CS133: Developing Programming Principles

Lecture 11

Array Introduction,
Arrays as parameters

Arrays

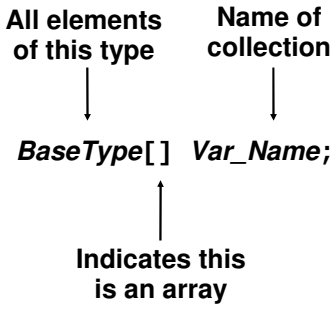
- A special kind of object that stores a collection of variables.
- All variables are of the same type.
- The number of entries is fixed when the array is created.
- Individual elements can be accessed and updated.

Using an array

1. Choose a name and "base" type (declaration)
2. Set size of array (allocation of memory)
3. Set values of entries (initialization)

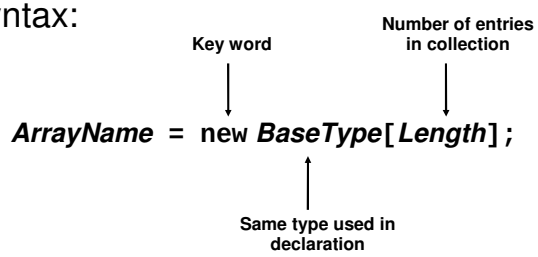
Declaring an array

- Syntax:



Allocating memory

- Syntax:



Note: Declaration and allocation can be combined

***BaseType*[] *ArrayName* = new *BaseType*[*Length*];**

Initializing values

- When array is created, individual values are not initialized.
- Entries should be initialized before being used:

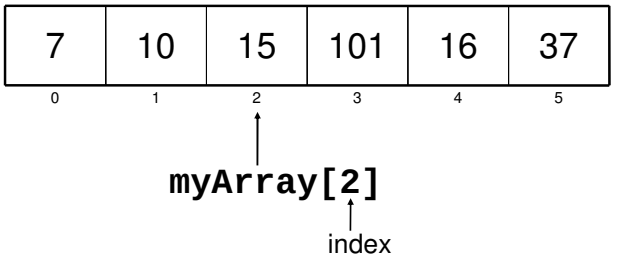
***ArrayName*[*i*] = *Expression*;**

0 <= i < length **Of type BaseType**

Example

```
int[] myArray = new int[6];
myArray[0] = 7;
myArray[1] = 10;
myArray[2] = 15;
myArray[3] = 101;
myArray[4] = myArray[2] + 1;
myArray[5] = 37;
```


Example continued



Length of **`myArray`** is 6.

Accessing arrays

- The first index is 0.
- The last index is length-1.
- Treat **`myArray[n]`** as a variable of the array's type (the base type).

The length of an array

- We can get the length of an array after it was created.

***MyArray*`.length`**

Name of array **Notice: no brackets**

Example of arrays

- Create an array, of size 100, of boolean values, which stores whether or not the integer index is a prime.
- For example:
  `prime[4]` should be **`false`**
  `prime[5]` should be **`true`**

Example continued

1. Declare array
   ```
   boolean[] prime;
   ```
2. Allocate memory
   ```
   prime = new boolean[10];
   ```
- Note: valid entries are **`prime[0]`**, **`prime[1]`**, …, **`prime[9]`**

Looping

```
// Set all entries to false
for (int i = 0; i < 10; i++) {
  prime[i] = false;
}
// Set primes true
prime[2] = true;
prime[3] = true;
prime[5] = true;
prime[7] = true;
```


Iterating through an array

- General format:

```
for (int i = 0; i < myArray.length; i++)
{
   // code
}
```

Iterating through an array example

- Print out values:

```
for (int j = 0; j < prime.length; j++)
{
  if (prime[j]) {
   System.out.println(j + " is prime.");
  } else {
   System.out.println(j + " is not prime.");
  }
}
```


Passing an array as a parameter

- Declare array in parameter list in “usual way”:
  `public` ***ReturnType MethodName*****(*****BaseType*****[] arrayName)**
- BaseType can be primitive or class name
- Use array in method in “usual way”

calcPrimeArray

```
// pre: val > 2, pList != null, pList.length >= val
// post: sets pList[val] == true iff val is prime
public void calcPrimeArray(boolean[] pList, int val)
{
  int upperBound = (int)(Math.sqrt(val) + 1);
  int factor = 2;
  pList[val] = true;
  while (factor <= upperBound) && pList[val]) {
     if (pList[factor] && (val % factor == 0)) {
        pList[val] = false;
     }
     factor++;
  }
}
```


Array parameters

```
boolean[] prime = new boolean[100];
// Initialize element to false as before
// Set first few entries manually
prime[0] = false;
prime[1] = false;
prime[2] = true;
// Set rest of entries using help
for (int i = 3; i < prime.length; i++) {
  calcPrime(prime, i);
}
```

`ArrayIndexOutOfBounds` Exception

- If you specify an index that is larger than (length-1) or less than zero, you get this error.
- Remember: **`myArray.length`** is **not** a valid index.

Declaring and allocating arrays of objects

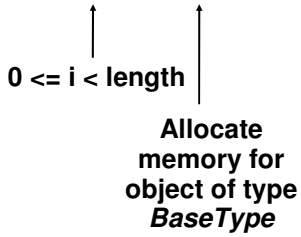
***BaseType*`[]` *ArrayName* =**
`new` *`BaseType`*`[`*`Length`*`];`

Class name

- Note: memory allocated for array elements but individual entries are not initialized.

Initializing array entries

***ArrayName*[*i*] = new BaseType(*Arguments*);**



Array of objects example

- Given: **`Date(int y, int m, int d)`** constructs a Date object with given year, month and day.
- Create an array of birthdays:

```
Date[] birthday = new Date[5];
birthday[0] = new Date(88, 9, 14);
birthday[1] = new Date(87, 1, 11);
birthday[2] = new Date(88, 6, 18);
birthday[3] = new Date(86, 5, 13);
birthday[4] = new Date(87, 11, 12);
```

Example: initialize birthday array

```
Date[] birthday = new Date[5];
Scanner in = new Scanner(System.in);
for(int i = 0; i < birthday.length; i++)
{
  int yr = in.nextInt();
  int mon = in.nextInt();
  int day = in.nextInt();
  birthday[i] = new Date(yr, mon, day);
}
```


Example: counting September birthdays

```
int septCount = 0;
for(int i = 0; i < birthday.length; i++)
{
  if (birthday[i].getMonth() == 8)
    septCount++;
}
```

- Where **`public int getMonth()`** is a method in **`Date`** that returns a number representing the month.

ERROR: undefined
OFFENDING COMMAND:

STACK: