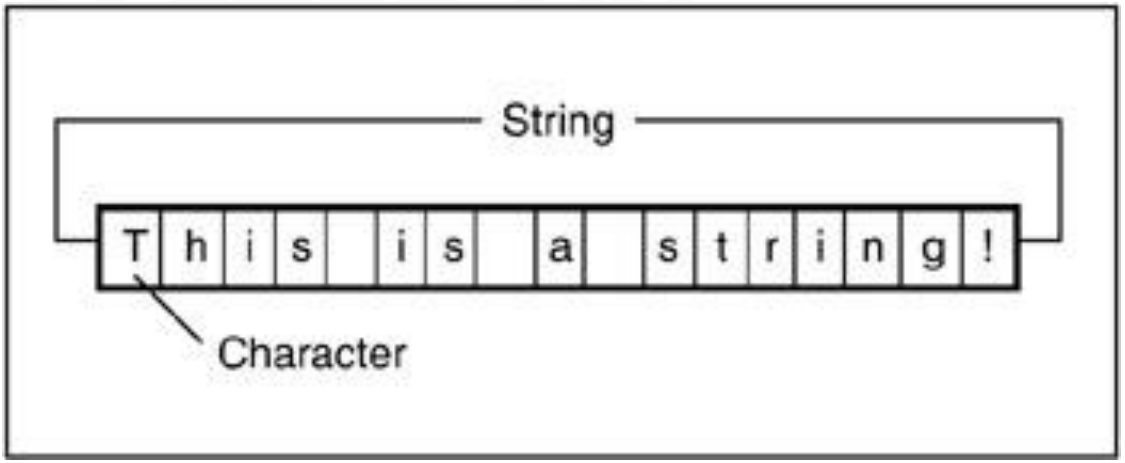
DEVMASTER.VN

Bài 6
Mảng và Chuỗi

Mục tiêu

- Giới thiệu mảng (Array)
- Mảng đơn chiều, đa chiều
- ArrayList
- Giới thiệu chuỗi (String)
- Lớp StringBuilder



Giới thiệu

- Hãy xem xét tình huống người dùng muốn lưu trữ điểm của **10 sinh viên**.
- Liệu có thể tạo **10 biến** kiểu số thực?
- Nhưng….nếu có **1.000 sinh viên**?
- Liệu có thể lưu trữ **tất cả** các nhãn vào **một vị trí** và truy cập thông qua **chỉ mục** (index)?
- Java cung cấp cách thức cho phép lưu trữ nhiều giá trị cùng kiểu dữ liệu trong 1 biến gọi là **mảng**.

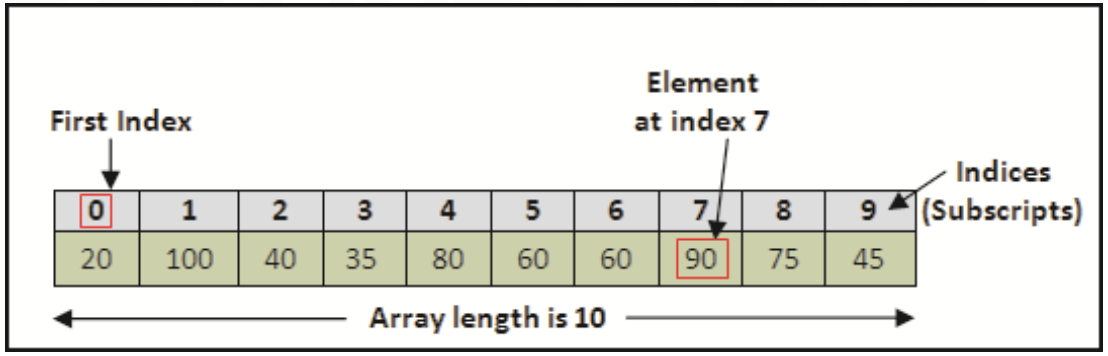
Giới thiệu

Mảng là vùng lưu trữ dữ liệu đặc biệt có thể cố định một số lượng giá trị liền kề nhau trong cùng một vị trí bộ nhớ duy nhất.

Nó triển khai như một đối tượng.

Kích thước của mảng phụ thuộc vào số lượng giá trị của nó có thể lưu trữ và chỉ định khi nó được tạo

Sau khi mảng được tạo, kích thước của nó là cố định. Hình sau thể hiện mảng của một tập hợp số:



Giới thiệu

Mảng có lợi ích như sau:

- Mảng là cách tốt nhất để thao tác nhiều dữ liệu cùng loại đồng thời.
- Mảng sử dụng được tối ưu nguồn tài nguyên bộ nhớ so với biến.
- Bộ nhớ chỉ cấp phát cho mảng khi nó thực sự sử dụng , như vậy bộ nhớ không bị tổn hao khi mảng được khai báo.

