

ICS 3M1 - Computer Science Java

1D Arrays / Simple Tables

Arrays

Singleton Identifiers look like this:

name0	name1	name2
Jo	Sim	Lea

- Each identifier is found in a different memory location
- Each identifier has its own name
- Repetition is difficult due to the different identifier names (can't re-use an identifier and retain the previous input)

Array Identifiers look like this:

name[0]	name[1]	name[2]
Jo	Sim	Lea

- Each identifier is found in a **different, consecutive, memory** location
- Each identifier has a **SHARED name**
- Repetition is easier ... just use the index identifier (from the for repetition) as the subscript
- name[0] - said as “**name at 0**”
- name[2] - value stored here is called the “**element**” of the array
- The [] is referred to as the “**subscript**” of the array

Arrays

How to declare an array?

(Any of the following configurations work for any identifier type)

```
int age[] = new int [5];    // declares an array called age, and at same time
                           // determines that there are 5 elements
                           // (counting from 0-4)
```

```
String name[];             // declares String name
name = new String [10];    // defines name to hold 10 elements
```

```
double price [ ] = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6 };
                        // defines and initializes simultaneously, with the
                        // initialization list determining the number of elements
                        // (6 in this case)
```

```
final double TAX [] = {0.07, 0.08};
                        // using an array as a CONSTANT (unchanging
                        // identifier)
```

Arrays

How to fill an array?

```
String item [];  
double price[], total=0;           // arrays and non-arrays in same line!  
int count;  
  
c.print ( "How many entries do you wish to make? ");  
count = c.readInt();               // you would need to validate this question!  
  
item = new String [count];         // dynamically allocates memory (ie as required)  
price = new double [count];  
  
for (int x=0; x<count; x++)  
{ c.print ( "Enter Item #");  
  c.print ( x+1);                  // user friendly output ... counting from 1  
  c.print ( ": ");  
  item[x] = c.readLine();  
  c.println();  
  c.print ( "Enter the price: ");  
  price[x] = c.readDouble();       // validated, of course (code not in this example)  
  total = total + price[x];       //calculating from arrays are done in loops too!  
}
```

Arrays

What happened in RAM?

	item		price
0	paper	0	4.95
1	pen	1	2.49
2	pencil	2	0.99
3	eraser	3	0.15
4	binder	4	7.80
5	ruler	5	4.50
6	dividers	6	3.10
7	pda	7	500.00
8	Ipod	8	800.00
9	laptop	9	3000.00

total
4323.98

- Arrays were created in consecutive memory
- Length of item was unknown until all entries were made due to COMPLEX identifier of String
- Length of price was predetermined since double always take 8 bytes, therefore $8 \times 10 = 80$ bytes of memory used
- Total was a primitive singleton identifier of 8 bytes

Arrays

How to print from an array?

```
for (int x=0; x<count; x++)           // x can be reused, since x was a 'local' index
{ c.print (x+1,5);                    // ensures right alignment of numbers
  c.print ( "/t" + item[x]);
  c.setCursor(x+3, 30);               // assuming first output on line 3
  c.println ( price[x],8,5);         // field descriptives
}

c.print ( "Total value is: $");
c.println( total,0,2);                // ensures 2 decimals, with no defined width
```