Arrays

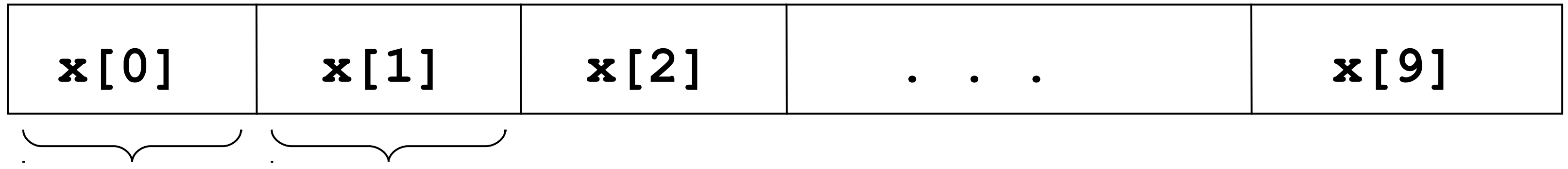
James Brucker

Arrays

An array is a series of elements of the same type, which occupy consecutive memory locations.

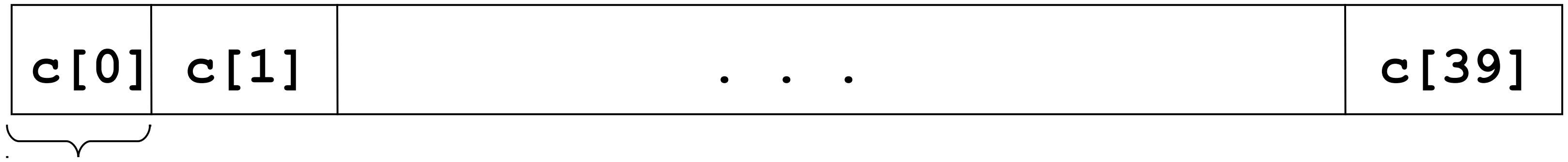
```
float[] x = new float[10];   // array of 10 float vars
char[] c = new char[40];     // array of 40 char vars
```

Array **`x[]`** in memory:



4 Bytes = size of float

Array **`c[] = new char[40]`** in memory :



2 Bytes = size of char

Array is an Object

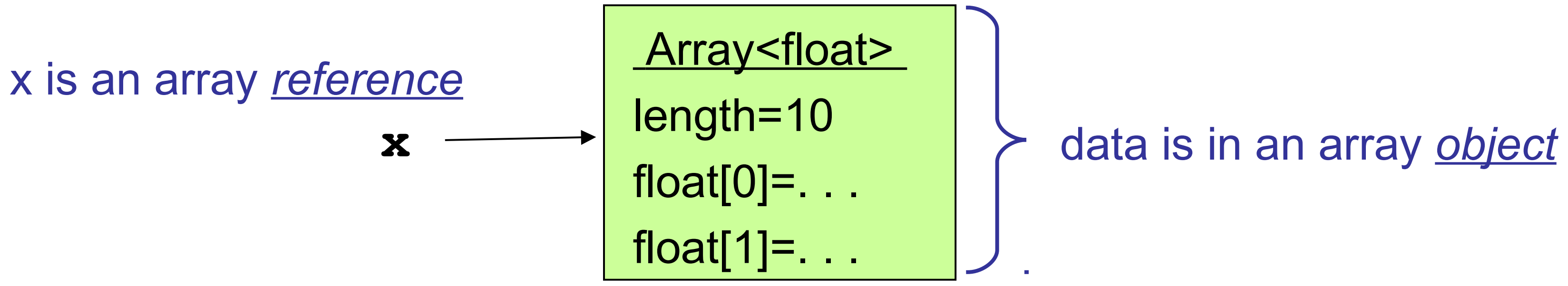
In C, C++, Fortran, etc., an array is just a collection of sequential memory locations (as in previous slide).

`x[0]`	**`x[1]`**	**`x[2]`**	. . .	**`x[9]`**

In Java and C#, an array is an ***object***.

This means an array can have methods and attributes, such as the length of the array

Array **`x[]`** in Java is an ***object***: it *encapsulates* data.

