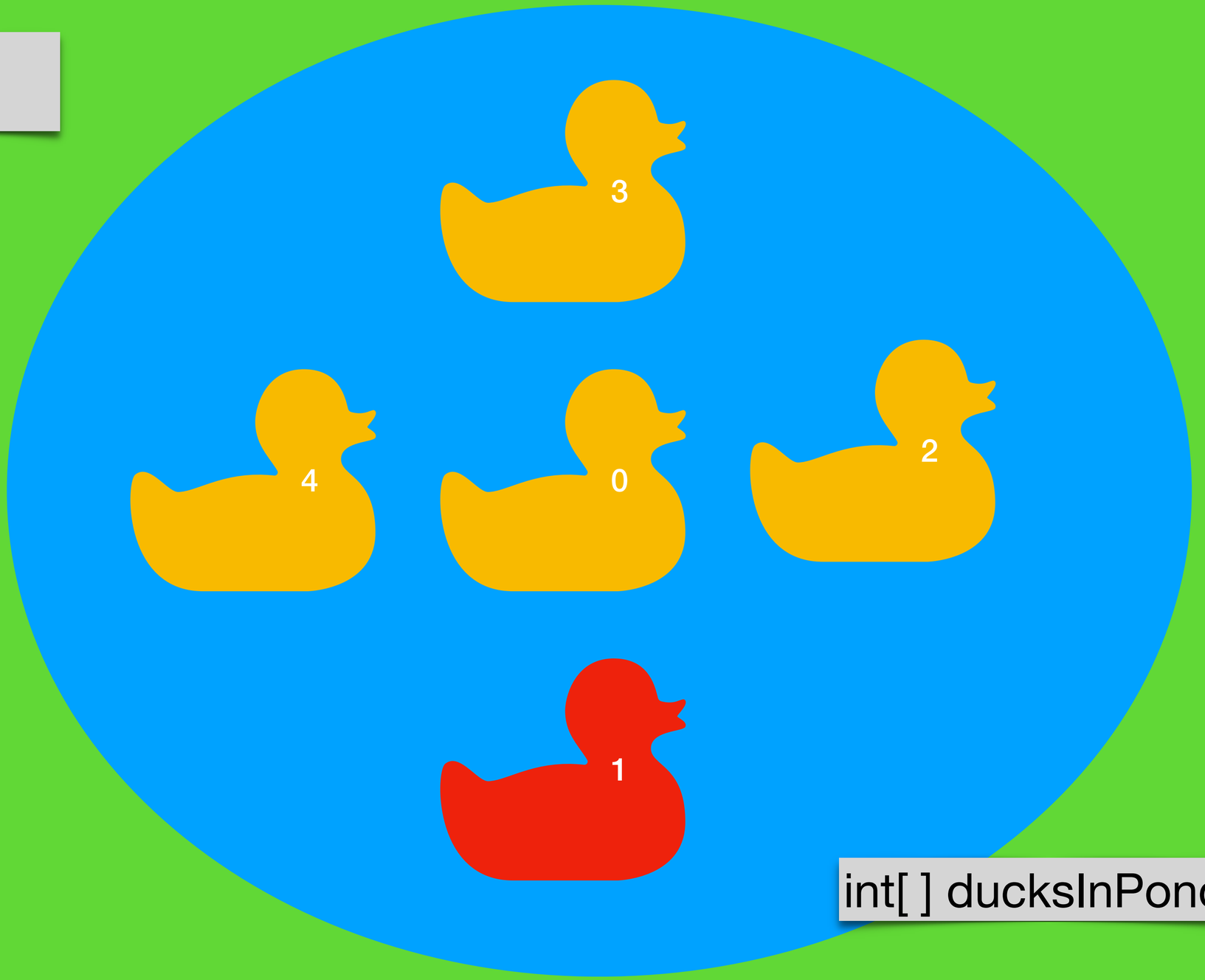

```
int singleDuck;
```

5 ducks



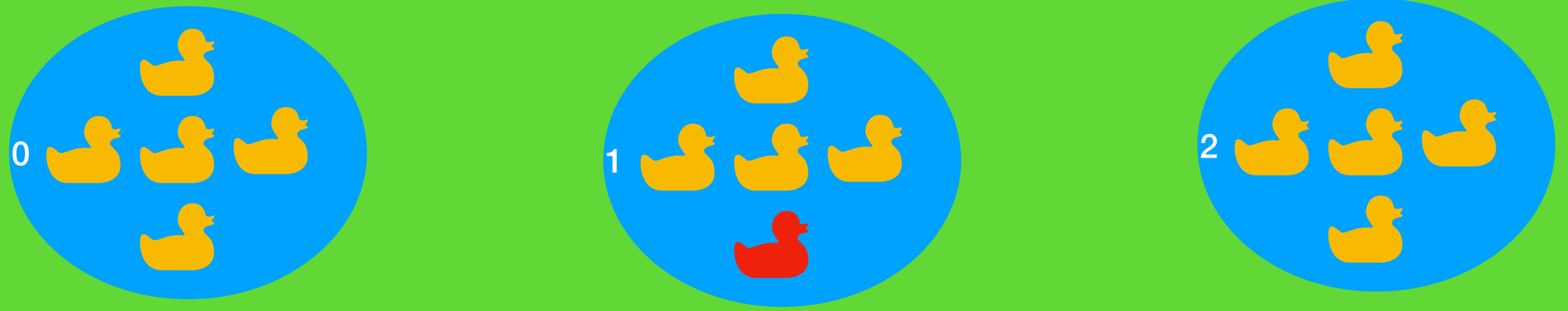
int[ ] ducksInPond;

5 ducks



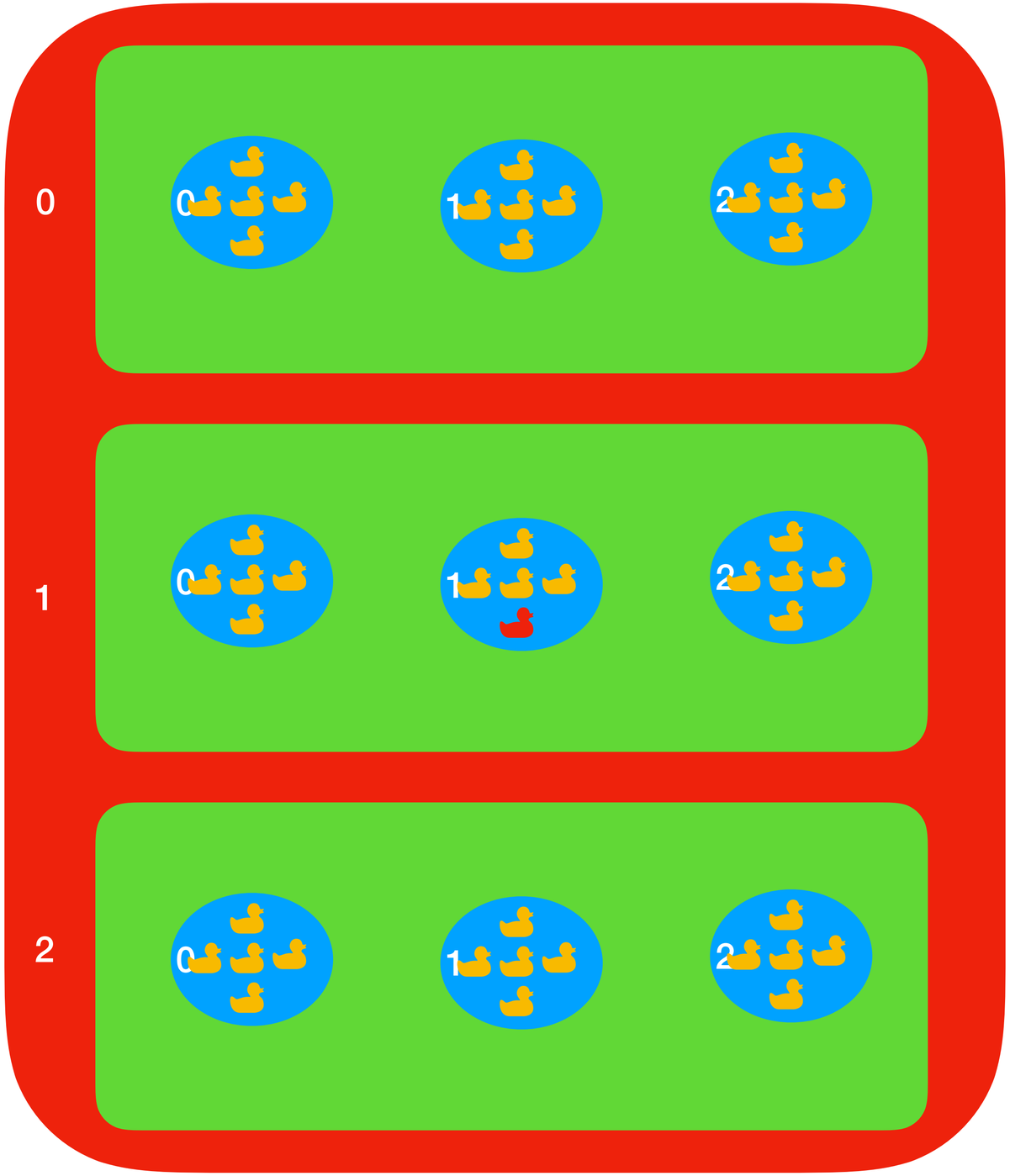
int[ ] ducksInPond;