Mảng đơn-đa chiều

Mảng có 2 kiểu trong Java:

Mảng đơn-đa chiều

Hình bên thể hiện mảng điểm của sinh viên với các giá

Một mảng đơn chiều chỉ có 1 chiều và đại diện trực quan chỉ có duy nhất một cột với số dòng dữ liệu xác định.

Mỗi phần từ của mảng được truy cập bằng cách kết hợp tên của mảng với chỉ mục (index) tại vị trí của phần từ. Index được đánh số từ 0.

Việc truy cập vào phần tử có chỉ mục (index) nằm ngoài giới hạn của mảng sẽ phát sinh lỗi ngoại lệ. VD: marks[4]…

marks[4]	
Element	Value
marks[0]	65
marks[1]	47
marks[2]	75
marks[3]	50

Mảng đơn-đa chiều

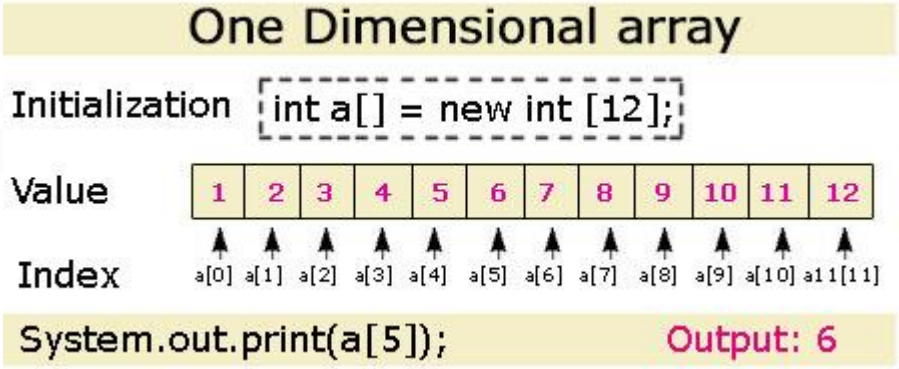
DEVMASTER.VN

Cú pháp khai báo mảng

Syntax

```
datatype[] <array-name>;
```

- byte[] byteArray;
- float[] floatsArray;
- boolean[] booleanArray;
- char[] charArray;
- String[] stringArray;



Syntax

```
datatype[] <array-name> = new datatype[size];
```

Mảng đơn-đa chiều

Mảng đa chiều là mảng mà các phần tử lại là một mảng.

Cú pháp:

Syntax

```
datatype[][] <array-name> = new datatype [rowsize][colsize];
```

Ví dụ:

```
int[][] marks = {{23,65}, {42,47}, {60,75}, {75,50}};
```

Rows	Columns	
	0	1
0	23	65
1	42	47
2	60	75
3	75	50

Mảng đơn-đa chiều

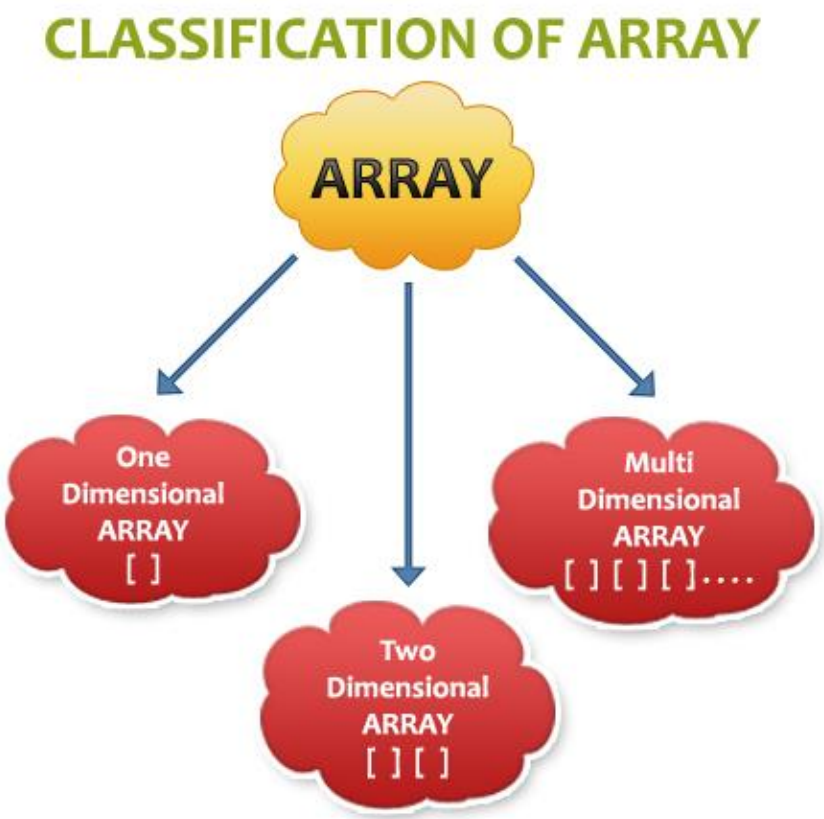
Để truy cập mảng, sử dụng vòng lặp.

```
...
public void displayMarks() {
    System.out.println(“Marks are:”);

    // Display the marks using for loop
    for(int count = 0; count < marks.length; count++) {
        System.out.println(marks[count]);
    }
}
...
```

Mảng đơn-đa chiều

Như vậy có thể tạo mảng với số chiều không giới hạn.



