Review Repetition Structure

Write a program to receive an non-negative integer and display 0,1,2 to the input.

1) Use while loop
2) Use for loop

Write a program to receive an non-negative integer and display 1,2,3 to the input.

1) Use while loop
2) Use for loop

Write a program to receive an non-negative integer and display 2,4,6 to the input.

1) Use while loop
2) Use for loop

Write a program to receive an non-negative integer and display 1,3,5 to the input.

1) Use while loop
2) Use for loop

Write a program to receive an non-negative integer and display the integer divisor of the input.

1) Use while loop
2) Use for loop

Array

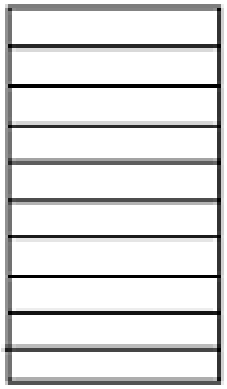

CAMT
College of Arts, Media and Technology
Chiang Mai University

Arrays
One name for many data

- An *array* is a named sequence of contiguous memory locations capable of holding a *collection* of data of the same type.
- Unlike the variables of previous programs, an array can store *more than one* value.
- An array can store a list of thousands or even millions of data

A variable in contrast to an array

a variable: storage for one value



an array: contiguous memory storage for many values.

Array Declaration

- `type[] name:`
  - type is a *data type* such a int, char, double or boolean
  - name is a valid Java identifier that provides a name to an array

- `int[] myArray` and `double[] yourArray`:
  - The variables myArray and yourArray are *reference* variables.
  - A *reference* variable does not hold an integer, a floating point number, a character or a boolean value.
  - A reference variable holds *a memory address*.

Array Declaration

- Java provides two types of variable primitive and reference:
  - A primitive variable stores a *single* value of type byte, short, int, long, float, double, boolean, or char.
  - A reference variable holds a memory address or reference.

- Each of the two variables `myArray` and `yourArray`, when assigned a value, holds
  - a memory address
    - the address of the *first* cell of a block of storage locations

Array Instantiation

- The references `myArray` and `yourArray` are *uninitialized*.
- A declaration does not *create* an array.
- Once an array reference is declared
  - memory for the array must be allocated.

Array Instantiation

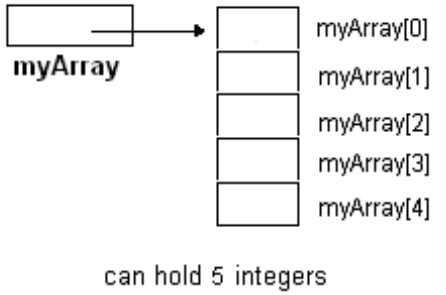
- An array is created or instantiated via the new operator:

```
type[] name;              // declaration
name = new type[size];          // array instantiation
       or
type[] name = new type[size]  // declaration and
             // instantiation
```

- type is a data type
- size is a positive integer or any expression that evaluates, or is automatically converted, to a positive integer.
- The integer size indicates the *length* of the array, i.e., the number of cells in the array.

Array Instantiation

- The values held in an array must all be the *same data type*.
- When an array is created, each cell is automatically given a *unique name* .
- The names of the cells of the array referenced by `myArray` are `myArray[0], myArray[1], myArray[2], myArray[3], and myArray[4]`
- In this case, the array is indexed from 0 to 4
- The first index of every array is 0.



Array Instantiation

- When an array is instantiated, each memory cell is initialized with the "*zero value*" of its data type.

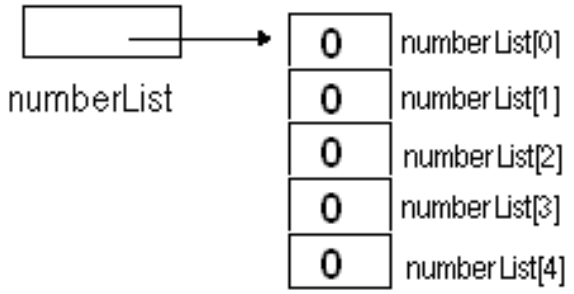
- Thus every cell of an array of `int` or `char` data is initialized to *0* and all cells of an array of `doubles` are set to *0.0*.

- Each cell of a `boolean` array is initialized to `false`.