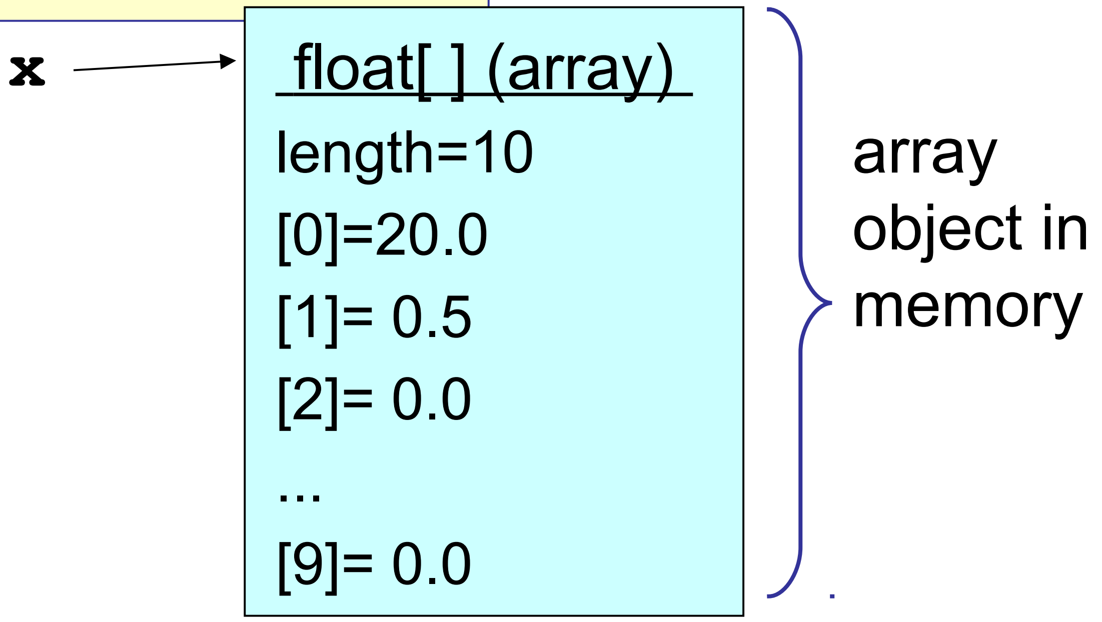


Structure of an array

The first element has index 0.

An array has a fixed length (size cannot be changed).

```
float[] x = new float[10];
x[0] = 20;
x[1] = 0.5F;
```



Array knows its own size!

Every array has an *attribute* named **`length`**

```
    double[] x = new double[20];
    x.length   // returns 20
```

`x.length is 20.`

The *first* element is **`x[0]`**,

the ***last*** element is **`x[x.length -1]`**.

Don't forget -1 !

In Java, an array is an ***object***.

`length` is a **property** (attribute) of the array object.

Why Use Arrays?

- Make it easy to process lots of data using loops.
- Perform operations on vectors and matrices.

Examples are given in later slides.

3 Steps to create an array

There are 3 steps to define & initialize an array.

Memorize them! A common programming error is to omit one of these steps.

1. Define array variable (reference)	`double[ ] x;`	`String[ ] colors;`

2. Create the array & specify its size.	`x =`
	`new double[10];`
		`colors =`
		`new String[3];`

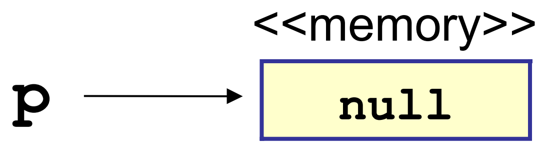
3. Assign values to array elements.	`x[0] = 10.0;`
	`x[1] = 0.5;`
	`. . .`
		`colors[0] = "red";`
		`colors[1] = "blue";`
		`colors[2] = "green";`

1. Define array reference

Declare p as type "array of int".

OK to omit space after "int" and between [ ].

```
int [] p;
```



This creates an array *reference* p,
but does not create an array.

p does not refer to anything yet!
Just like:

```
        String s;
```

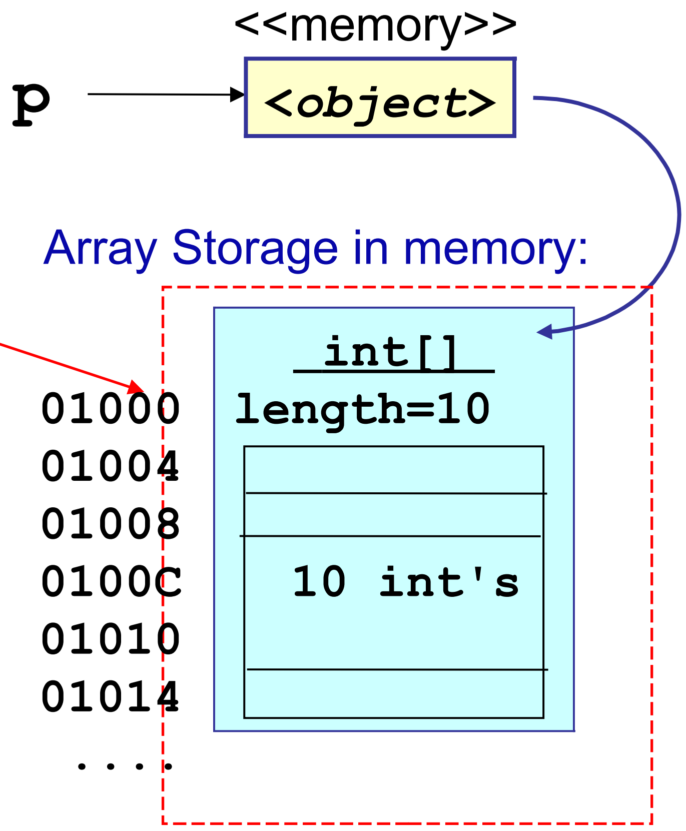
defines a String *reference* but
does not create a string.

2. Create the Array object

Create the array using "new".

```
array = new DataType[ size ]
```

```
p = new int[10];
```

