Arrays

Array of Native Type Values

- Creating an array of native type values includes two steps:

  Declaring the variable that should hold the reference for the array (the object).

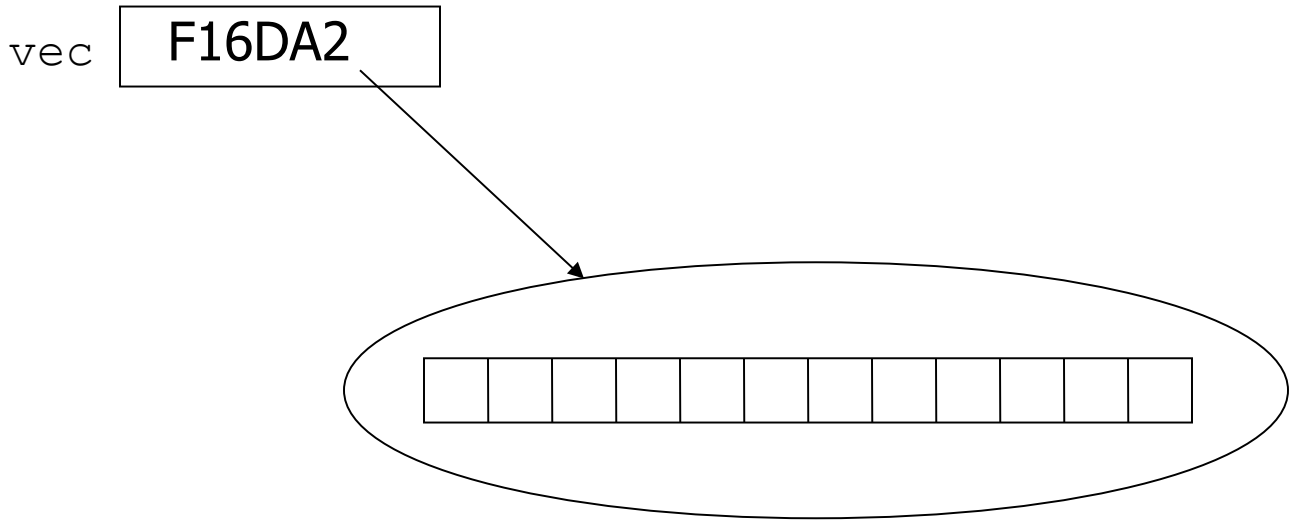
```
int vec[];
```

  Creating an array (an object) and assign its reference to the vec variable.

```
vec = new int[12];
```

Array of Native Type Values

- The result of these two statements is:



Arrays of Native Type Values

- Each and every value is stored in a cell. Every cell has an index number. The indexing starts at 0.

```
vec[0] = 12;
vec[2] = vec[0] + 3;
```

Arrays of Objects

- Creating an array of objects includes the following two steps:

  Declaring the variable that holds the reference to the array (the object).

  ```
  Student vec[];
  ```

  Creating the object (creating the array).

  ```
  vec = new Student[12];
  ```

Arrays of Objects

- When the array is created it holds `null` in each one of its cells.
- Instantiating the objects and place their references in the array's cells can be considered as the third step.

