

Key Points – Arrays and Vectors

We have explored a variety of new ways to work with data — namely through array data structures or using Java's `Vector` class. Here are the key points we have learned:

- Arrays are useful data structures to store a collection of data of the same type.
- Java supports arrays of objects (actually object references) as well as arrays of primitive data types.
- Elements in an array are accessed through an integer index. (For example, if `inventory` is an array and `i` is an integer, `inventory[i]` is the i^{th} element in the array.)
- For each array a public data field named `length` holds the length of the array. (For example, if `gizmo` is an array, `gizmo.length` is the length of the array.)
- A common error in using arrays is attempting to access an element that does not exist. This error is not caught by the compiler; rather, it causes an "array index out of bounds" run-time error.
- Valid array indices range from 0 to "one less than" the length of the array. (For example, if `key` is an array the last element in the array is `key[key.length - 1]`.)
- An array can be initialized when it is declared using an *initialization list*. (For example, `String[] medals = { "GOLD", "SILVER", "BRONZE" };`)
- An array is an object, however its implementation as an object is through the compiler and the run-time interpreter.
- Java supports multi-dimensional arrays, for example, to represent matrices.
- An array is a *static* data structure. Once an array is declared, its size cannot change.
- Java supports *dynamic* arrays through its `Vector` class.
- A `Vector` class object is a dynamic array of objects. Its size increases and decreases automatically as elements are removed or added.
- Objects stored in or retrieved from a `Vector` object are of the `Object` class.
- If an object retrieved from a dynamic array (a `Vector` object) must be cast before it can be assigned to an object variable (unless the object variable is of the `Object` class.)