3 ponds



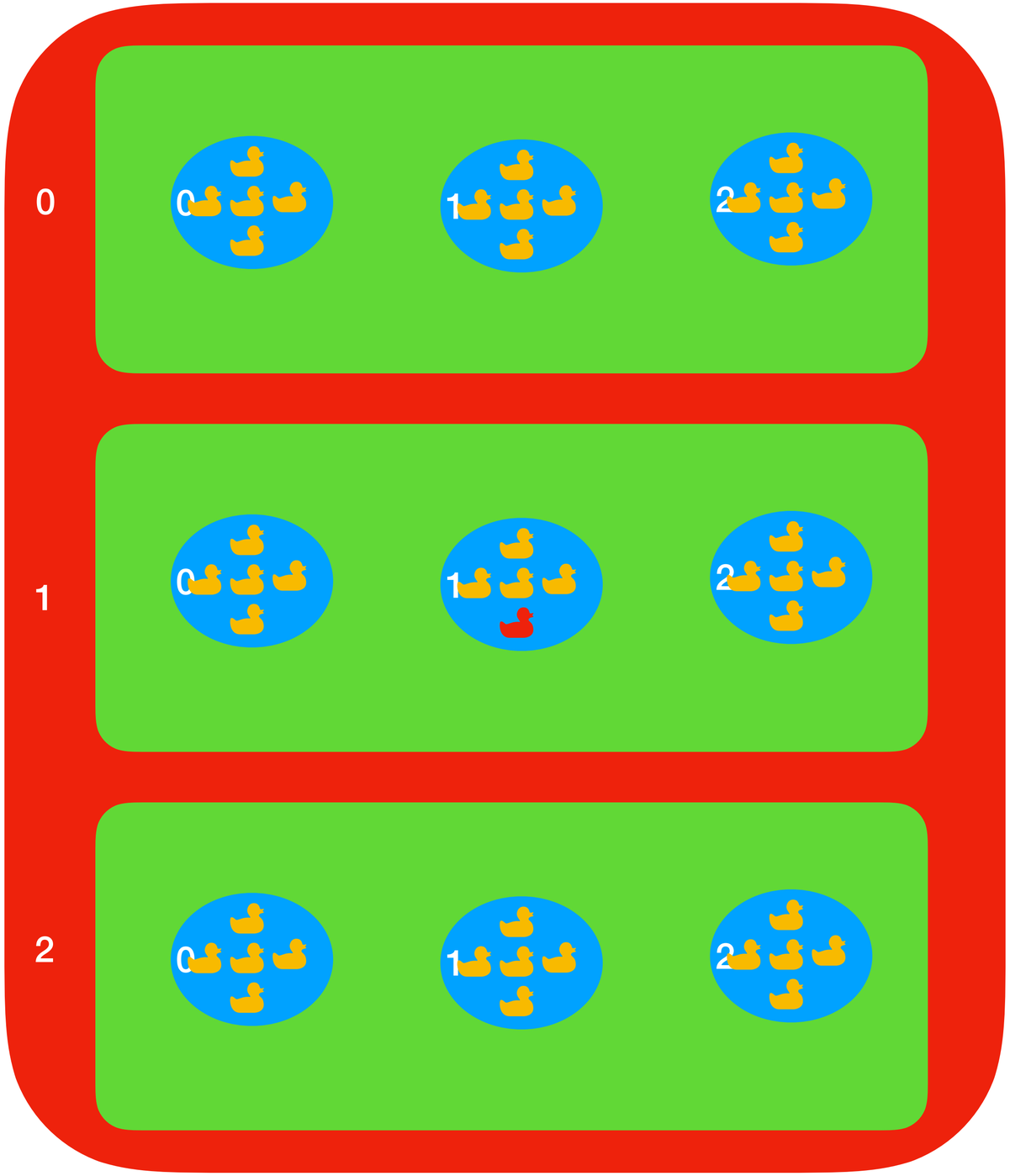

```
int[ ][ ] DuckFarm;
```

3 farms



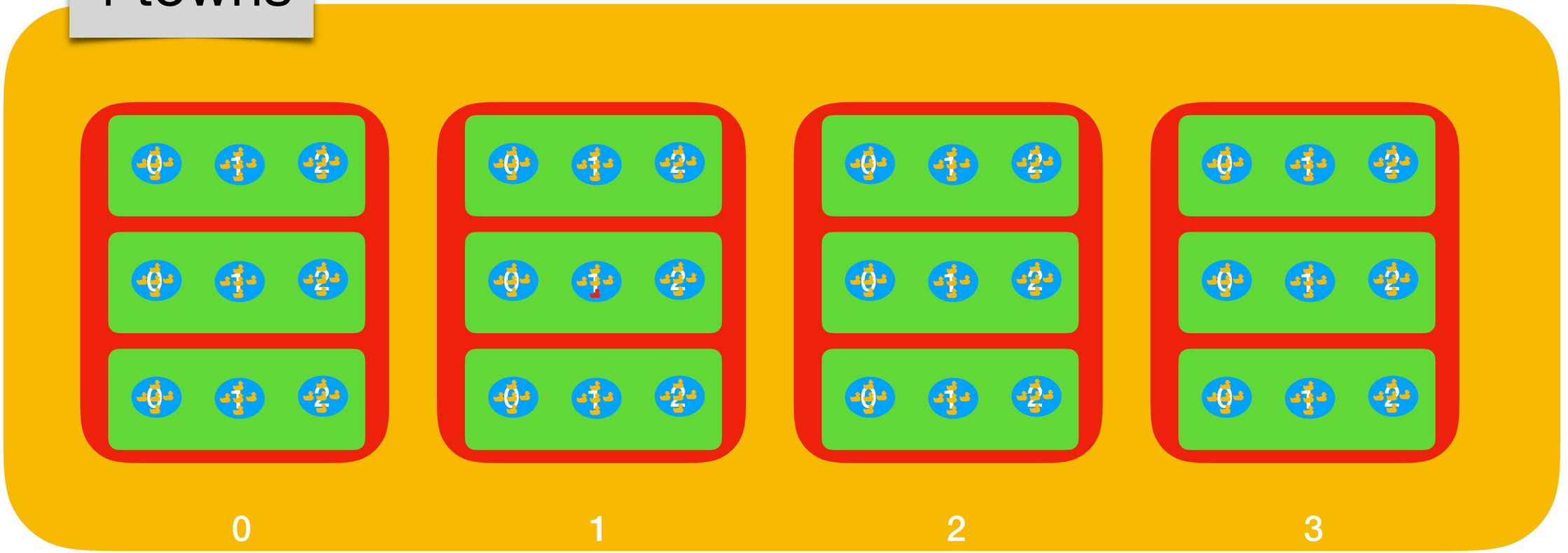

```
int[ ][ ][ ] duckTown;
```

3 farm



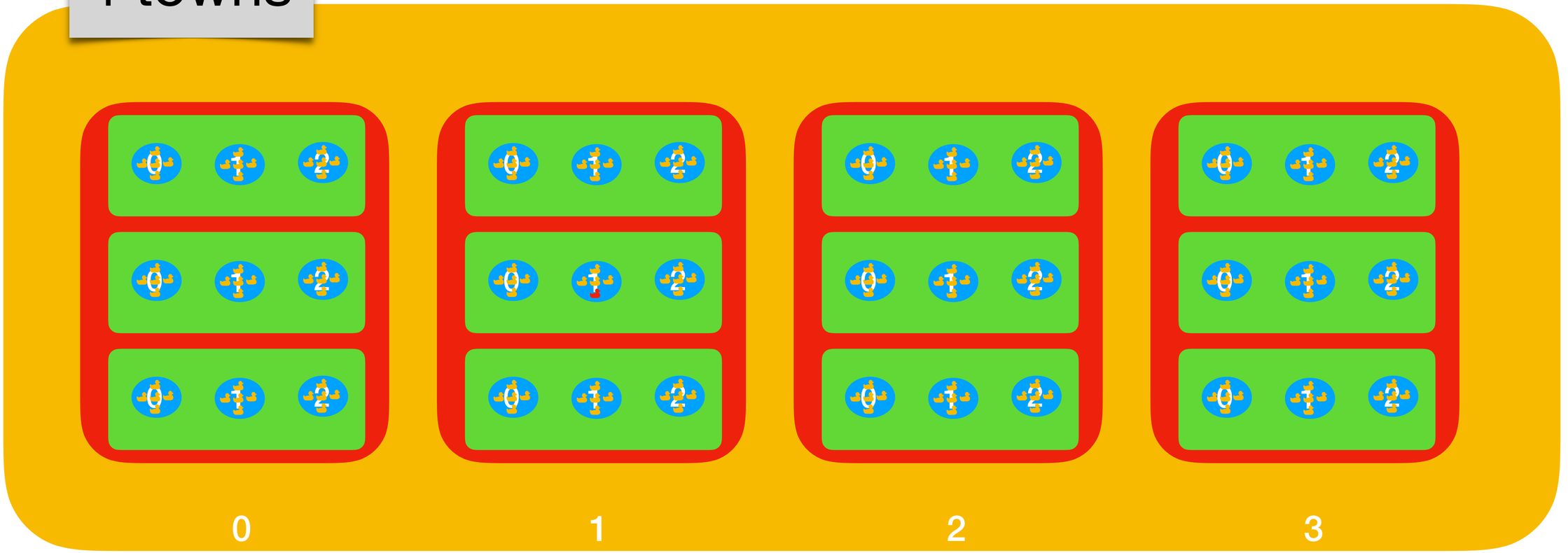

```
int[ ][ ][ ] duckTown;
```

4 towns



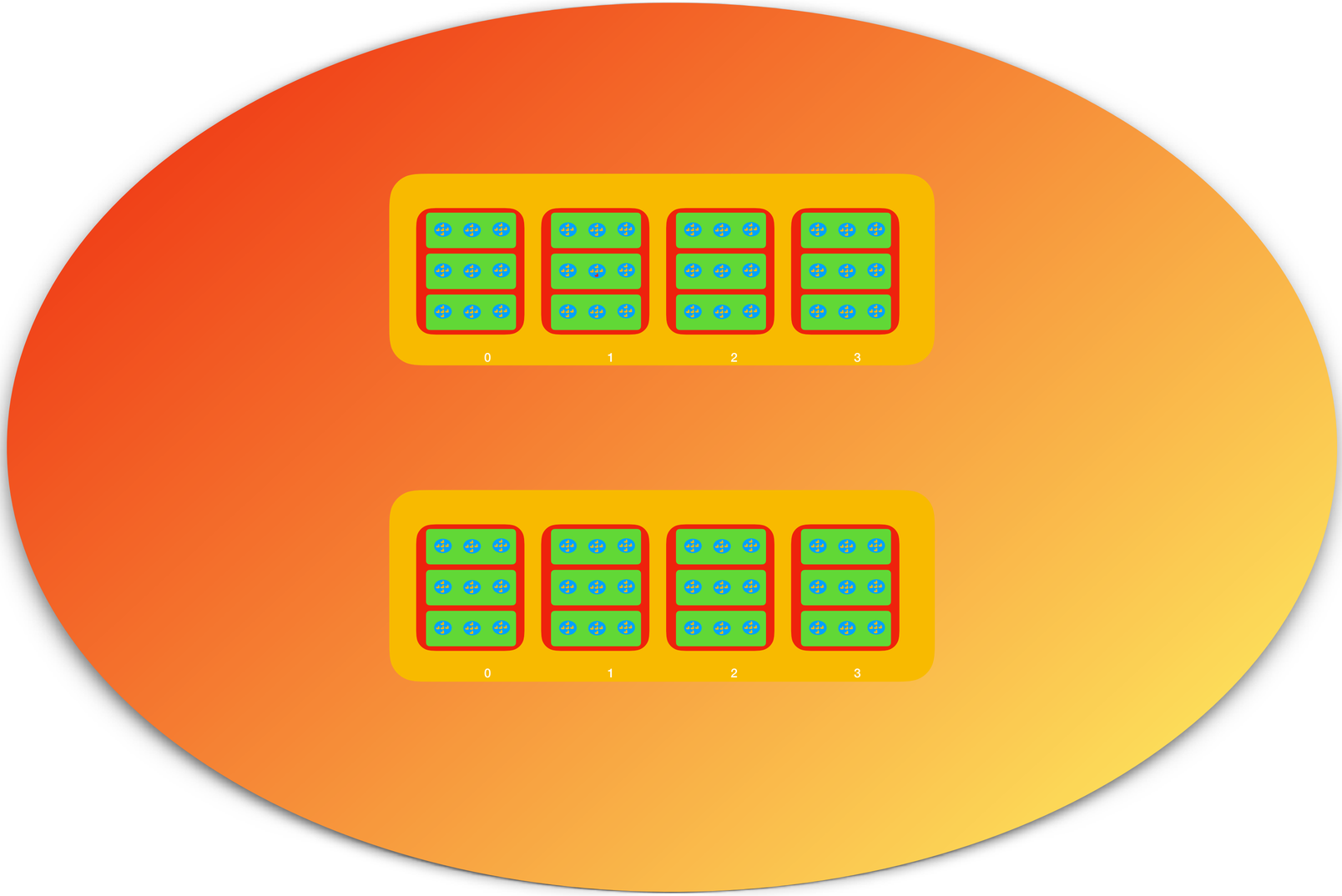

