4-arrays-and-lists

Contents

Arrays

Basic declaration of an array is of the form `type var-name[]` while to define one we have to also initialize it to an object in memory such as `var-name = new type[size]`.

```
    int integers[];
    integers = new int[100]; // create an array of 100 primitives and assign
       the object reference to `integers`
```

Initialization lists for arrays, are represented as elements surrounded by curly braces, compared to String literals however the Initialization lists are not interned, meaning that no special pool is created, if we create the same object twice they will not have the same reference like String literals would, even though the initialization list is defined and known at compile time. Arrays are also mutable, we can modify the individual elements of the array, while we can not modify the length, to do that a new one has to be created instead.

```
    int month_days[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
    int month_days2[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
    assert(month_days == month_days2) // false, they are not like string
       literals, no interning for initialization lists
```

```
The elements in the array allocated by new will automatically be initialized
   to:
    -   zero for numeric types,
    -   null for reference types
    -   false for booleans types
```

Multidimensional arrays can be defined by stacking square brackets, in this case as many as we want, there is really no limit to how many dimensions we create or define for an array where the minimum is 1

```
    int array0[][] = new int[][]; // invalid, compile time error, there is
       nothing to `new` here, no size specified
    int array1[][] = new int[4][5]; // 4 rows, by 5 columns, initialized at
       0, the elements per row are fixed to 5
    int array2[][] = new int[4][]; // 4 rows, but each row points to null,
       have to manually initialize the second array per row
    int array3[][][] = new int[4][][]; // 4 rows, but the inner dimensions
       are not initialized yet, do this manually per row
```

```
array3[0] = new int[1][1] // the 0-th row is initialized to a two-dim
   array with sizes 1x1
```

Initialization lists for multi dimensional arrays are just like regular ones, where each entry is a list itself, and `interning is NOT done !`

```
int array[][] = {
    { 0*0, 1*0, 2*0, 3*0 },
    { 0*1, 1*1, 2*1, 3*1 },
    { 0*2, 1*2, 2*2, 3*2 },
    { 0*3, 1*3, 2*3, 3*3 }
}
```

Alternate declaration syntax such as `type[] var-name`, allows us to declare multiple arrays on the same line with the following example

```
int nums[], nums2[], nums3[]; // all variables declared on this line are
   arrays
int[] nums, nums2, nums3; // all variables declared on this line are
   arrays
int nums, nums2[], nums3; // only the nums2 on this line is actually an
   array
```