CS100J 13 March 2008 Arrays. Reading: Secs 8.1, 8.2, 8.3
Listen to the following lectures on loops on your Plive CD. They are only 2-3 minutes long, and each has an insightful message.

1. The 3 lectures on Lesson page 7-6 -read the whole page.
2. The 4 lectures in Lesson page 7-5.
Computational simplicity

If you are writing too much code -it gets longer and longer, with no end in sight: stop and look for a better way.
If your code is getting convoluted and you have trouble understanding it: stop and look for a better way.

Learn to keep things simple, to solve problems in simple ways. This sometimes requires a different way of thinking.
We are trying to teach not just Java but how to think about problem solving.
A key point is to break a problem up into several pieces and do each piece in isolation, without thinking about the rest of them. Our methodology for developing a loop does just that.

Array: object. Can hold a fixed number of values of the same type. Array contains 4 int values.

The type of the array:

## int []

Variable contains name of the array. $\quad x a 0$ int[]


Basic form of a declaration:
<type> <variable-name> ;
A declaration of $x$.
int[] x ;
Does not create array, it only declares $x$.
$x$ 's initial value is null.
Elements of array are numbered: $0,1,2, \ldots$, x.length -1 ;
Make everything as simple as possible, but no simpler. Einstein

| $\boldsymbol{i n t}[] \mathrm{x}$; | x null | Arrays |  |
| :---: | :---: | :---: | :---: |
| - |  |  | a0 |
| $\mathrm{x}=$ new int[4]; | Create array object of length 4, | 0 | 0 |
|  | store its name in x | 1 | 0 |
|  |  | 2 | 0 |
|  | x a0 ${ }^{\text {int[] }}$ | 3 | 0 |
| $\begin{aligned} & x[2]=5 ; \\ & x[0]=-4 \end{aligned}$ | Assign 5 to array element 2 and -4 to array element 0 |  | a0 |
|  |  | 0 | -4 |
|  |  | 1 | 0 |
| x [2] is a reference to element number 2 of array $x$ |  | 2 | 5 |
|  |  | 3 | 0 |
| int $\mathrm{k}=3$; |  |  | a0 |
| $x[k]=2 *$ | Assign $2 * x[0]$, i.e. -8 , to $x[3]$ Assign 6 to $x[2]$ | 0 | -4 |
| $\mathrm{x}[\mathrm{k}]=2$ $\mathrm{x}[0]$ |  | 1 | 0 |
| $x[k-1]=6$; |  | 2 | 6 |
|  |  | 3 | $-8{ }^{4}$ |


| Difference between Vector and array |  |
| :---: | :---: |
| Declaration: int[] a; | Vector v; |
| Elements of a: int values | Elements of v: any Objects |
| Creation: $\mathrm{a}=$ new int[ n ; | $\mathrm{v}=$ new Vector(); |
| Array always has n elements | s Number of elements can change |
| Reference: $\quad \mathrm{a}[\mathrm{e}]$ | v.get(e) |
| Change element: $\mathrm{a}[\mathrm{e}]=\mathrm{e} 1$; | v.set(e, e1) ; |
| Array locations $\mathrm{a}[0], \mathrm{a}[1], \ldots$ in successive locations in memory. Access takes same time no matter which one you reference. <br> Elements all the same type (a primitive type or class type) | Can't tell how Vectors are stored in memory. Referencing and changing elements done through method calls <br> Elements of any Object type (but not a primitive type). Casting may be necessary when an element is retrieved. |


| Array initializers |  |  |
| :---: | :---: | :---: |
| Instead of |  |  |
| int[] c= new int[5]; |  | 5 |
| $c[0]=5 ; \mathrm{c}[1]=4 ; \mathrm{c}[2]=7 ; \mathrm{c}[3]=6 ; \mathrm{c}[4]=5$; |  | 4 |
|  |  | 7 |
| Use an array initializer: |  | 6 |
| int[] c= new int[ ] \{5, 4, 7, 6, 5\}; |  | 5 |
| No expression between brackets []. <br> array initializer: gives values to be in the array initially. Values must have the same type, in this case, int. Length of the array is the number of values in the list |  |  |
| Computer science has its field called computational complexity; mine is called computational simplicity. Gries |  |  |


static; object assigned to it will be created only once.
Public: can be seen outside class D.
final: it cannot be changed.