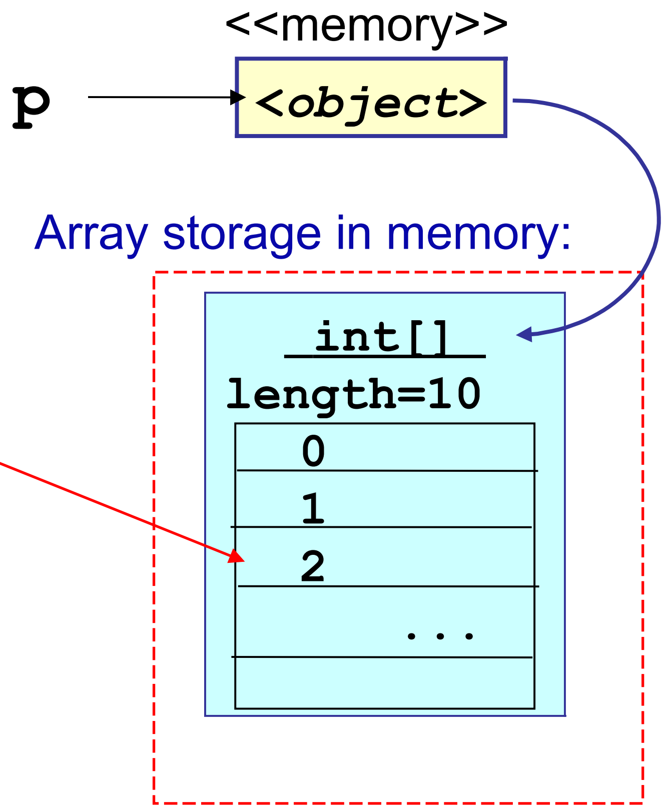


"new" creates a new object.
Here, it creates an *array* containing 10 "int" values.
It sets p to *refer* to this object.

3. Initialize elements of the array

When you create the array, Java does not initialize the array elements. You must do this.

```
for(int k=0; k < 10; k++)
    p[k] = k;
```



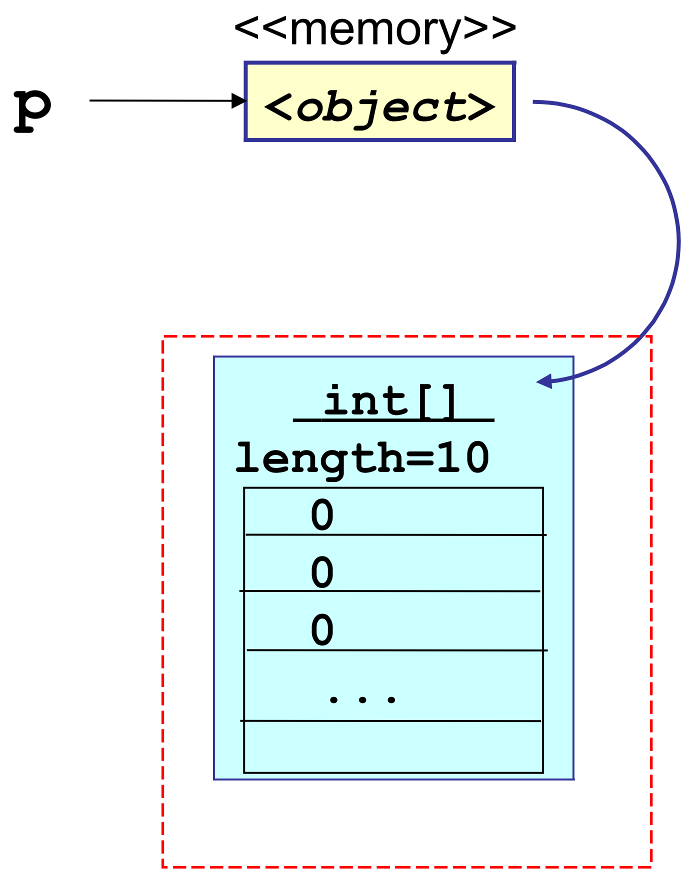
You can initialize array elements any way you like.

Some examples in later slides.

Short-cut to create an Array

You can combine steps (1) and (2) into one statement:

```
int[] p = new int[10];
```



This statement does two things:

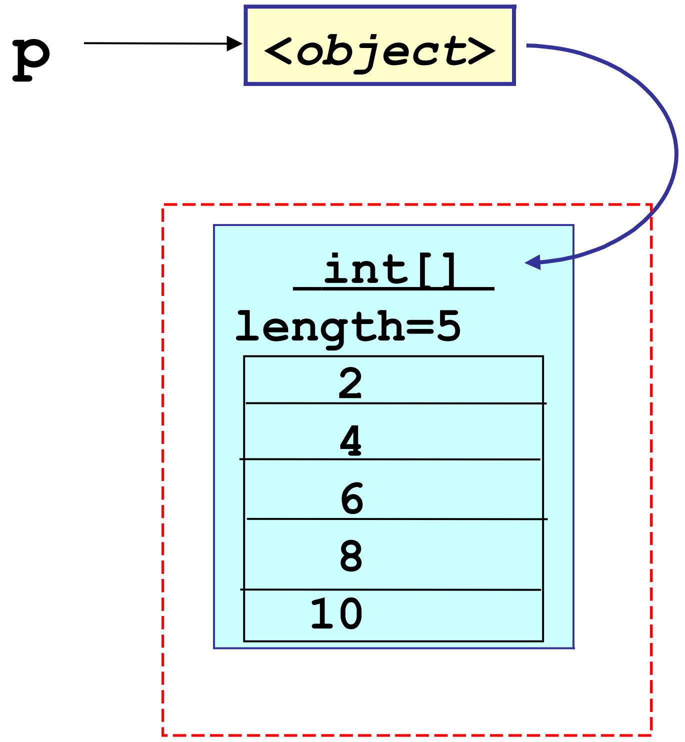
1) define p as an array reference

2) create an array with 10 elements and assign it to p

Another short-cut

If you have fixed values to put in the array, you can combine steps 1 - 3 into one statement:

```
int[] p = { 2, 4, 6, 8, 10};
```



This statement does 3 things:

1) define p as an array reference
2) create array with 5 int's
3) stores values 2, 4, ... 10 in the array

Summary: steps to create array

1. Define an array reference:

```
double [] x;
```

2. Create the array (allocate storage for elements) :

```
x = new double[10];
```

3. Assign values to the array elements:

```
for(int k=0; k<x.length; k++) x[k] = 2*k;
```

Short-cut: define array reference and create object

```
double[] x = new double[10];
```

Meaning of `[]` in "`String[] x`"

The **`[ ]`** means *"array of ..."* *or "... array"*.