```
int[ ][ ][ ][ ] duckCounty;
```

4 towns




```
int[ ][ ][ ][ ] duckCounty;
```

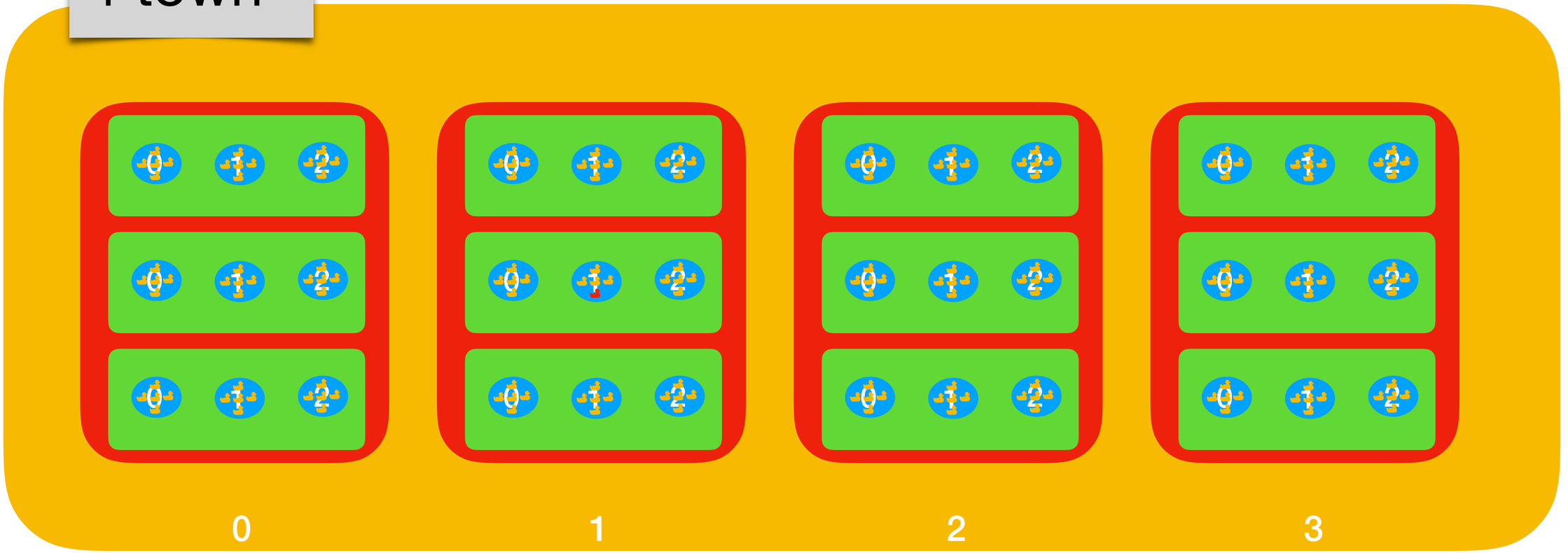
CYBERTEK




```
int[ ][ ][ ][ ][ ] duckDynasty;
```

CYBERTEK

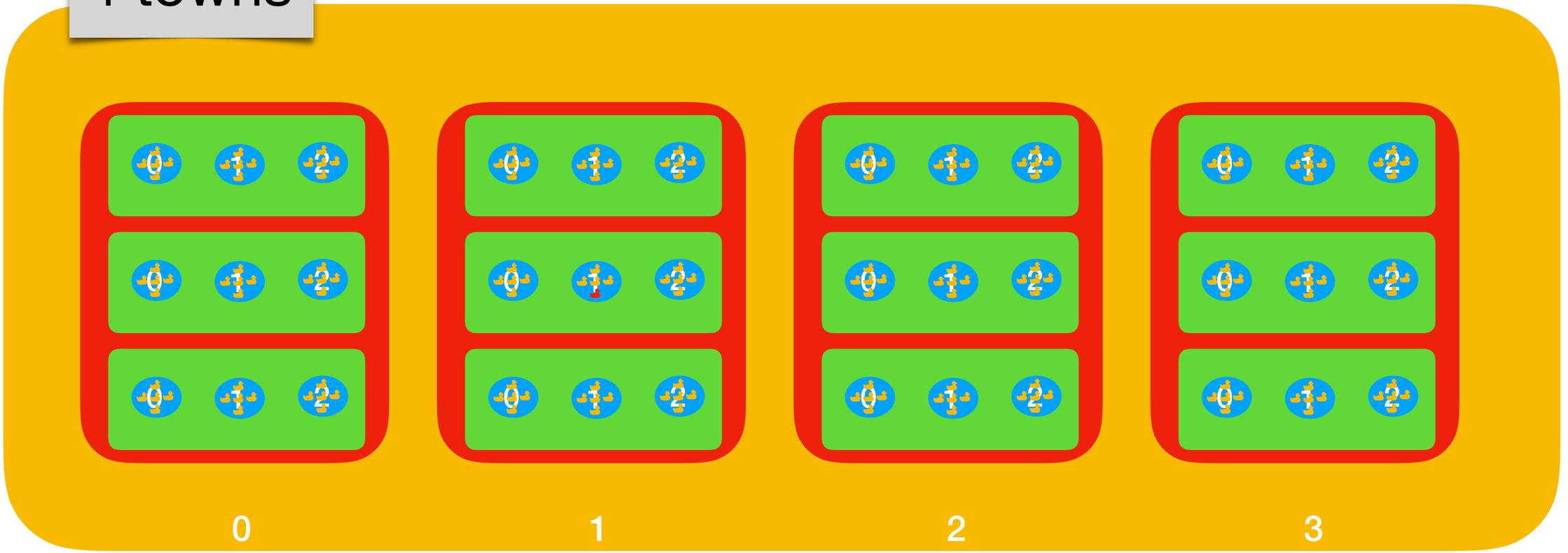
4 town




```
int[ ][ ][ ][ ] duckCounty;
```

