List of Concepts Involved:

- Why Array?
- What is Array?
- How to create an array?
- 1d, 2d, 3d and regular jagged with memory

Topic 1: Why Array?

If we use a traditional approach, then to store 5 values we need to create 5 variables.
Similarly to store 100 values we need to create 100 variables.
The drawback in the traditional approach is that remembering the variables names is complex, so to avoid this problem we need to use **"Arrays"**.

Topic 2: What is Array?

It refers to index collection of fixed no of homogeneous data elements.
Single variable holding multiple values which improves readability of the program.

Topic 3: How to create an Array?

Array declarations

1. Single Dimension Array

Declaration of array

- int[] a;//recommended to use as variable is separated from type.
- int a[];
- int []a;
- int[6] a; // compile time error. we cannot specify the size.

Array Construction

Every array in java is an object hence we create using a new operator.

Example

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

- For every type corresponding classes are available but these classes are part of java language but not applicable at the programmer level.
  int[] [I
  float[] [F
  double[] [D

- For every type corresponding classes are available but these classes are part of java language but not applicable at the programmer level.
  int[] [I
  float[] [F
  double[] [D

Rule1

At the time of Array construction compulsorily we should specify the size.

Example:

```
int[] a=new int[5];
    int[] a =new int[];//ce:: array dimension is missing.
```

Rule2

It is legal to have an array with size zero.

Example:

```
int[] a =new int[0];
     System.out.println(a.length);// 0
```

Rule3

If we declare an array with negative size it would result in a Negative Array size exception.

Example:

```
int[] a=new int[-5]; //NegativeArraySizeException.
```

Rule4

The allowed datatypes to specify the size are byte,short,int,char.

Example:

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

        byte b=10;
        int[] a =new int[b];//valid

        short s=25;
        int[] a =new int[s];//valid

        char c='A';
        int[] a=new int[c];//valid

  int[] a=new int[10L];//CE
  int[] a=new int[3.5f];//CE
```

Rule5

The maximum allowed array size in java is the maximum value of int size.

```
int[] a=new int[2147483647]; //but valid:: OutOfMemoryError
   int[] a=new int[2147483648]; //CE
```

ArrayInitialisation

Since arrays are treated as objects,internally based on the type of data we keep inside array JVM will keep default values.

Example

```
int[] a =new int[5];
System.out.println(a);//[I@....
System.out.println(a[0]);//0
```

Example

```
int[] a=new int[4];
a[0]=10; a[1]=20; a[2]=30;
System.out.println(a[3]); //0
System.out.println(a[4]); //ArrayIndexOutOfBoundsException.
System.out.println(a[-4]);//ArrayIndexOutOfBoundsException.
```

Shortcut of way declaration,construction,initialisation in single line

```
int[]a = {10,20,30,40};
char[] a= {'a','e','i','o','u'};
String[] a= {"sachin","ramesh","tendulkar","IND"};
```

Array Element Assignments

case 1:

In case of primitive array as an array element any type is allowed which can be promoted to declared type.

```
int[] a=new int[10];
    a[0]=97;
    a[1]='a';
    byte b= 10;
    a[2]=b;
    short s=25;
     a[3]=s;
     a[4]=10L;//CE: possible loss of precession
```

case 2

In case of Object type array as an array elements we can provide either declared type object or its child class objects.

Example

```
Object[] obj=new Object[5];
obj[0] =new Object();//valid
obj[1] =new Integer(10);//valid
obj[2] =new String("sachin");//valid
```

case 3

In case of interface type array as an array element we can provide its implementation class Object.

Example

```
Runnable[] r=new Runnable[5];
r[0]= new Thread("sachin");
r[1]= new String("dhoni");//CE
```

case 4

In case of abstract class type array as an array element we can provide its child class Object.

Array variable assignment

Case 1

Element level type promotion is not applicable

Example

char value can be type promoted to int, but char[] can't be type promoted to int[].

```
int[] a= {1,2,3};
char[] c={'a','b','c'};
int[] b = a;
int[] a = c;//invalid
```

Case2

In case of an Object type array,its child type array can be assigned.

Example

```
String[] names={"sachin","saurav","dhoni"};
              Object[] obj=names;
```

- int[] => int[] (valid)
- char[] => int[] (invalid)
- int => long (valid)
- int[] => long[] (invalid)
- char => int(valid)
- char[] => int[] (invalid)
- String => Object (valid)
- String[] => Object[](valid)

Case3

Whenever we are assigning one array reference to another array reference,its just the references which are being copied not the array elements.

While copying the reference only its type would be given importance,not its size.

Example

```
int[] a= {10,20,30,40};
int[] b= {100,200};
a=b;
b=a;
```

Case4
Whenever we are copying the array,its reference will be copied but we should match it with the array dimension and its type, otherwise it would result in compile time error.

Example
```
int[][] a= {{10},{20},{30}};
int[] b={100,200,300};
b= a; //CE: incompatible type
```

Note:
In array assignment,its type and dimension must be matched otherwise it would result in compile time error.

- **int[] a ={10,20,30};System.out.println(a); //[I@..**
- **float[] f={10.0f,20.0f}; System.out.println(f); //[F@..**
- **boolean[] b= {true,false};System.out.println(b); //[Z@..**
- **Integer[] i={10,20,30}; System.out.println(i); //[L@...**
- **Float[] f = {10.0f,20.0f};System.out.println(i); //[L@...**

2-D Array

2D-Array =1D-Array + 1D-Array
(ref) (data)

Declaration(All are valid)
```
int[][] a ;
 int  a[][];
 int  [][]a;
 int[] []a;
 int[] a[];
 int []a[];
```

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

ArrayInitalisation
```
a[0][0] = 10;
 a[2][3] = 5;
```

Tricky Question

- int[] a,b; // a-> 1D,b->1D
- int[][] a,b; //a->2D,b->1D
- int[] a[],b; //a->2D,b->1D
- int[] a[],[]b; //CE
- int[] []a,b; //a->2D,b->1D

Rule

If we want to specify the dimension, we need to specify that before the first variable,but from the second variable onwards rule is not applicable.

shortcut of working with 2-D array

```
int[][] a= {{10,20},{100,200,300},{1000}};
```