- **`int[]`** means "*int array*" or "*array of int*".
- **`Foo[]`** means "*Foo array*" or "*array of Foo*".

`int[ ] x;`	x is <u>type</u> "int array"
`main(String[] args)`	args is <u>type</u> "String array"
`char[] c = {'c','a','t'}`	c is <u>type</u> "char array"
`double[] getScores()`	getScores returns

"array of double"

Error: invalid array index

The elements of an array **a** have indices from **0** to **a.length - 1**.

If a program tries to access an array using an invalid index, then Java throws an

`ArrayIndexOutOfBoundsException`.

```
double [ ] a = new double[10];
a[10] = 1.0;
```

```
Exception in thread "main"
    java.lang.ArrayIndexOutOfBoundsException: 10
        at MyProgram.badAccess(MyProgram.java:82)
        at MyProgram.main(MyProgram.java:68)
```

Inspect an array using BlueJ

Demo in class.

Use BlueJ to see inside an array
(called *inspection*)

Read Data into an Array

Suppose we want to read some *words* from the input into an array. Maybe we know that the input will never contain more than 100 words. We could write...

```
// create Scanner to read input
Scanner input = new Scanner( System.in );
// create array of words (Strings)
String [ ] words = new String[100];
// read the data
int count = 0;
while(input.hasNext() && count < words.length)
{   words[count] = input.next( );
    count++;
}
// now count = number of words actually read
```

Sort Data in an Array

`java.util.Arrays -` provides static methods for arrays.

One method is: **`Arrays.sort( array[] )`**

```
/** Sort the words[ ] array from last slide */
/** You must "import java.util.Arrays".     */

Arrays.sort( words );
```

Input data

```
dog
cat
frog
DOGS
ANT
```

$\xrightarrow{\text{Arrays.sort( )}}$

Result:

```
words[0] = "ANT"
words[1] = "DOGS"
words[2] = "cat"
words[3] = "dog"
words[4] = "frog"
```

Sort part of an Array

The previous slide is not quite correct.

Since we only have data for part of the array, we should sort only that part. Use:

`Arrays.sort(array[ ], start_index, end_index)`

```
// sort elements 0 until count (exclusive)
Arrays.sort( words, 0, count );
```

This sorts only the elements

```
  words[0] words[1] ... words[count-1]
```

Output the Elements of an Array

Now lets print the values of the array.

```
// write a loop to display each array element

for( int k=0; k < count ; k++ )
    System.out.printf("%d: %s\n", k, words[k]);
```

Output:

```
0:  ANT
1:  DOGS
2:  cat
3:  dog
4:  frog
```

An Array of Fibonacci Numbers

This example shows how to process all elements of an array.

The important point is that the "for" loop starts at **`k=0`** and tests **`k < fib.length (false`** when **`k=fib.length)`**

```
final int ARRAYSIZE = 20; // a constant value
long [ ] fib = new long[ ARRAYSIZE ];
fib[0] = fib[1] = 1;
for(int k = 2; k < fib.length; k++ )
   fib[k] = fib[k-1] + fib[k-2];

// output the values
for(int k = 0; k < fib.length; k++ )
   System.out.printf("f[%d] = %d\n",k, fib[k]);
```

Array as parameter

Use the same syntax as declaring an array variable.

```
/** Print the array elements. */
public void printArray( String[] array ) {
    for(int k=0; k< array.length; k++)
        System.out.printf("[%d] = %s\n",
                          k, array[k] );
```

```
/** Return maximum element in array. */
public double max( double[] array ) {
    double max = array[0];
    for(int k=1; k<array.length; k++) {
        if (array[k] > max) max = array[k];
    return max;
}
```

`main` has String array parameter

The main method accepts array of Strings.

```
/** args = command line arguments */
public static void main( String[] args ) {
    for(int k=0; k < args.length; k++)
        System.out.printf("args[%d] = %s\n",
                          k, args[k] );
```

The parameters to `main` are strings given on command line when running the class in the JVM.

For example:

```
cmd>  java MyClass hi there
args[0] = "hi"
args[1] = "there"
```

Method can return an array

A method can return an array:

```
/** Create an array and fill it with "1" */
static double[] makeOnes(int size) {
    double x = new double[size];
    // use Arrays.fill() is better
    for(int k=0; k<size; k++) x[k] = 1.0;
    return x;
}
```

Avoid this Common Mistake!

What does "`b = a`" do?

What will be printed?