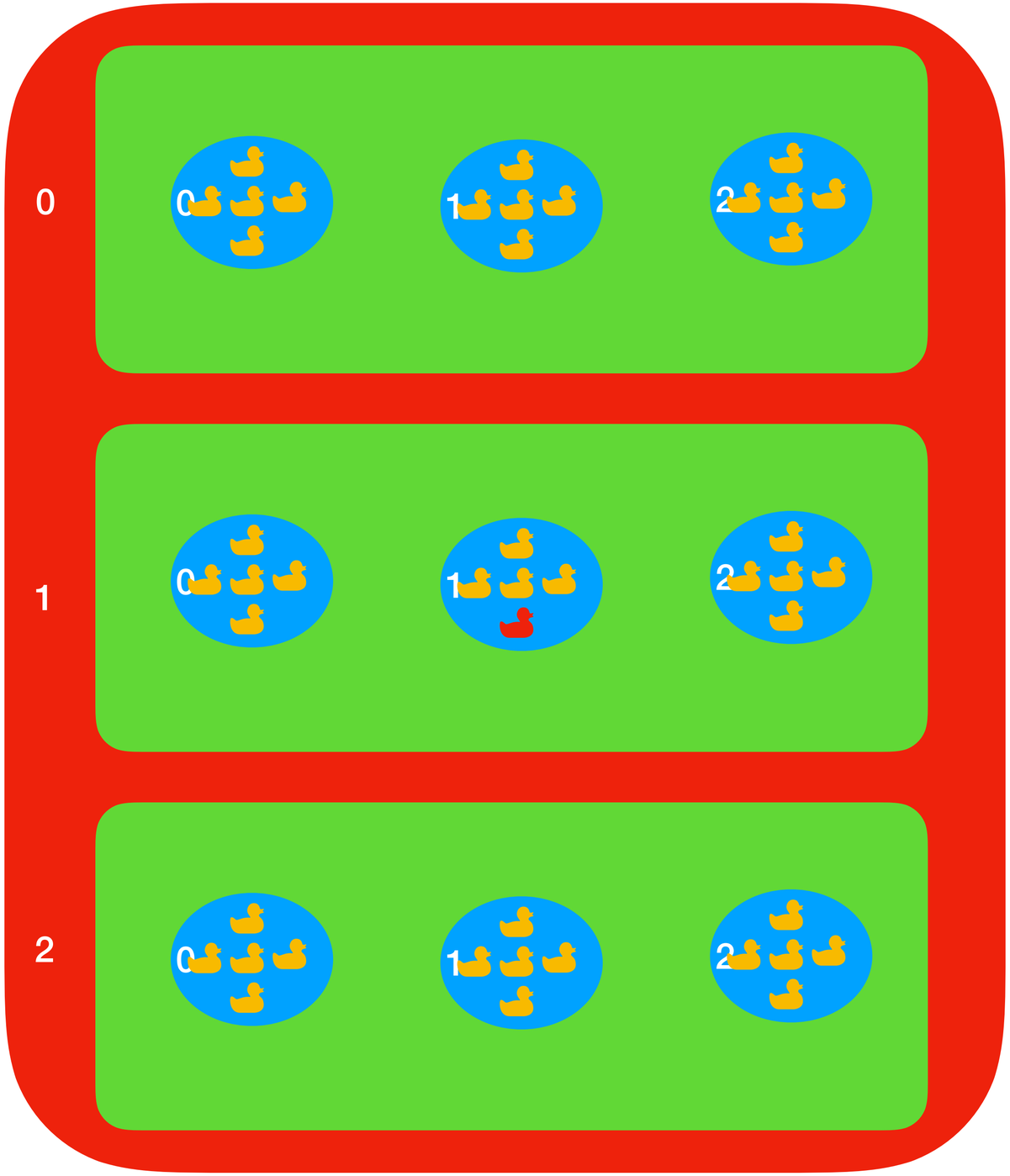
CYBERTEK

4 towns



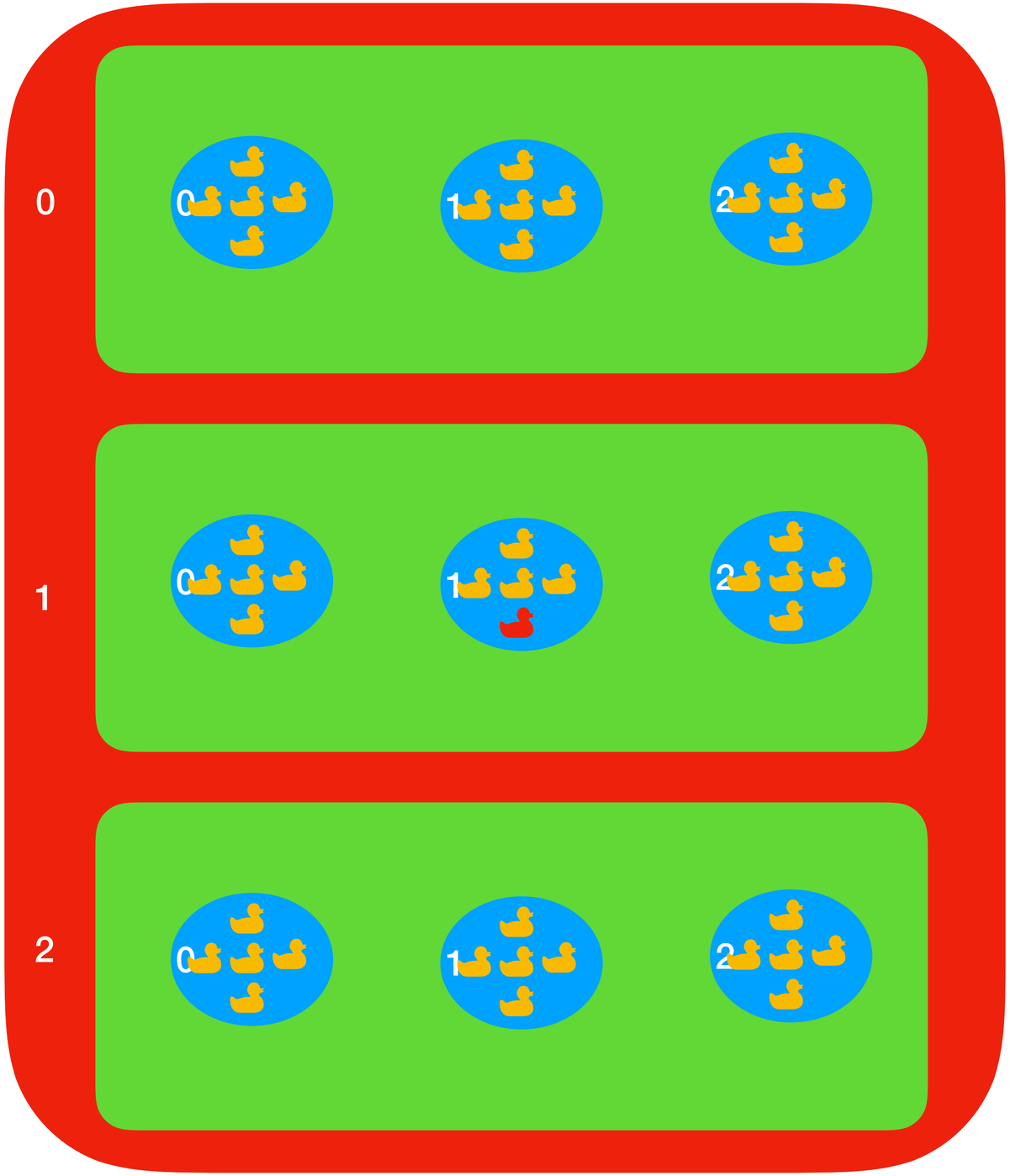

```
int[ ][ ][ ][ ] duckCounty;
```

3 farm



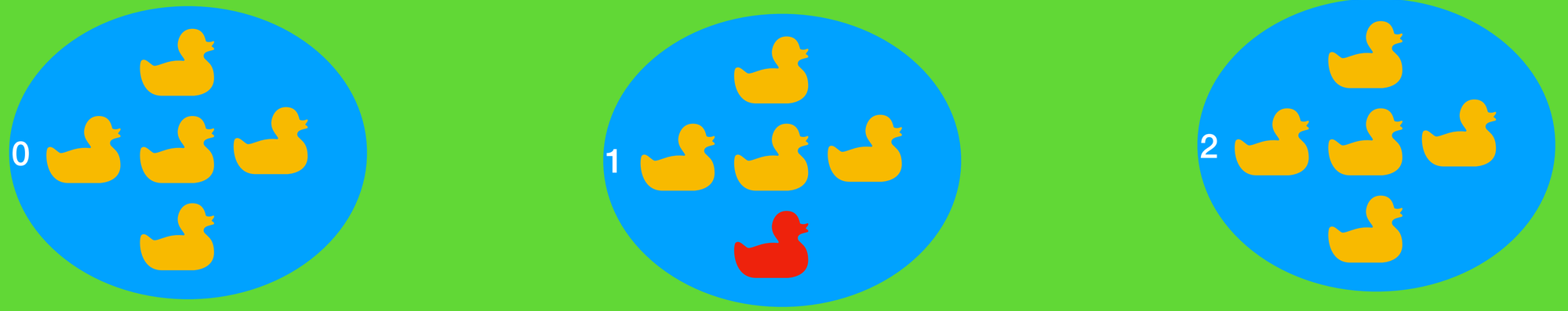

```
int[ ][ ][ ] duckTown;