Array Instantiation

```
int [] numberList;
numberList = new double[5];
```

- **********
- numberList refers to an array of 5 integers
- Each array cell is initialized to 0

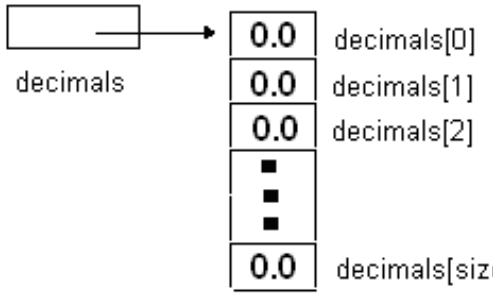


CAMT
College of Arts, Media and Technology
Chiang Mai University

Array Instantiation

```
System.out.println("Enter array size:  ");
int size = input.nextInt();
double [] decimals;
decimals = new double[size];
**********
```

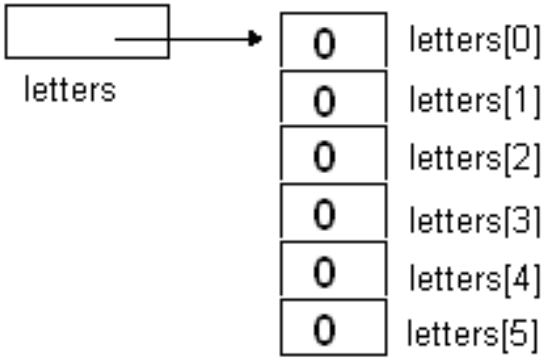
- The length of the array (size) is supplied at runtime
  The name decimals refers to an array of d
- Each memory cell is initialized to 0.0.



Array Instantiation

```
int x = 3;
char [] letters = new char[2*x];
```

- **********
- letters refers to an array of char data.
- Each memory cell is i



Array Instantiation

- Once an array is created, its length is *fixed*.
- The length of an array cannot be *altered*.
- If variable `x` refers to an array, then `x.length` gives the number of memory cells allocated to the array.

Using an Array

- You can use array variables in assignment statements or any expression.
- The statement:

```
int[] numbers = new int[5]
```

- declares and instantiates an array named numbers such that:
  - numbers is indexed from 0 to 4.
  - numbers is capable of storing 5 integers in locations numbers[0], numbers[1], numbers[2], numbers[3], and numbers[4].
  - numbers.length has the value 5.
  - the initial value stored in each cell of numbers is 0.

Using an Array

- Each memory cell is a variable:

```
numbers[0] = input.nextInt();                // reads a value into numbers[0]

numbers[1] = 213 + numbers[1];               // adds 213 to numbers[1]

System.out.println(5*numbers[1] + 3*numbers[0]);
```

Using an Array

Program statement:

- Write a program that prompts for 10 integers and displays those same numbers in reverse order. For example, if you enter the numbers:

0 11 2 33 4 55 6 77 8 99

the program's output is:

99 8 77 6 55 4 33 2 11 0

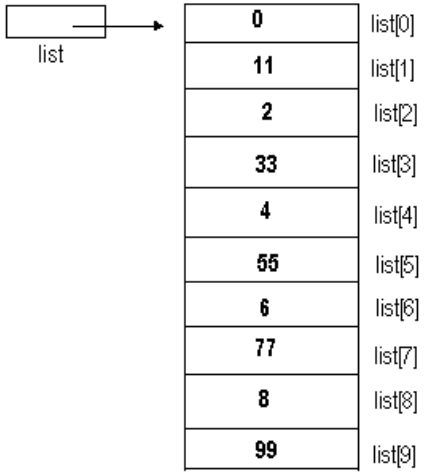
Solution

- The following application stores 10 integers in an array named list.
- The array consists of 10 memory cells that are named list[0], list[1]...list[9]. The user supplies 10 numbers.
- The numbers are stored in these 10 cells and finally, the numbers are displayed in reverse order.

```
1. import java.util.*;
2. public class ReverseList
3. {
4.     public static void main(String[] args)
5.     {
6.         Scanner input = new Scanner(System.in);
7.         int [] list;    // declare list an array variable
8.         list = new int[10]; //instantiate or create an array named list
9.         System.out.print("Enter 10 integers: ");
10.         // read values into list[0], list[1],..,list[9]
11.         for (int i= 0; i < 10; i++)
12.             list[i] = input.nextInt();
13.         System.out.print("List in reverse : ");
14.         //print values stored in list[9], list[8], … , list[0]
15.         for (int i= 9; i >= 0; i--)
16.             System.out.print(list[i] + " ");
17.         System.out.println();
18.     }
19.}
```

Discussion

- The program prompts for 10 integers. After the data is entered (lines 11-12):



