Fundamentals of Java

Arrays and Lists

What is an array?

An array is an object that stores a list of values of the same type.

`{ 3, 7, 2, 9 }`	An array of int values
`{ "cat", "dog", "chair" }`	An array of String objects
`{ new Person("John", new Person("Sally") }`	An array of Person objects

What is the purpose of an array?

An array can be used to store a collection of values into a single variable:

```
int[] numbers   =  { 3, 7, 2, 9};
String[] words  =  { "cat", "dog", "chair" };
Person[] friends = { new Person("John"),
                     new Person("Sally") };
```

The type of the variable must contain `[]`.

e.g. An array of Strings has type `String[]`.

Array length

Given array `a`, its length is `a.length`

e.g.

```
String[] words = { "cat", "dog", "chair" };
System.out.println("There are " +
                   words.length + " words.");
```

Accessing elements

Use the `[n]` suffix to access individual elements:

```
String[] words = { "cat", "dog", "chair" };

System.out.println("The first  word is " + words[0]);
System.out.println("The second word is " + words[1]);
System.out.println("The third  word is " + words[2]);
```

Note: The first position is 0.

Storing into elements

Each element is actually a variable, and can be subject to assignment.

```
String[] words = { "cat", "dog", "chair" };

words[2] = "house";   <-- A variable assignment
```

The array content has changed to:

```
{ "cat", "dog", "house" };
```

Storing elements into a 2D array

2D arrays are accessed with [row][col].

```
int[][] grid = {    {1, 2, 3, 4},
                    {5, 4, 1, 7},
                    {3, 8, 9, 6}
                };
grid[2][1] = 1;     <-- A variable assignment
```

The array content has changed to:

```
{   {1, 2, 3, 4},
    {5, 4, 1, 7},
    {3, 1, 9, 6}   }
```

Passing arrays to methods

Arrays can be passed as arguments:

```
void main()
{
    String[] myWords = { "cat", "dog", "chair" };
    printWords(myWords);
}

public void printWords(String[] list)
{
    for (int i = 0; i < list.length; i++)
        System.out.println(list[i]);
}
```

Problem solving

Virtually all array problems can be solved using a for loop:

```
for (int i = 0; i < words.length; i++)
    System.out.println(words[i]);
```

The counter i starts at zero.

Example problems

```
int[] nums = { 2, 4, 7, 8, 3 };

for (int i = 0; i < nums.length; i++)
    System.out.println(nums[i]);

for (int i = nums.length-1; i >= 0; i--)
    System.out.println(nums[i]);

for (int i = 0; i < nums.length; i+=2)
    System.out.println(nums[i]);

for (int i = 0; i < nums.length; i++)
    if (nums[i] < 5)
        System.out.println(nums[i]);
```

Problem: Find the biggest number

```
int[] nums = { 2, 4, 7, 8, 3 };

int biggestSoFar = -100;

for (int i = 0; i < nums.length; i++)
    if (nums[i] > biggestSoFar)
        biggestSoFar = nums[i];

System.out.println("The biggest number is "
                   + biggestSoFar);
```

Problem: Find the biggest number

Another approach:

```
int[] nums = { 2, 4, 7, 8, 3 };

int biggestSoFar = nums[0];

for (int i = 1; i < nums.length; i++)
    if (nums[i] > biggestSoFar)
        biggestSoFar = nums[i];

System.out.println("The biggest number is "
                   + biggestSoFar);
```

Problem: Find the longest string

```
String[] words = "dog", "house", "tree";

String longestSoFar = words[0];

for (int i = 1; i < words.length; i++)
    if (words[i].length() > longestSoFar.length())
        longestSoFar = words[i];

System.out.println("The longest string is "
                        + longestSoFar);
```

Problem: Find the total

```
int[] nums = { 2, 4, 7, 8, 3 }

int sum = 0;

for (int i = 0; i < nums.length; i++)
    sum += nums[i];

System.out.println("The total is " + sum);
```

Problem: Find the average

```
int[] nums = { 2, 4, 7, 8, 3 }
int sum = 0;

for (int i = 0; i < nums.length; i++)
    sum += nums[i];

int average = sum / nums.length;

System.out.println("The average is "
                                + average);
```

Creating blank arrays

A blank array is created with `new type[size]`.

```
String[] dictionary = new String[150000];
```

This creates an array of size 150000.