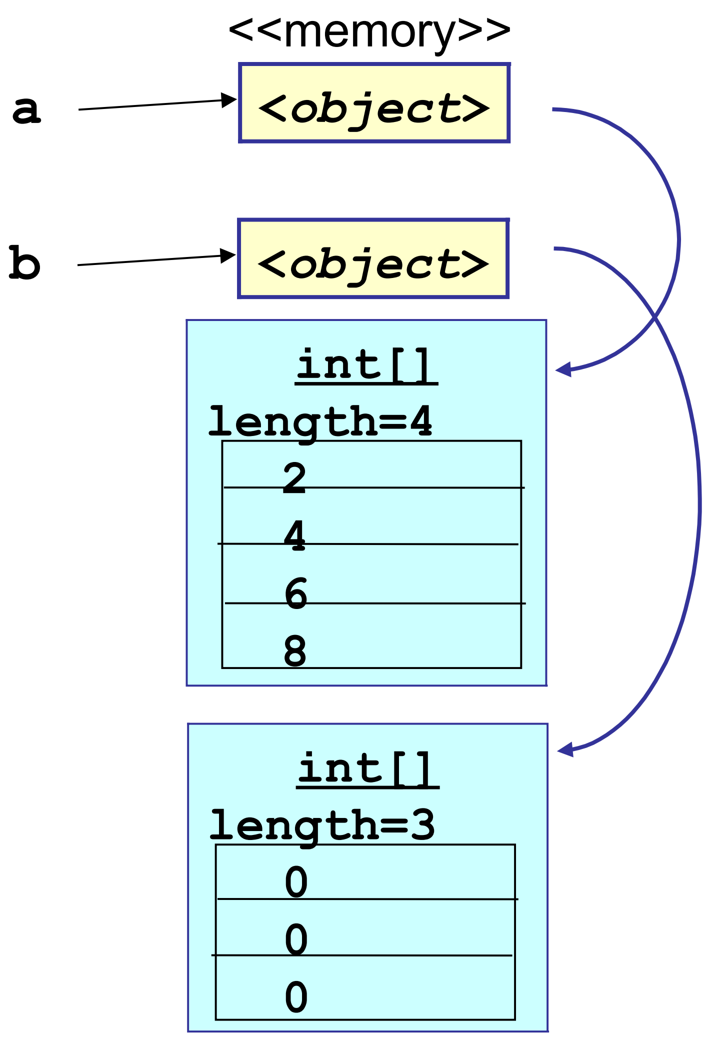
```
int [] a = { 2, 4, 6, 8 };
int [] b = { 0, 0, 0 };
b = a;                    // What does this do?
b[2] = 999;
System.out.println( a[2] );
System.out.println("b.length=" + b.length );
```

An Array Variable is a *Reference*

What does "`b = a`" do?

What will be printed?

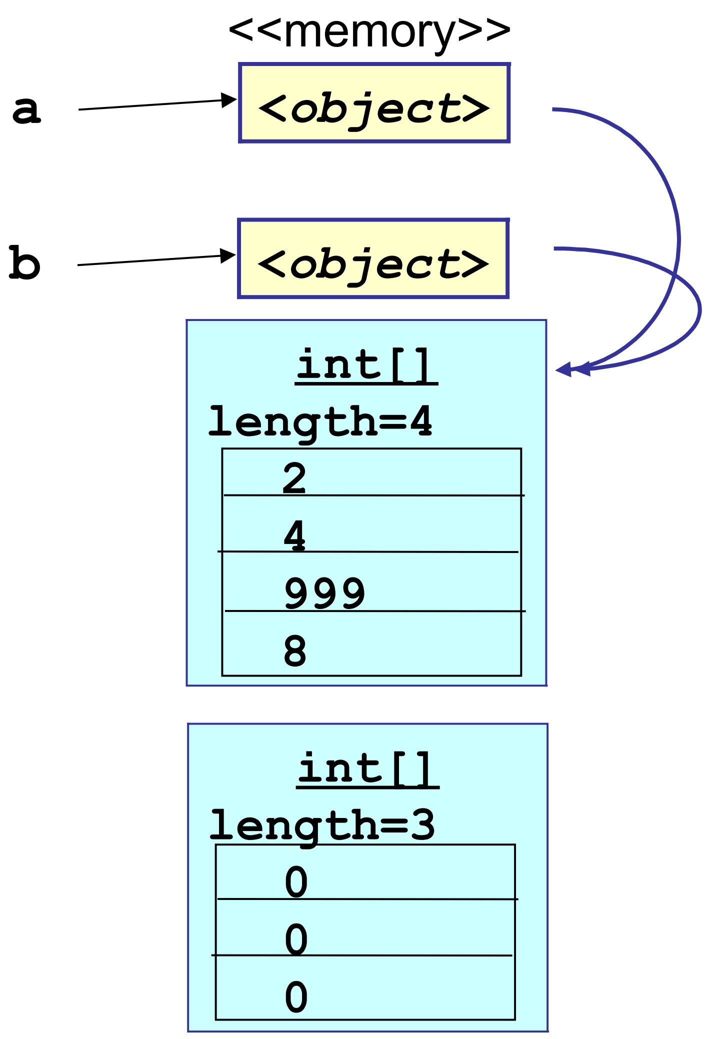
```
int [] a = { 2, 4, 6, 8 };
int [] b = { 0, 0, 0 };
b = a;        // Does what?
b[2] = 999;
System.out.println(a[2]);
System.out.println(
  "b.length=" + b.length );
```



"b = a" copies the *reference, not the array*

`b = a;`
makes **b** refer to same array as **a**.