- The array list holds 10 integers.
- The individual memory cells are designated list[0], list[1], list[2], ... , list[8], and list[9]

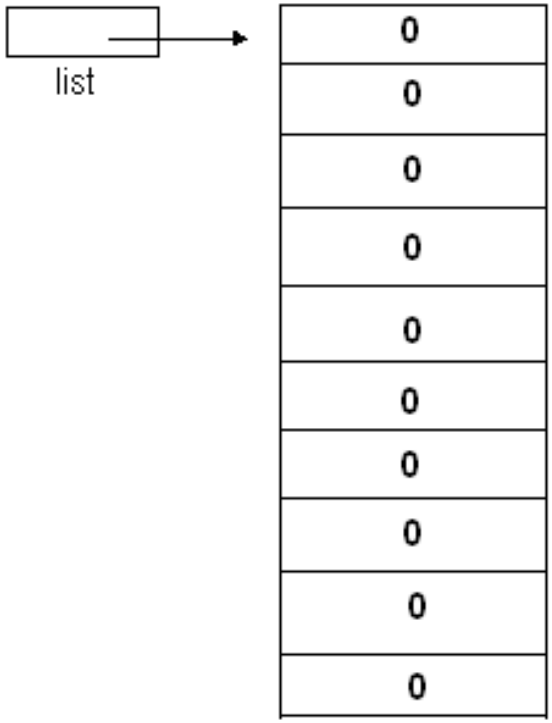
Discussion

- Line 7: `int [] list;`
  Line 7 is an array declaration as indicated by the square brackets [].
- Like any variable, list must be *declared* before its use.
- The statement on line 7 declares that list is a reference variable
- The declaration does not assign a value to list.
- No memory has been *allocated* yet; no array *exists* yet.
-

Discussion

- `Line 8: list = new int[10];`
- The segment `new int[10]` *allocates* a block of memory large enough to store ten integers
  - returnsthe starting address of this memory chunk.
- The new operator *creates or instantiates* a new array.
- The operator reserves a *consecutive* block of storage locations in memory
  - return the starting address of the block. All are initialized to 0:

list

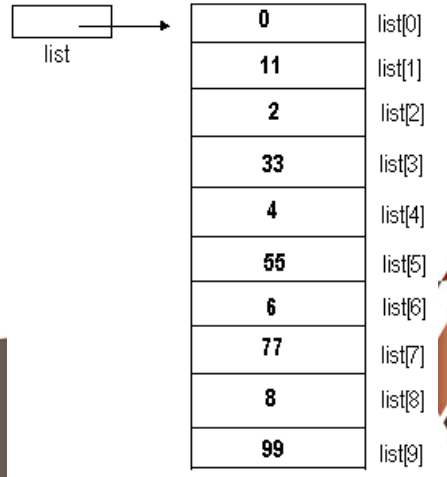


Discussion

- Lines 11 and 12:

```
for (int i= 0; i < 10; i++)
    list[i] = input.nextInt();
```

Lines 11 and 12 comprise a for loop that accepts interactive input and stores the values in list[0], list[1], list[2], list[3], list[4], list[5], list[6], list[7], list[8], and list[9].



Discussion

- Lines 15 and 16:

```
for (int i= 9; i >= 0; i--)
System.out.print(list[i] + " ");
```

  - The for loop prints the array items in reverse.: list[9], list[8], …, and list[0].

Activity

- Group of two or three people
- Write source code to create this array

1.1
2.2
3.2
4.5
5.4

Array Initialization

- Java provides a convenient form of array initialization.
- The following code segment declares and explicitly initializes an array of characters:

```
char letters[] = {'a', 'b', 'c'};
```

- The new operator is not *explicitly* used.
- This initialization of letters is equivalent to:

```
char letters[] ;
letters = new char[3];
letters[0] = 'a';
letters[1] = 'b';
letters[2] = 'c';
```

-
- .