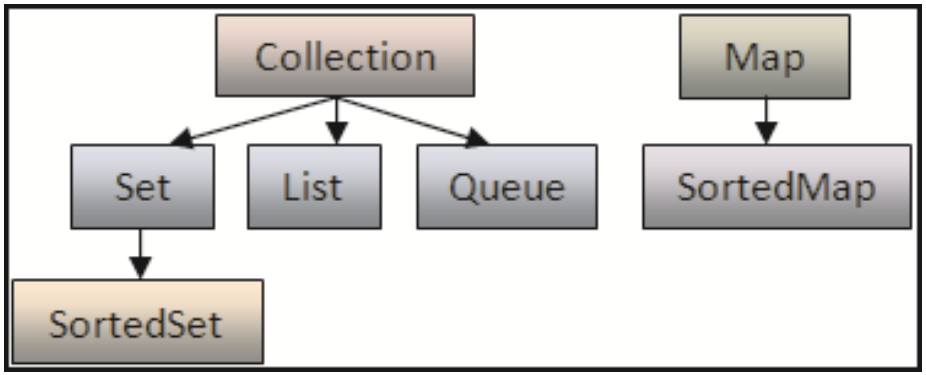
ArrayList

Một nhược điểm lớn của mảng đó chính là kính thước cố định sau khi được tạo. Kích thước này không thể thay đổi sau đó trong quá trình sử dụng.

Để giải quyết vấn đề này cần có một cấu trúc mà bộ nhớ có thể co giãn khi có yêu cầu.

Như vậy, việc thêm-bớt phần tử sẽ trở lên dễ dàng hơn rất nhiều. Java cung cấp khái niệm Collection để giải quyết vấn đề này.

Collection là một đối tượng độc lập nhóm đa phần tử vào trong một đơn vị duy nhất.



ArrayList

Bảng sau thể hiện các class trong Collection

Interfaces	Hash table	Resizable array	Tree	Linked list	Hash table + Linked list
Set	HashSet	-	TreeSet	-	LinkedHashSet
List	-	ArrayList	-	LinkedList	-
Queue	-	-	-	-	-
Map	HashMap	-	TreeMap	-	LinkedHashMap

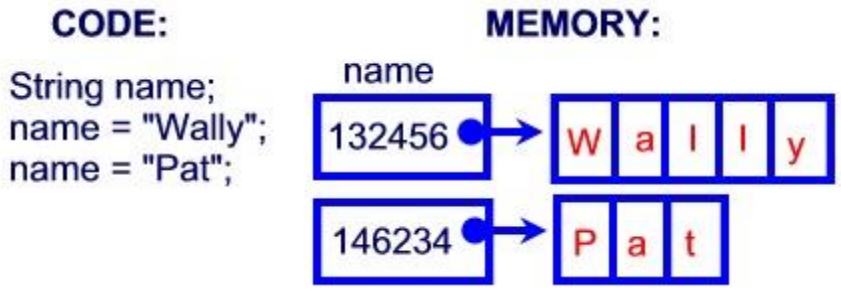
ArrayList

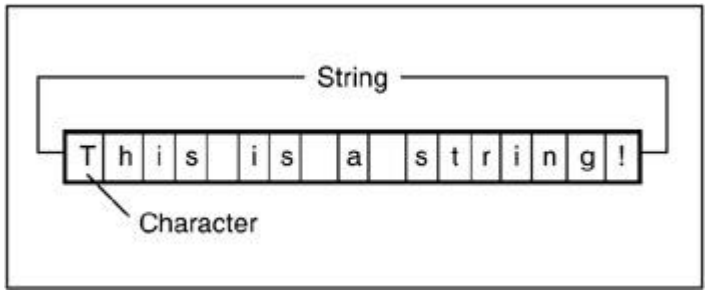
Lớp ArrayList là một phần trong Collection, nó có các đặc điểm sau:

- Rất mềm dẻo, có khả năng tăng hoặc giảm kích thước khi cần.
- Cung cấp một số phương thức hữu ích để tương tác với dữ liệu tập hợp.
- Thêm/Xóa dữ liệu rất đơn giản.
- Có thể sử dụng vòng lặp để truy cập lần lượt từng phần tử.