```

3 farms



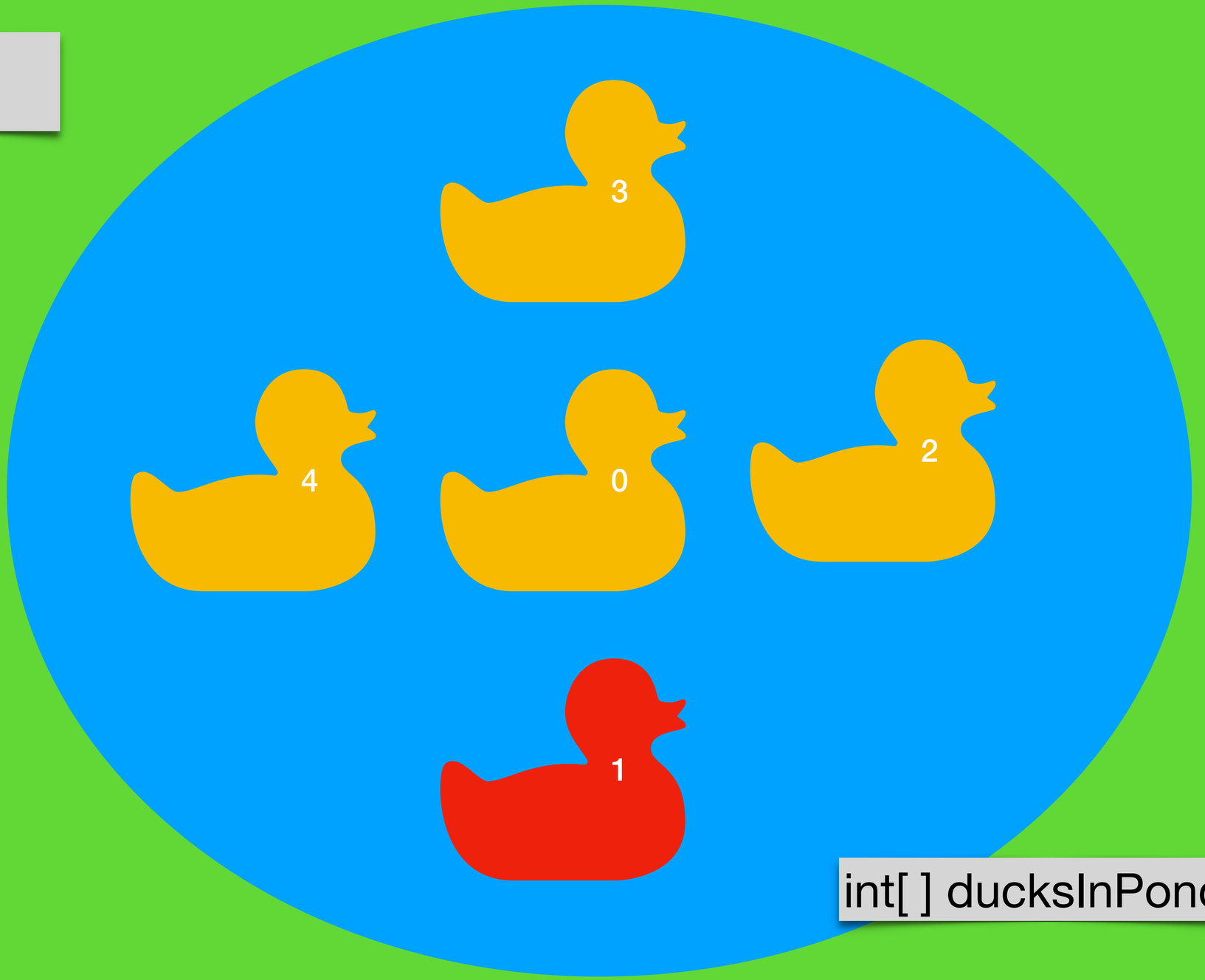

```
int[ ][ ][ ] duckTown;
```

3 ponds



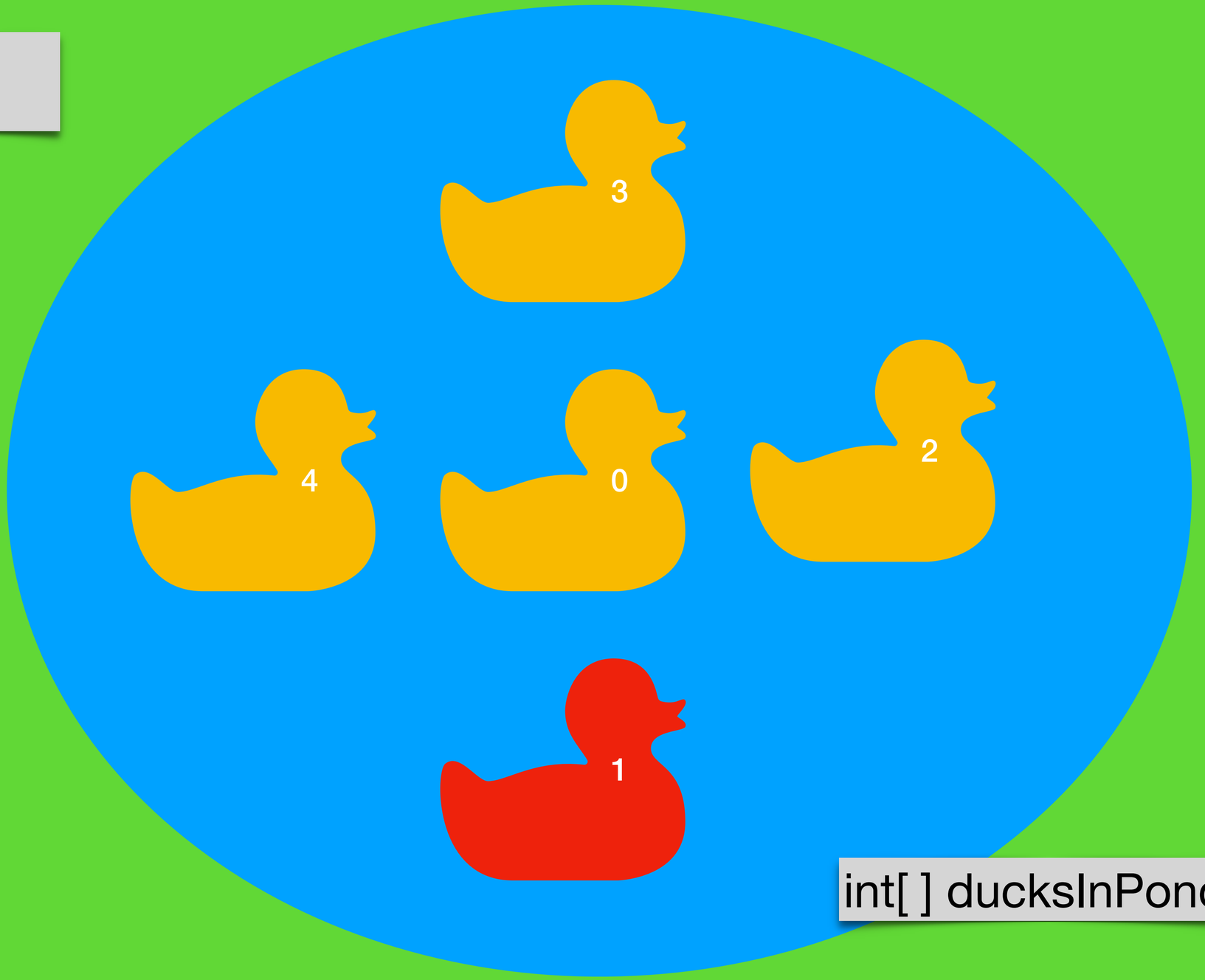

```
int[ ][ ] DuckFarm;
```

5 ducks



int[ ] ducksInPond;