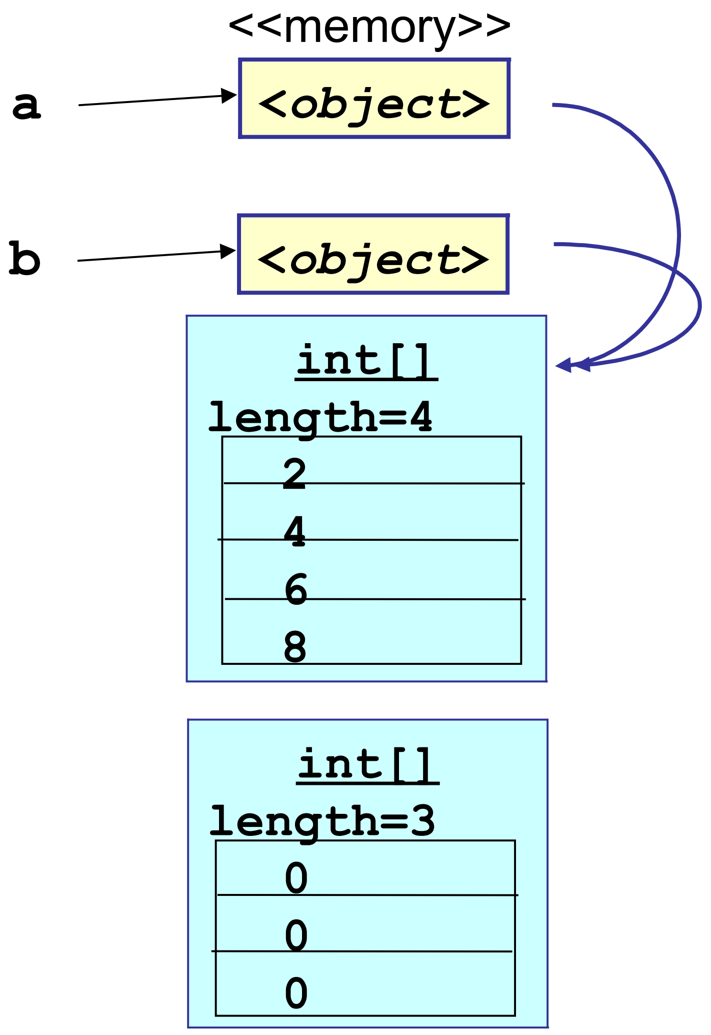
```
b = a;
b[2] = 999;
System.out.println(a[2]);
System.out.println(
  "b.length=" + b.length );
```



The result:

```
b = a;
b[2] = 999;
System.out.println(a[2]);
System.out.println(
 "b.length=" + b.length );
```

```
999
b.length = 4
```



How do you *really* copy an array?

Here is one solution:

```
int[] a = { 2, 4, 6, 8 };


// java.util.Arrays.copyOf( ... )
// creates a new array for copy.
int[] b = Arrays.copyOf( a, a.length );
```

See also: `System.arraycopy( ... )`

Really Copying An Array

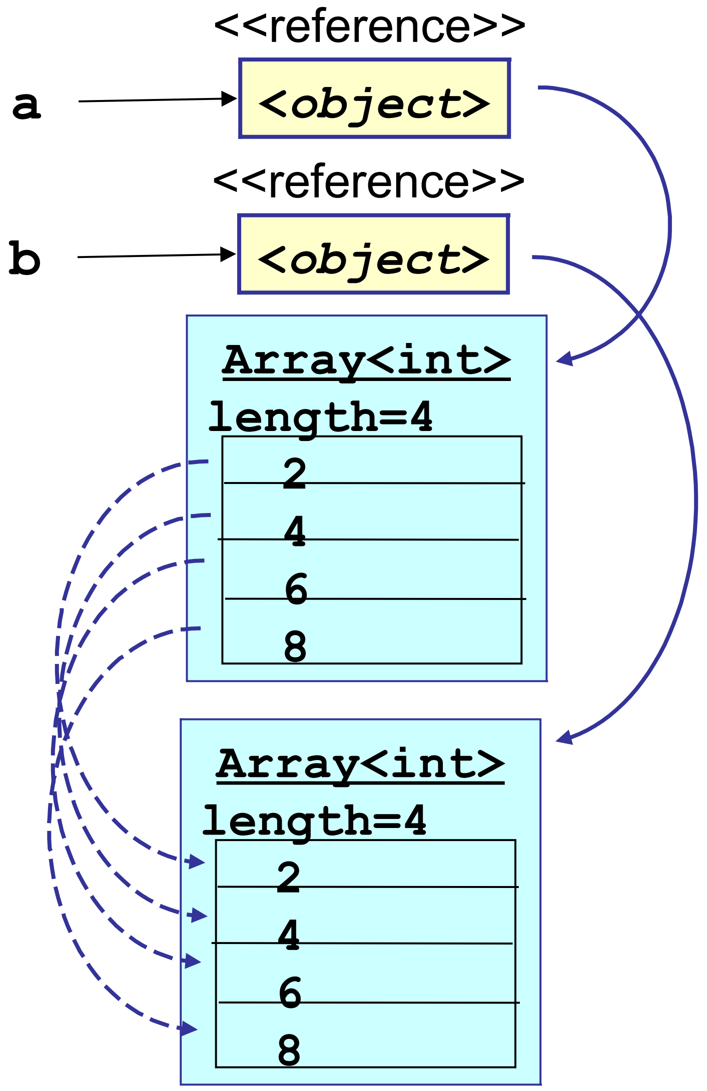
To copy the **contents** of an array, you must copy each element.

```
// copy each element of array
for(int k=0; k<a.length; k++)
   b[k] = a[k];
```

This copies all elements of a.

There is an easier way:

```
System.arraycopy(a, 0, b, 0, 4);
```



System.arraycopy

System.arraycopy(src, src_start, dest, dest_start, length)

copies elements from src array to dest array.
It copies length elements only.

```
// copy each element of array
System.arraycopy(a, 0, b, 0, a.length );
```

"foreach" version of "for" loop

To iterate over every element of an array you can use a special "for" syntax.

```
int [] array = { 2, 4, 6, 8, 10 };

// Iterate over all elements in array.
// "for each x in array do ..."

for(int x:  array) System.out.println( x );
```

Array versus ArrayList (a List)

Array

```
// array of coins
Coin[] coins;
coins = new Coin[10];
coins[0] = new Coin(5);
coins[1] = new Coin(20);
System.out.println( coins[4] );  // print null
```

ArrayList is a kind of List

A List can hold any amount of data.

