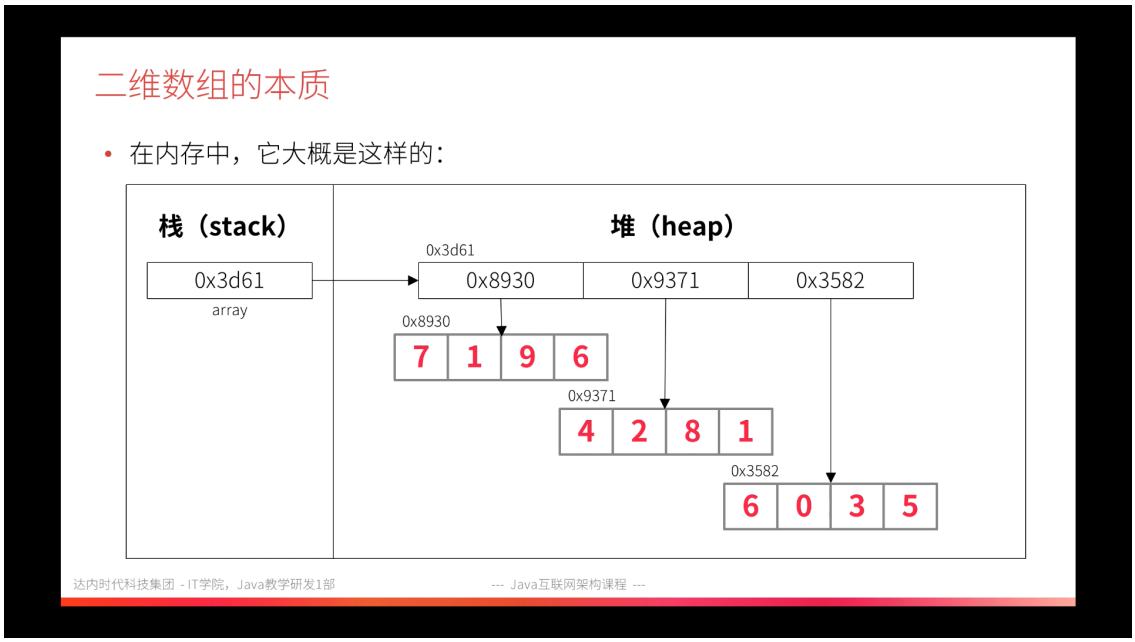
1. 二维数组用于表示“矩阵”类的数据。
2. 二维数组的元素可以是其他类型，但类型必须统一。
3. 二维数组的静态初始化与使用二维数组的元素：

```
int[][] array = {{7, 1, 9, 6}, {4, 2, 8, 1}, {6, 0, 3, 5}}; //二维数组的静态初始化
System.out.println(array[0][2]); //使用二维数组的元素
```

4. 二维数组的本质：将多个一维数组作为另一个一维数组的元素。
5. 二维数组的内存图：

> 堆栈存什么?
>
> 存在哪?堆中存new出来的对象(包括成员变量)、栈中存局部变量(包括方法参数)
>
> 装什么?引用数据类型变量中存的是地址，基本数据类型变量中存的是值



6. 在二维数组中，每个小数组的长度可以短一点，比如：

```
int[][] array = {{7, 1}, {4, 2, 8, 1}, {6, 0, 3}};
```

   但在绝大部分情况下，二维数组的每个小数组的长度是相同的。

7. 二维数组的动态初始化：

```
int[][] array = new int[3][4];
```

   初始化了一个长度为3的一维数组，且元素都是数组

   每个元素数组的长度都为4

8. 由于二维数组的本质仍然是一个一维数组，所以二维数组在初始化时也是在初始化一维数组，那么，在不确定“第二维度”的数组的元素数量时，创建数组对象的第2个中括号可以留空。

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

9. 如果在动态初始化时第2个中括号可以留空了，那么在使用这个二维数组之前，必须对这个数组再进行初始化，才可以使用：

```
int[][] array = new int[3][];
//相当于创建了一个名为array的一维数组，长度为3
//array数组元素均为null

array[0] = new int[2];
//array数组的第一个元素是数组，长度为2
//这个数组没有名字，只能通过array[0]来表示他
//这个数组的2个元素的值均为0

array[0][0] = 7;
array[0][1] = 1;

array[1] = new int[]{4, 2, 8, 1};
```

10. 二维数组的遍历：

    使用嵌套循环，先遍历第一维度，再遍历第二维度。

    循环可以是任何一种：

    - while
    - do...while
    - for
    - 增强for(foreach)

    内外层的循环类型可以不统一，比如外层while内层for

    一般内外层循环都用for

```
int[][] array = {{7, 1, 9, 6}, {4, 2, 8, 1}, {6, 0, 3, 5}};
for (int i = 0; i < array.length; i++) {
    for (int j = 0; j < array[i].length; j++) {
        System.out.println(array[i][j]);
    }
}
```

11. 增强for循环(foreach)——一维数组：

```
int[] numbers = {10, 20, 30, 40, 50};
for (int x : numbers) {
    System.out.println(x);
}
```

12. 增强for循环(foreach)——二维数组：

```
int[][] array = {{7, 1, 9, 6}, {4, 2, 8, 1}, {6, 0, 3, 5}};
for (int[] i : array) {
    for (int j : i) {
        System.out.println(j);
    }
}
```