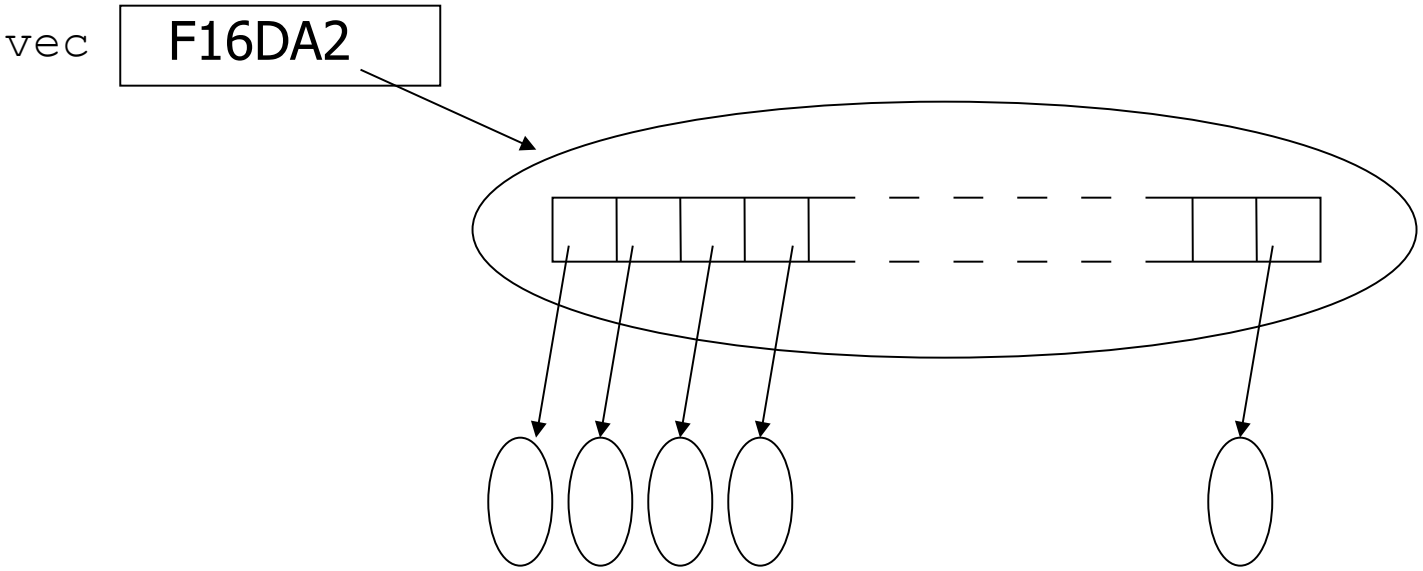
Arrays of Objects

- We can create all objects using a simple loop and place their references in the array's cells.

```
for(int i=0; i<12; i++)
{
     vec[i] = new Student();
}
```

Arrays of Objects

- The result of these three statements can be described using the following diagram:



Arrays of Objects

The Detailed Syntax

```
Student vec[];
vec = new Student[3];
vec[0] = new Student(“Moshe”);
vec[1] = new Student(“David”);
vec[2] = new Student(“Ramy”);
```

The Short Syntax

```
Student vec[] = {   new Student(“Moshe”),
                    new Student(“David”),
                    new Student(“Ramy”)    };
```

The Square Brackets Position

- The square brackets can be placed either before the variable name or after it.
- Placing the square brackets before or after has a a different meaning.

```
int vec[], number1, number2;
```
— number1 and number2 are simple variables

```
int []vec, number1, number2;
```
— number1 and number2 are variables that can hold references for arrays

Copying Array Values

- In order to copy the values of one array to an other one you should use the method `System.arraycopy()`

```
public static void arraycopy(
                Object src,
                int src_position,
                Object dst,
                int dst_position,
                int length                    )
```

Multi-Dimensional Array

- A multi-dimensional array is an array of arrays. There are two ways for creating multi-dimensional arrays:

Detailed Way

```
int matrix[][];
matrix = new int[3][];
matrix[0] = new int[4];
matrix[1] = new int[4];
matrix[2] = new int[4];
```

Short Way

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

The `length` variable

- Each and every array has a variable that its name is length. This variable holds the size of the array.

```
int vec[] = {12,32,42,55};
for(int i=0; i<vec.length; i++)
{
    System.out.println(vec[i]);
}

int mat[][] = {{1,2,3,7,8}, {3,2}, {4,6,5,5}};
```

The `for` Loop

- When there is a need to iterate the items an array or a collection holds, we can use the `for` loop as if it was a foreach loop.