## Procedure swap

## public class D \{

$/ * *=$ Swap x and $\mathrm{y} * /$
public static void swap (int $x$, int $y$ ) \{ int temp $=x$;

| $\mathrm{x}=\mathrm{y} ;$ | A call will NOT swap a and b. |
| :--- | :--- |
| $\mathrm{y}=$ temp; | Parameters x and y are initialized to |
| the values of a and b, and thereafter |  |

\}
he values of $a$ and $b$, and thereafter,
....
swap(a, b);
frame for call just after
frame created and args assigned to pars: there is no way to change $a$ and $b$.

Procedure swap
public class D \{
/** $=$ Swap b[h] and b[k] */
public static void swap (int[] b, int $h$, int $k$ ) \{ int temp $=b[h]$; $\mathrm{b}[\mathrm{h}]=\mathrm{b}[\mathrm{k}] ;$ $\mathrm{b}[\mathrm{k}]=$ temp; \}
$\}^{\}}$
$\cdots$
$\operatorname{swap}(\mathrm{c}, 3,4) ;$$\quad$ с a 0


Does swap $b[h]$ and $b[k]$, because parameter $b$ contains name of the array.

| a 0 |
| ---: |
| 5 |
| 4 |
| 7 |
| 6 |
| 5 |

Two-dimensional arrays

| 0 | 1 | 2 | 3 |  |
| :--- | :--- | :--- | :--- | :--- |
| $b$ | 5 | 4 | 7 | 3 |$\quad$ b.length one-dimensional array

d | $\left.\begin{array}{llll}0 & 1 & 2 & 3 \\ 0 & \begin{array}{llll}5 & 4 & 7 & 3 \\ 4 & 8 & 9 & 7 \\ 5 & 1 & 2 & 3 \\ \hline\end{array}\end{array}\right)$ |
| :---: |

2
rectangular array: 5 rows and 4 columns

| Type of d is int[][] ("int array array", |  |
| :---: | :---: |
| "an array of int arrays") |  |
| To declare variable d: | number of rows |
| int d[][]. |  |
| To create a new aryay and assign it to d: |  |
| d= new int[3][4]; |  |
| $\begin{array}{l}\text { To reference element at row r column c: } \\ d[r][\mathrm{c}]\end{array}$ | number of cols |
|  |  |



## Linear search

public class D \{
/** $=$ index of first occurrence of c in $\mathrm{b}[\mathrm{h} . . \mathrm{k}-1]-$
$=\mathrm{k}$ if c is not in $\mathrm{b}[\mathrm{h} . \mathrm{k}-1]^{* /}$
public static int findFirst (int c, int [] b, int h, int k$)\{$
// if c is in $\mathrm{b}[\mathrm{h} . . \mathrm{k}-1]$, return its index in $\mathrm{b}[\mathrm{h} . . \mathrm{k}-1]$
// \{ invariant: c is not in $\mathrm{b}[\mathrm{h} . \mathrm{t}-1]$ \}
for $($ int $t=h ; t<k ; t=t+1)\{$

| $/ /$ Process $\mathrm{t} ;$ | Remember |
| :--- | :--- |
| if $(\mathrm{b}[\mathrm{t}]==\mathrm{c})$ | h. $\mathrm{h}-1$ is the |
| return $\mathrm{t} ;$ | empty range |

\} empty range
$/ /\{\mathrm{c}$ is not in $\mathrm{b}[\mathrm{h} . \mathrm{k}-1]\}$
return k ;
\}
\}