Array Initialization

- Similarly:

```
int squares = {0,1,4,9,16,25,36,49,64,81,100};
```

is shorthand for:

```
int squares = new int[11];
for (int i= 0; i<11; i++ )
squares[i] = i*i;
```

- Explicit method of array initialization is *convenient*
  - only when the size of the array is not particularly large

The = Operator

- The assignment operator (=) can be used with array references, but such use can lead to some unexpected results and subtle bugs:

```
int [] a = {5,4,3,2,1};
int [] b = new int[5];  // initialized to 0's
b = a;
a[0] = 100;
System.out.println(" a[0] is " + a[0]+
                   " and b[0] is " +b[0]);
```

-

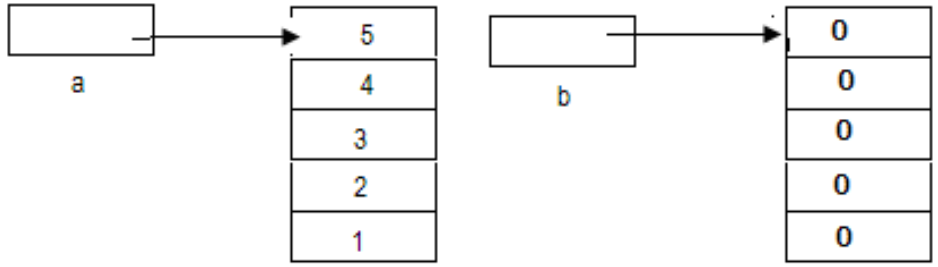
Output:

a[0] is 100 and b[0] is 100

- Both a[0] and b[0] are 100.
- a and b are both *references*, i.e., a and b each holds a *single address*.

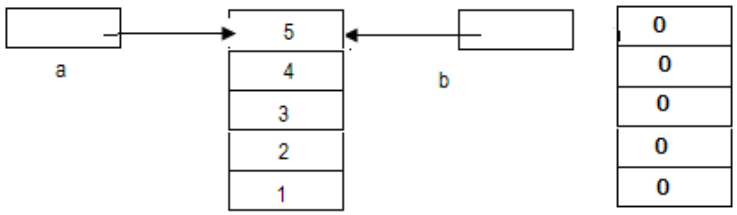
The = Operator

```
int a = {5,4,3,2,1}
int b = new int[5]
```



The = Operator

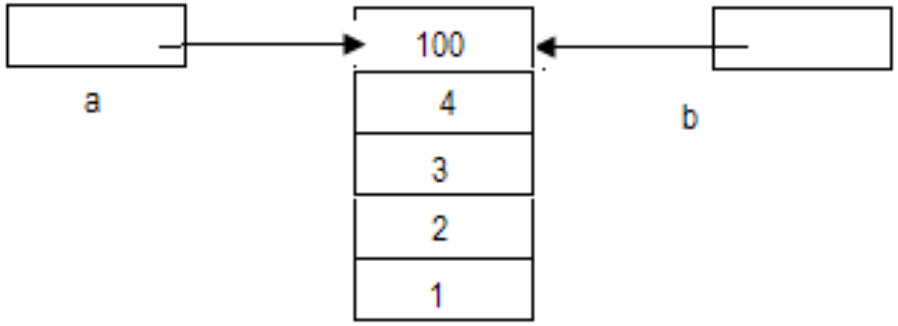
```
b = a
```



After the assignment b=a

The = Operator

```
B[0] = 100;
```



After executing the assignment b=a, the references a and b both refer to the same memory and any changes to a[i] affect b[i].

The == Operator

- The == operator does not *compare* the contents of the arrays; the == operator compares references.

```
int[]  a = {5,4,3,2,1};
int[]  b = {5,4,3,2,1};
int[]  c = a;
```

- The expression `a == b` is false because a and b hold different references
- `a == c` evaluates to true.

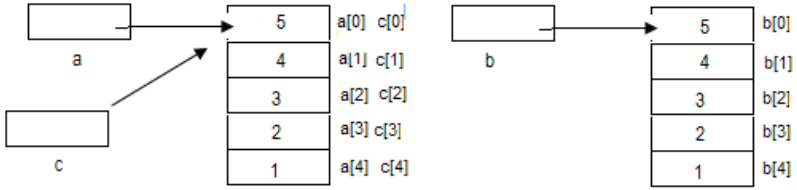


Figure 7.13 Comparing array references

Example

- Create a program to read 5 integers from user and display the highest value, the lowest value and the average of the list.

Example

- Create a program to read 10 doubles from user. Then, the inputs two doubles. The second input will replace the first value in the list.

Example

- Create a program to receive 2 sequences of 5 integers. Display both of them and copy the first set to the second sequence and display them.

Example

- Create a program to receive a sequence of 10 doubles. Then, receive an double and display if the input is in the list or not.

Example

- Create a program to receive a sequence of 10 doubles. Then, receive an double and display the number of value that is larger than or equal to the input.

Example

- Create a program to receive the list of 5 student's weight and display who is the highest.

Example

- Create a program to random 10 non-negative integer and sort the array from the lowest to the largest.