5 ducks



int[ ] ducksInPond;

```
int singleDuck;
```

1 Dimensional Array

An int Array

Element	12	13	43	5	66
Index	0	1	2	3	4

```
int[ ] intArray = {12, 13, 43, 5, 66} ;
```

CYBERTEK

2 Dimensional Array

Element	12	13	43	5	66
Index	0	1	2	3	4

Element	12	13	43	5	66
Index	0	1	2	3	4

Element	12	13	43	5	66
Index	0	1	2	3	4

Element	12	13	43	5	66
Index	0	1	2	3	4

Element	12	13	43	5	66
Index	0	1	2	3	4

Element	12	13	43	5	66
Index	0	1	2	3	4

An Array of Int Array

```
int[ ][ ] ArrayOf_IntArray ={ intArray, intArray, intArray,
                              intArray intArray, intArray}    ;
```

0

Element	12	13	43	5	66
Index	0	1	2	3	4

1

Element	12	13	43	5	66
Index	0	1	2	3	4

2

Element	12	13	43	5	66
Index	0	1	2	3	4

3

Element	12	13	43	5	66
Index	0	1	2	3	4

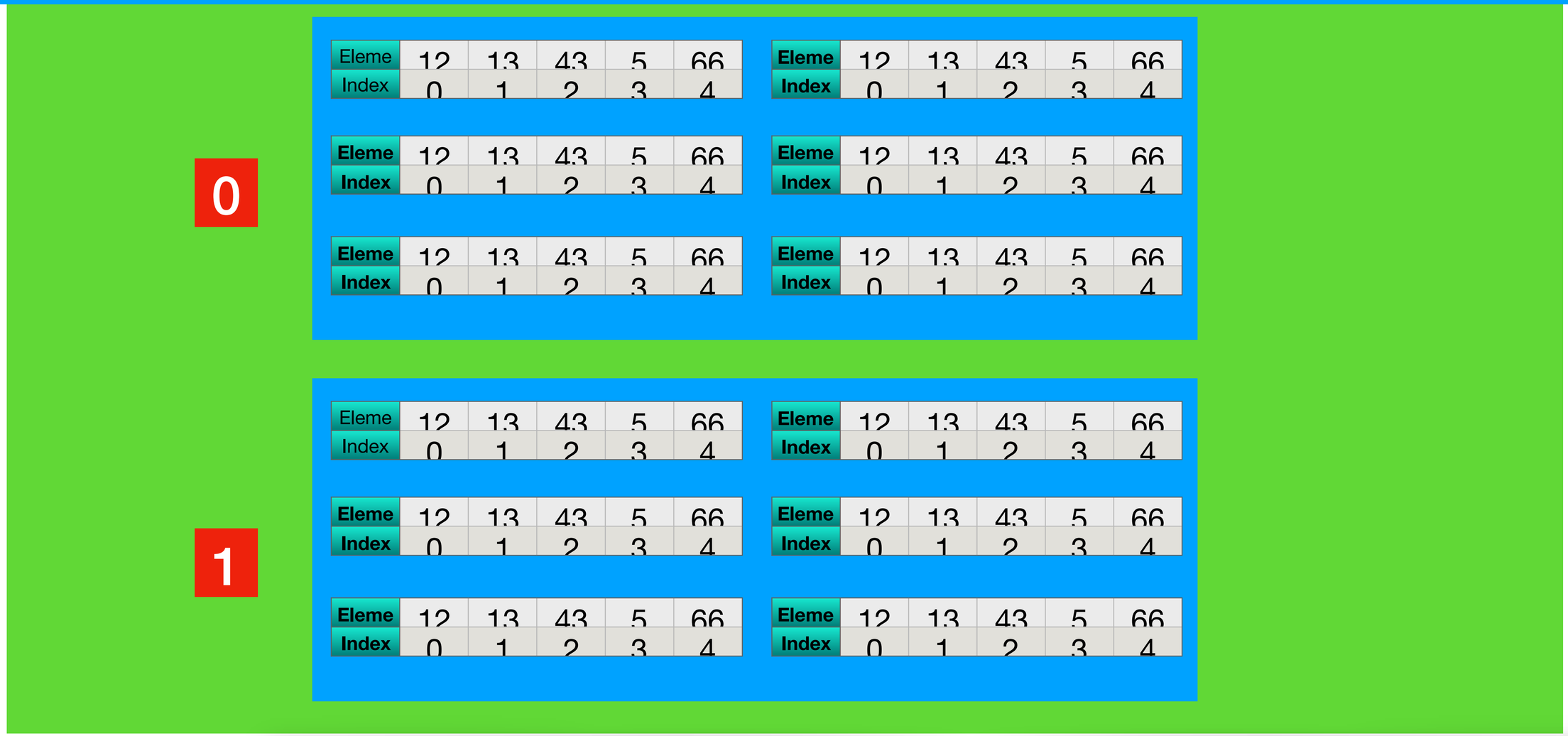
4

Element	12	13	43	5	66
Index	0	1	2	3	4

5

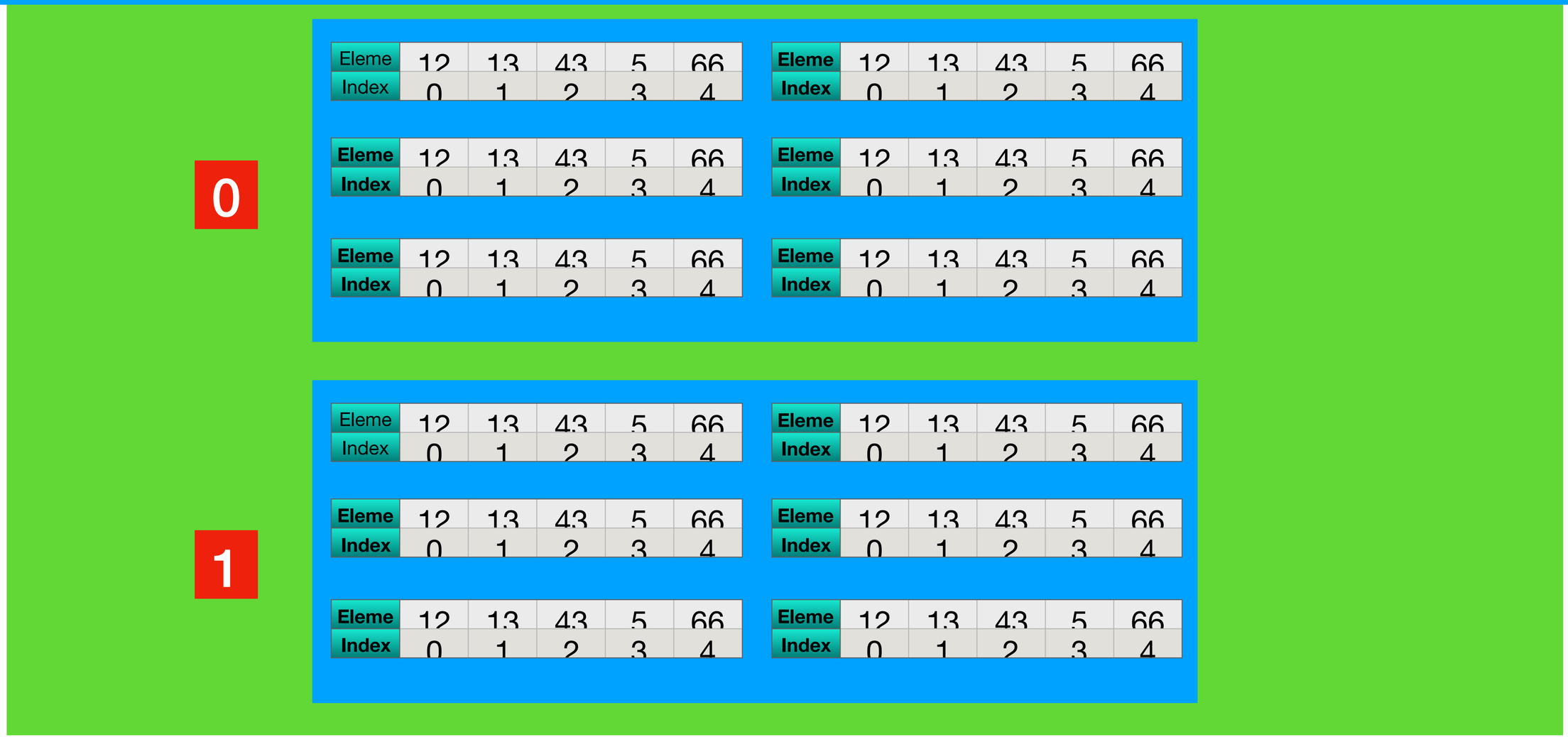
Element	12	13	43	5	66
Index	0	1	2	3	4

3 Dimensional Array



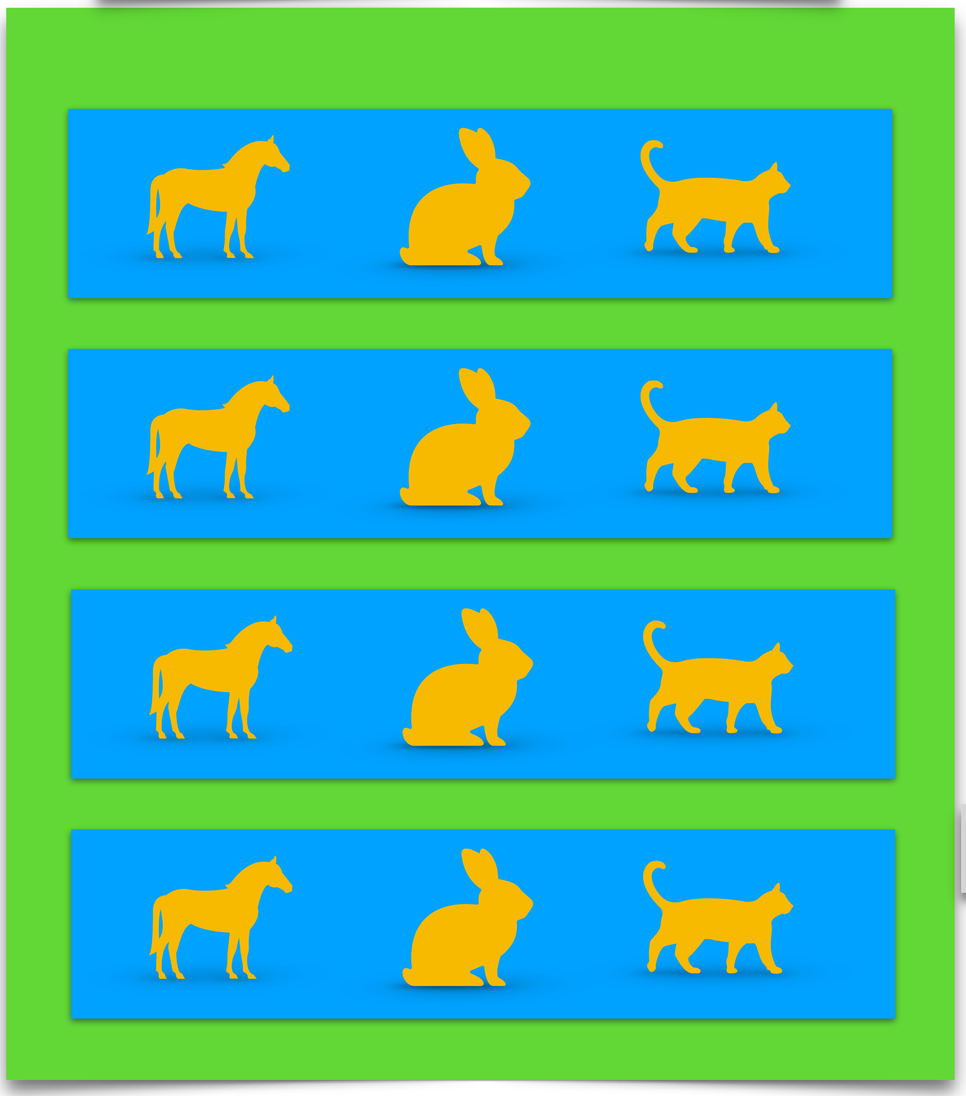

```
int[ ][ ][ ] ThreeDArray ={ArrayOf_IntArray, ArrayOf_IntArray };
```

```
int[ ][ ][ ] ThreeDArray ={ArrayOf_IntArray, ArrayOf_IntArray };
```



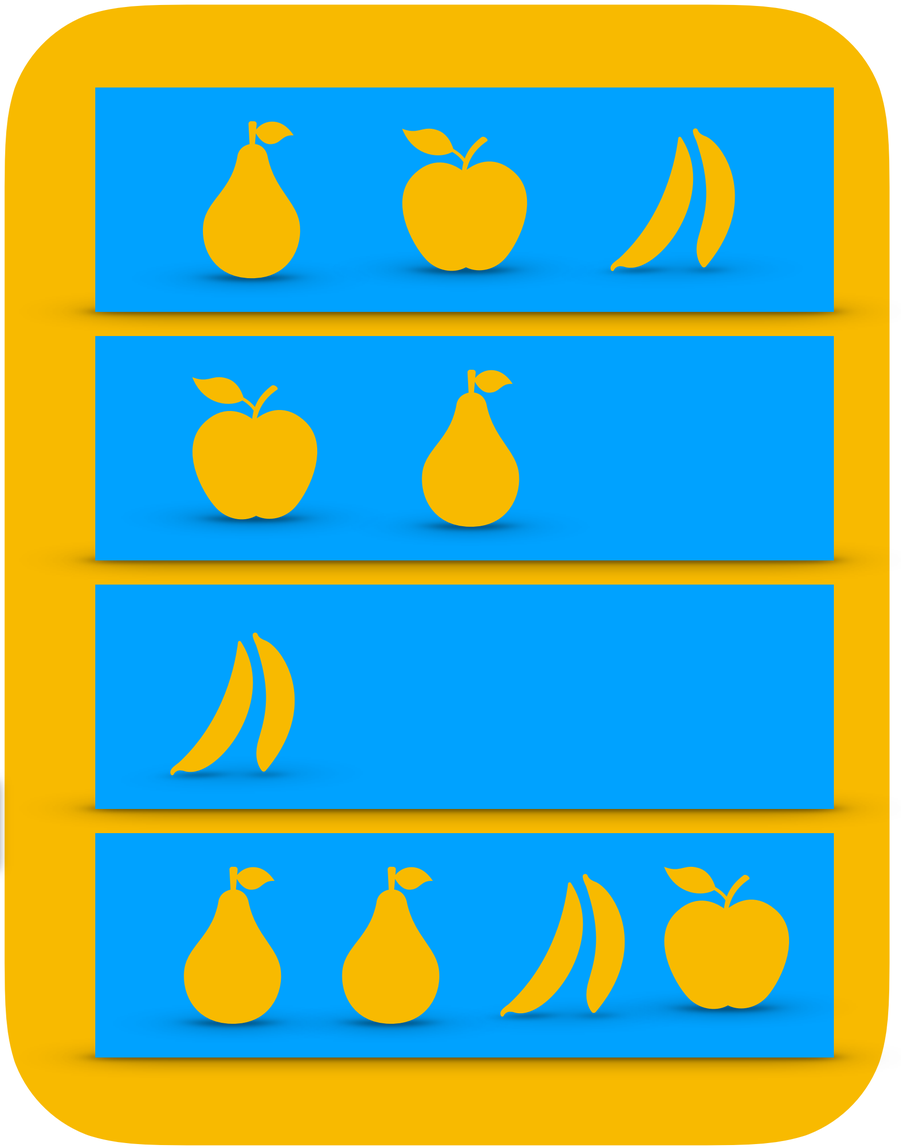
CYBERTEK

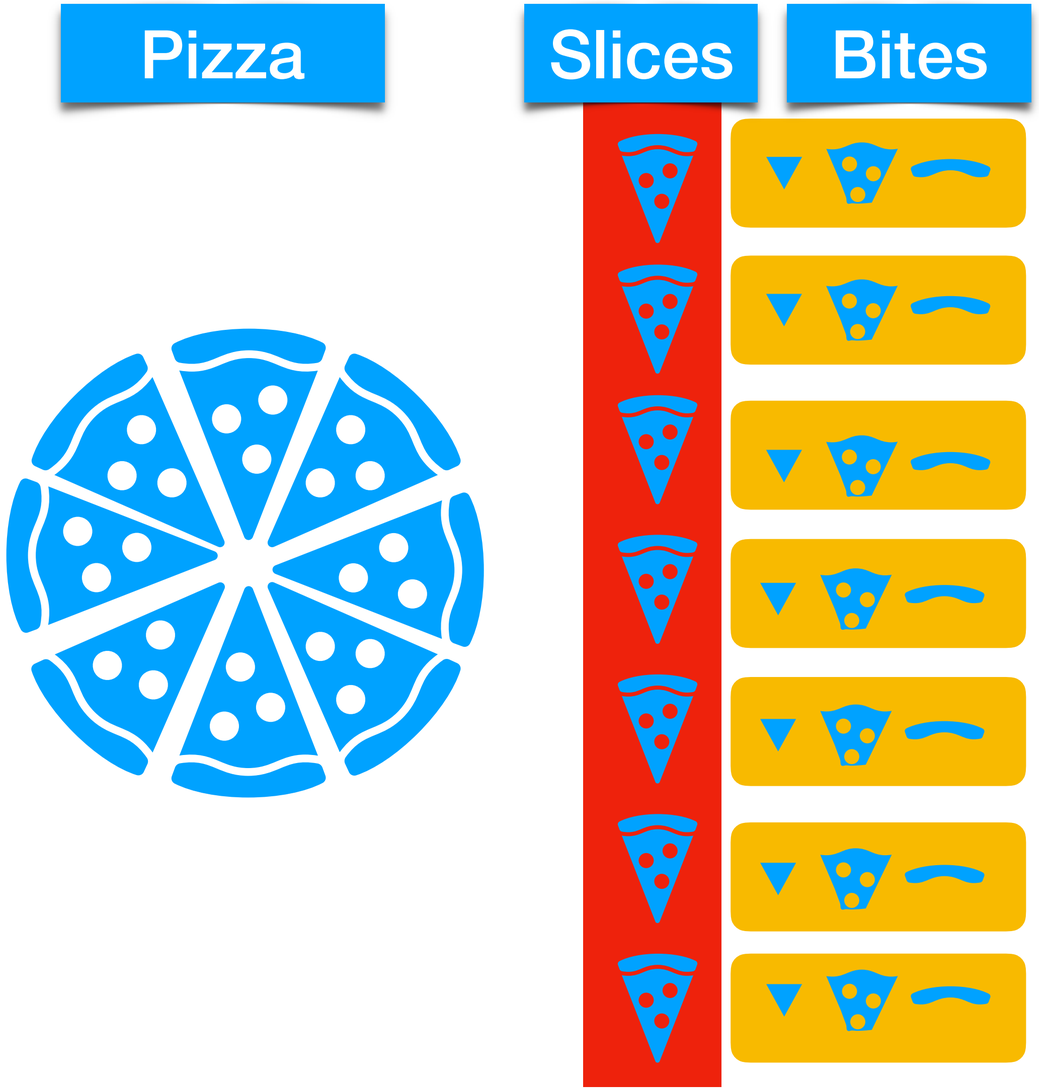
Animal[ ][ ] animalCages ;