Unlike C, the size of an array can be chosen at runtime. i.e. it can be any expression.

Example

```
System.out.print("How many words do you want?
");
int numberOfWords = keyboard.nextInt();

String[] words = new String[numberOfWords];

for (int i = 0; i < words.length; i++)
{
    System.out.print("Enter a word: ");
    words[i] = keyboard.nextLine();
}
```

Array cookbook - example #1

```
int findBiggest(int[] numbers)
{
    int biggestSoFar = numbers[0];
    for (int i = 1; i < numbers.length; i++)
    {
        if (numbers[i] > biggestSoFar)
            biggestSoFar = numbers[i];
    }
    return biggestSoFar;
}
```

Array cookbook - example #2, #3

```
int total(int[] numbers)
{
    int total = 0
    for (int i = 1; i < numbers.length; i++)
    {
        total += numbers[i];
    }
    return total;
}
double average(int[] numbers)
{
    return total(numbers) / (double)numbers.length;
}
```

Array cookbook - example #4

```
int linearSearch(int[] numbers, int e)
{
    for (int i = 0; i < numbers.length; i++)
        if (numbers[i] == e)
            return i;
    return -1;
}
```

Array cookbook - example #5

```
boolean equal(int[] a, int[] b)
{
    if (a.length != b.length)
        return false;
    for (int i = 0; i < a.length; i++)
        if (a[i] != b[i])
            return false;
    return true;
}
```

Array cookbook - example #6

```
boolean allPositive(int[] numbers)
{
    for (int i = 0; i < numbers.length; i++)
        if (numbers[i] < 0)
            return false;
    return true;
}
```

Lists

The java.util package provides a class called ArrayList which wraps all the behaviour of arrays in an ordinary object:

```
ArrayList words = new ArrayList(6);
words.set(0, "the");
words.set(1, "cat");
words.set(2, "sat");
words.set(3, "on");
words.set(4, "the");
words.set(5, "mat");
println(words.size());
println(words.get(2));
```

```
String[] words = new String[6];
words[0] = "the";
words[1] = "cat";
words[2] = "sat";
words[3] = "on";
words[4] = "the";
words[5] = "mat";
println(words.length);
println(words[2]);
```

Lists

Array-style programming is possible with lists

Lists:

```
for (int i = 0; i < words.size(); i++)
    System.out.println(words.get(i));
```

Arrays:

```
for (int i = 0; i < words.length; i++)
    System.out.println(words[i]);
```

List element types

When you define an array, you also define the type of the elements it contains:

```
String[] words;
Person[] friends;
```

To specify the element type for lists, use <...>:

```
LinkedList<String> words;
LinkedList<Person> friends;
```

List element types

If you don't specify the element type, Java won't know the type of element, or what methods are supported on the element:

```
LinkedList cars = .....;
cars.get(i).drive();        // compile error

LinkedList<Car> cars = .....;
cars.get(i).drive();        // OK
```

Always specify the element type!

List element types

The element type must be specified after EVERY reference to the ArrayList class name:

```
LinkedList<String> words = new LinkedList<String>(10);
words.set(3, "tree");
String fourth = words.get(3);
```

Think of `LinkedList` as only a partial class name, and `LinkedList<String>` as a complete class name.

Why use lists?

The size of an array is fixed at the time of creation. You must carefully decide on the size.

A List can grow and shrink over time.

- Start by creating an empty list.
- add() elements as needed.
- remove() elements as needed.

Why use lists?

```
ArrayList<String> words = new ArrayList<String>();

words.add("the");
words.add("cat");
words.add("sat");
words.add("on");
words.add("the");
words.add("mat");

System.out.println(words.get(2)); // prints "sat"
words.remove(2);
System.out.println(words.get(2)); // prints "on"
```

The for-each loop

Java provides a special syntax for looping over lists and arrays:

```
ArrayList<String> words = ....;

for (String word : words)
    System.out.println(word);
```

This is equivalent to:

```
for (int i = 0; i < words.size(); i++) {
    String word = words.get(i);
    System.out.println(word);
}
```

Example

```
ArrayList<String> words = new ArrayList<String>();

words.add("the");
words.add("cat");
words.add("sat");
words.add("on");
words.add("the");
words.add("mat");

System.out.println(words.get(2)); // prints "sat"
words.remove(2);
System.out.println(words.get(2)); // prints "on"
```