ArrayList is a kind of list.

List and ArrayList are in java.util.

```
// array of coins
Coin[] coins;
coins = new Coin[10];
coins[0] = new Coin(5);
coins[1] = new Coin(20);
System.out.println( coins[4] );  // print null
```

Array Exercises

- Write a method toString(double [] a) that prints the elements of the array, one per line.
- Write a method to reverse the elements of an array, e.g. reverse(double [ ] a) swaps a[0] and a[n-1], swaps a[1] and a[n-2], ..., for n = a.length;
- write a method name argmax(double [ ] a) that returns the *index* of the maximum element of a.
- write a method name argmax(Comparable [ ] a) that returns the *index* of the maximum element of a. a[] is any array of Comparable objects (has compareTo()).

Useful Array Methods

`a.length` returns the length of array `a`[ ]. This is an attribute, not a method.

Array methods defined in java.util.Arrays include:

`Arrays.fill( a, value );`

set all elements of `a`[ ] to a `value`

`Arrays.sort( a );`

sorts elements of `a`[ ]. sort works for primitive data types, Strings, and array of any type where two objects can be lexically compared using compareTo()

`Arrays.sort( a, start_index, end_index );`

sorts elements of `a`[ ] beginning at `start_index` (inclusive) and ending at `end_index` (exclusive).

Useful Array Methods

`Arrays.binarySearch( a, value )`

return index of element in **`a`**[ ] equal to **`value`**

array **`a`**[ ] must already be sorted.

`Arrays.equals( a, b )`

returns **`true`** of **`a`**[] and **`b`**[] are the same size,

same type, and all elements are equal.

If the elements of **`a`**[ ] and **`b`**[ ] are objects (like Date) then the object's equals method is used to compare them: a[k].equals(b[k])

`Arrays.toString( a )`

return a string representation of **`a`**, such as:

```
[ "apple", "banana", "carrot", "durian" ]
```

Still More Array Methods

`System.arraycopy( src, 0, dest, 0, length)`

copy the elements of `src`[] into the array `dest`[], starting at element 0 and copying `length` elements.

Knowing the **`Arrays`** class will save you time!

You should read the Java API for the java.util.**Arrays** class.

This class contains many useful methods for operations on arrays. You will encounter many situations where you can use these methods in your code.

If you don't know what methods are available, you will waste time writing the same code yourself!

Array Examples

Some examples of using an array.
OK to skip these slides.

Example: find the maximum value

Let's write a method to find the maximum value of an array.

Example: double [ ] a = { 0.5, 2.8 -3.7, 18.0, 9.5 };

max(a) is 18.0

```
/** Find the maximum value of array elements.
 *  @param a is an array of double values.
 *  @return the maximum value in a.
 */

public static double max( double [] a ) {
    // initialize max value to first element
    double maxval = a[0];
    // compare all elements of a to maxval
    for( double x : a )
        if ( x > maxval ) maxval = x;
    return maxval;
}
```

declare parameter **a** as an array.

Example: reverse a String

Write a method named reverse to reverse order of a String.

Example:

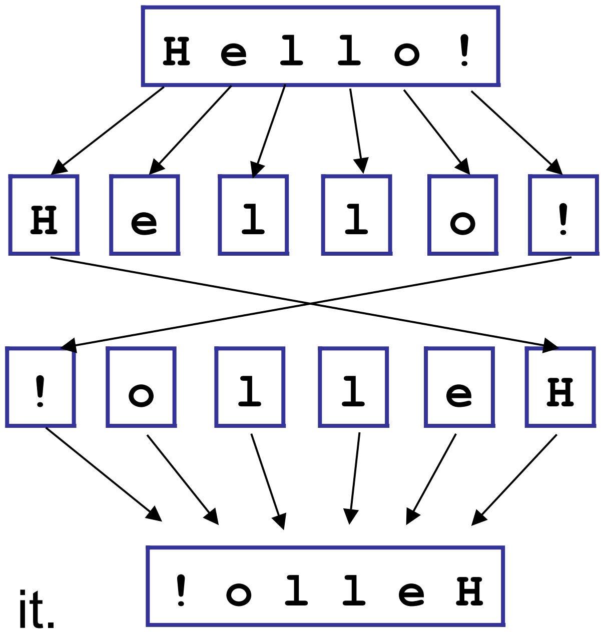
`reverse("Hello there")` returns `"ereht olleH"`

Algorithm:

1. Convert the parameter (String) to an array of characters. Use: `string.toCharArray( )`

2. Iterate over the 1st half of the char array. Swap characters with the 2nd half.

3. Convert char array into a String and return it.



Code: reverse a String

String API methods:

`"string".toCharArray( )` - create char array from String

`new String(char [ ] c)` - create new String from char array

```
public static String reverse( String text ) {
    char[] c = text.toCharArray( );
    // reverse the chars
    for( int k=0; k < c.length/2; k++ ) {
        int k2 = c.length - k - 1;
        char temp = c[k]; c[k] = c[k2];
        c[k2] = temp;
    }
    return new String( c );
}
```