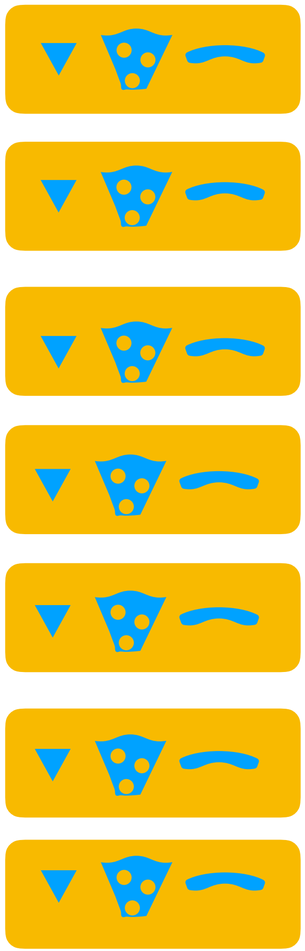


int[ ][ ] pizzaBites ;

Fruit[ ][ ] FruitBoxes ;




Pizza
Slices
Bites



2 Dimensional Array

Slices \ bites	0	1	2
0	00	01	02
1	10	11	12
2	20	21	22
3	30	31	32
4	40	41	42
5	50	51	52
6	10	11	12

CYBERTEK

2 Dimensional Array

Slices \ bites	0	1	2
0	00	01	02
1	10	11	12
2	20	21	22
3	30	31	32
4	40	41	42
5	50	51	52
6	10	11	12

```
//example for declaring 2 demential int array
// with size;

 int[][] intArray = new int[7][3] ;
 int[] intArray[] = new int[7][3] ;
 int intArray[][] = new int[7][3] ;

//example for declaring 2 demential
// String array with size;

 String[][] data = new String[7][3];
```

CYBERTEK

2 Dimensional Array- Excel

Columns / Rows	0	1	2
0	12	5	4
1	23	113	32

```
//example for declaring 2 demential int array
// with size and assigning value

int[][] data = new int[2][3] ;
  data[0][0] = 12;
  data[0][1] = 5;
  data[0][2] = 4;
  data[1][0] = 23;
  data[1][1] = 113;
  data[1][2] = 32;
```

2 Dimensional Array- Excel

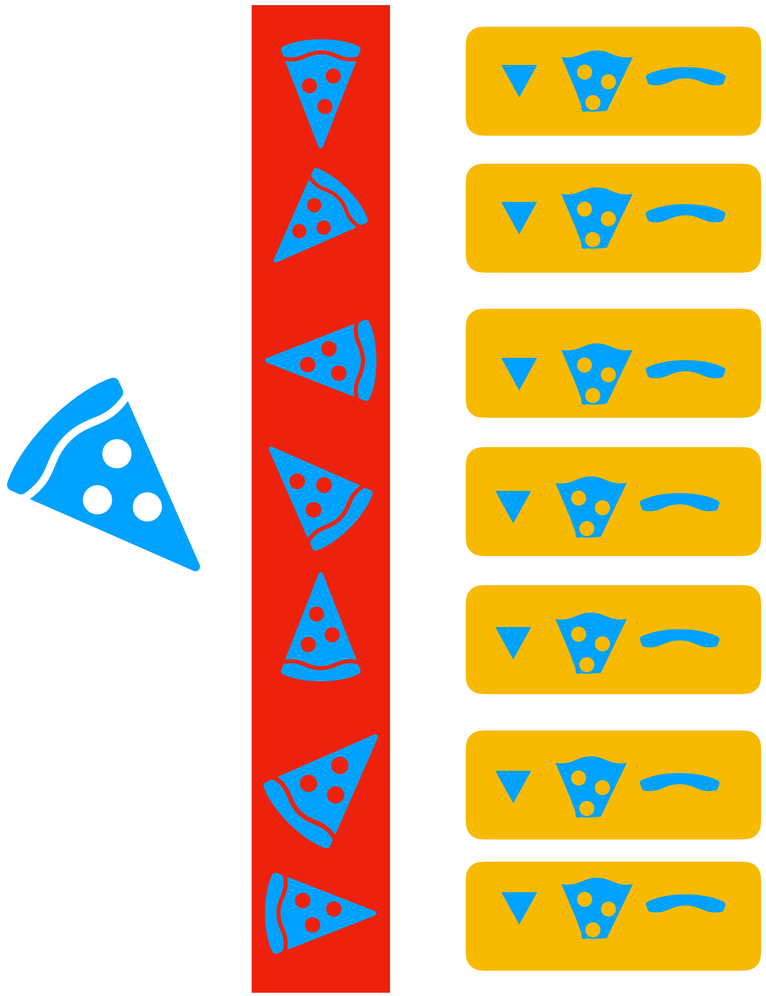
Columns / Rows	0	1	2
0	12	5	4
1	23	113	32

```
//example for declaring 2 dimensional int array
// with size and assigning value

int[][] data = new int[2][3] ;
  data[0] = new int[]{12,5,4};
  data[1] = new int[]{23,113,32};

// second square bracket size is optional

int[][] data = new int[2][] ;
  data[0] = new int[]{12,5,4};
  data[1] = new int[]{23,113,32};
```

2 Dimensional Array- Excel

Columns / Rows	0	1	2
0	12	5	4
1	23	113	32

```
//example for declaring 2 demential int array
// in one line and accessing them by index
  int[][] data ={ {12,5,4},{23,113,32} }.

  System.out.println(data[0][0]);
  System.out.println(data[0][1]);
  System.out.println(data[0][2]);
  System.out.println(data[1][0]);
  System.out.println(data[1][1]);
  System.out.println(data[1][2]);
```



2 Dimensional Array

Slices \ bites	0	1	2
0	00	01	02
1	10	11	12
2	20	21	22
3	30	31	32
4	40	41	42
5	50	51	52
6	10	11	12

- Mushroom
- Red Pepper
- Black Olive
- Spinach
- Green Pepper
- Banana Pepper

CYBERTEK

2 Dimensional Array- Excel

Columns / Rows	0	1	2
0	12	5	4
1	23	113	32

```
//example for declaring 2 demential int array
//in one line and accessing using for loop
  int[][] data ={ {12,5,4},{23,113,32} }.

  for (int i = 0; i < data.length; i++) {
       System.out.println("********Row : "+ i);

       for (int j = 0; j < data[i].length; j++) {
         System.out.print(data[i][j]+" ");
       }
       System.out.println();
     }
```

2 Dimensional Array- Excel

Columns / Rows	0	1	2
0	12	5	4
1	23	113	32

```
//accessing items using foreach loop
  int[][] data ={ {12,5,4},{23,113,32} };

  for (int[] rows : data) {
    System.out.println("*******Row : ");

    for (int columnData : rows) {
      System.out.print(columnData + " ");
    }
    System.out.println();

  }
```

2 Dimensional Array- Excel

Columns / Rows	0	1	2
0	12	5	4
1	23	113	32

2 Dimensional Array- Excel-1

Columns / Rows	0	1	2
0	12	5	4
1	23	113	32

3 dimensional array is collection of 2 dimensional arrays

```
// Declaring 2 two dimensional Array
  int[][] data1 ={ {12,5,4},{23,113,32} }
  int[][] data2 ={ {6,15,14},{20,3,88} }

  // Storing above 2 variable to another array to make it
  3 dimensional

  int[][][] threeDArray = {data1, data2}
  // or
  int[][][] threeDArray = new int[2][][];
  threeDArray[0] = data1;
  threeDArray[1] = data2;
```