Exercise: case-mangle a String

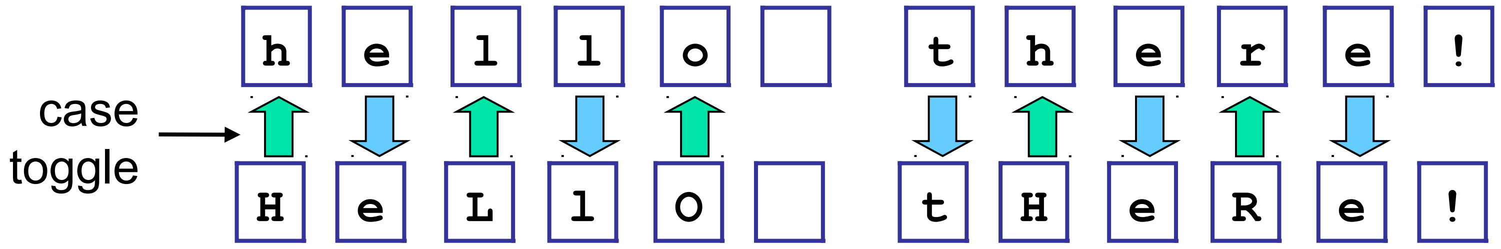
Write a method to mangle the case of a String, **`LiKe ThIs`**

Example:

`mangle("hello there")` returns "**`HeLlO tHeRe!`**"

<u>Algorithm:</u>

1. Convert the parameter (String)to an array of characters. Use: `string.toCharArray( )`
2. Iterate over each character.
   - 2a. if the character is not a letter do nothing.
   - 2b. if the character is a letter change case and record what was the last change (to uppercase or to lowercase).
3. Convert char array into a String and return it.



Resizing an Array

You cannot change the size of an array object.

To change the size, create a new array object and copy elements from old array to the new array.

```
double [ ] a = new double[10];
for(int k=0; k < a.length; k++)
  a[k] = input.nextDouble( );
/* oops! array is too small */

// create a larger array.
a = new double[100];
```

Error: the old values of a[ ] are lost!

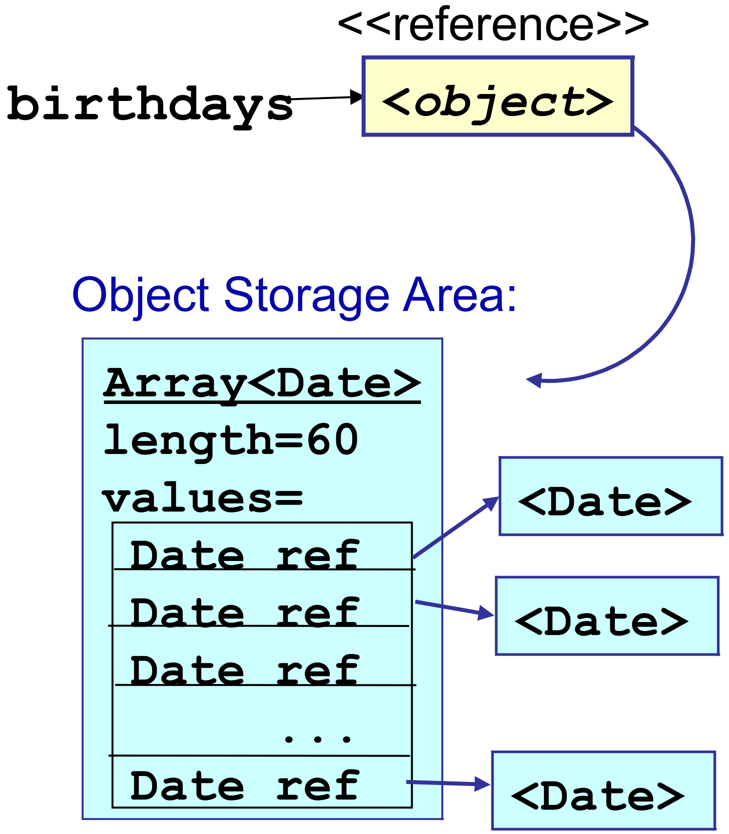
Better Way to Resize Array

- Better: if you don't know how much data you must store, use an **`java.util.ArrayList`** object.
- An **`ArrayList`** grows as needed.
- You can "convert" an ArrayList to an array of exactly the needed size <u>after</u> you have all the data.

```
ArrayList<String> list = new ArrayList<>();
while( scanner.hasNext() ) list.add( scanner.next() );
// Create array of Strings for data in ArrayList
String[] names = new String[ list.length() ];
list.toArray( names );   // copy data into array
```

Creating an Array of Objects

```
// 1. define array reference
Date [ ] birthdays;
// 2. allocate storage
birthdays = new Date[ 60 ];
// 3. create the objects
// that go in the array!
for(int k=0;
  k < birthdays.length; k++ )
  birthdays[k] = new Date( );
```



birthdays[k] is an *object reference.*

You must create the Date object that it will refer to.

Example: Array of Objects (1)

Suppose we have a file on student names and student ID, like this:

```
48540017    Watchara    Srisawasdi
48540165    Kan         Boonprakub
48540181    Keerati     Tangjitsomkid
48540223    Thunthoch   Laksulapan
48540231    Thanyawan   Tarnpradab
48540249    Palawat     Palawutvichai
48540256    Pitchatarn  Lertudomtana
........    more data
```

We want to store the Student ID and name of each student in an array for further data processing.

Example: Array of Objects (2)

Define a simple Student class with attributes for name and Student ID.

```
public class Student {
   String firstName;  // attributes of student
   String lastName;
   String studentID;

   /** constructor for new Student object */
   public Student(String fn, String ln, String id)
   {    studentID = id; // set the attributes
        firstName = fs; // using parameters of
        lastName  = ls;   // the constructor
   }
   ... remainder of class definition omitted...
}
```

Example: Array of Objects (3)

We can create a new Student object like this:

```
Scanner input = new Scanner( System.in );

/* read data for a student */
String id = input.next( );
String first = input.next( );
String last = input.next( );

/* create a new student object */
Student s = new Student( first, last, id );
```