String

- Người sử dụng muốn **lưu trữ tên** của một người.
- Một giải pháp là tạo **mảng các ký tự** (char).
- Nếu lưu trữ nhiều tên người có thể dùng **mảng đa chiều**.
- Tuy nhiên, số lượng ký tự là **giới hạn**.
- Do đó không thể sử dụng mảng để lưu trữ chuỗi ký tự.
- Java cung cấp **kiểu dữ liệu gọi là String** để lưu trữ dữ liệu các ký tự mà không cần phải tạo mảng.





String

Một số phương thức sử dụng trong String:

public class String	
String(String s)	*create a string with the same value as* `s`
int length()	*number of characters*
char charAt(int i)	*the character at index* `i`
String substring(int i, int j)	*characters at indices* `i` *through* `(j-1)`
boolean contains(String substring)	*does this string contain* `substring` *?*
boolean startsWith(String pre)	*does this string start with* `pre` *?*
boolean endsWith(String post)	*does this string end with* `post` *?*
int indexOf(String pattern)	*index of first occurrence of* `pattern`
int indexOf(String pattern, int i)	*index of first occurrence of* `pattern` *after* `i`
String concat(String t)	*this string with* `t` *appended*
int compareTo(String t)	*string comparison*
String toLowerCase()	*this string, with lowercase letters*
String toUpperCase()	*this string, with uppercase letters*
String replaceAll(String a, String b)	*this string, with* `a`*s replaced by* `b`*s*
String[] split(String delimiter)	*strings between occurrences of* `delimiter`
boolean equals(Object t)	*is this string's value the same as* `t`*'s ?*
int hashCode()	*an integer hash code*

String

length(String str)

- Phương thức `length()` sử dụng để lấy về chiều dài của chuỗi:
  - `String str = "Hello";`
  - `System.out.println(str.length()); // output: 5`

charAt(int index)

- Phương thức `charAt()` lấy về ký tự tại chỉ mục (index) trong tham số.
- Chỉ mục (index) nằm trong khoảng từ 0 tới `length()`.
- Chỉ mục của ký tự đầu tiên là 0. Ví dụ:
  - `System.out.println(str.charAt(2)); // output: 'l'`

concat(String str)

- Phương thức `concat()` dùng để nối chuỗi 2 và chuỗi 1.
- Nếu chiều dài chuỗi thứ 2 là 0 thì kết quả trả về là chuỗi 1, ngược lại sẽ tạo chuỗi đối tượng mới. VD:
  - `System.out.println(str.concat("World"));`
    `// output: 'HelloWorld'`

String

length(String str)

- Phương thức `length()` sử dụng để lấy về chiều dài của chuỗi:
  - `String str = "Hello";`
  - `System.out.println(str.length()); // output: 5`

charAt(int index)

- Phương thức `charAt()` lấy về ký tự tại chỉ mục (index) trong tham số.
- Chỉ mục (index) nằm trong khoảng từ 0 tới `length()`.
- Chỉ mục của ký tự đầu tiên là 0. Ví dụ:
  - `System.out.println(str.charAt(2)); // output: 'l'`

concat(String str)

- Phương thức `concat()` dùng để nối chuỗi 2 và chuỗi 1.
- Nếu chiều dài chuỗi thứ 2 là 0 thì kết quả trả về là chuỗi 1, ngược lại sẽ tạo chuỗi đối tượng mới. VD:
  - `System.out.println(str.concat("World"));`
    `// output: 'HelloWorld'`

compareTo(String str)

- Phương thức `compareTo()` dùng để so sánh 2 chuỗi.
- Kết quả trả về là một số nguyên (int).
- Có 3 dạng giá trị: nhỏ hơn 0, 0 và lớn hơn 0
- Ví dụ so sánh str = "Hello" với:
  - `System.out.println(str.compareTo("World"));`
    `// output: -15`
- Kết quả bằng 15 vì chuỗi "**World**" bắt đầu là '**W**' , ký tự này trong bảng chữ cái nằm sau ký tự '**H**' 15 vị trí ở chuỗi gốc.
- Thứ tự chữ cái tăng dần nên khoảng cách '**H**' and '**W**' là **15**. "H" nhỏ hơn nên kết quả là -15

indexOf(String str)

- Phương thức `indexOf()` trả về chỉ mục (index) đầu tiên của chuỗi str nếu tìm thấy trong chuỗi gốc.
- Nếu không có, kết quả trả về là `-1`. VD,
  - `System.out.println(str.indexOf("e"));  // output: 1`

String

lastIndexOf(String str)

- Phương thức `lastIndexOf()` trả về index vị trí cuối cùng xuất hiện str trong chuỗi gốc.
- Ký tự tìm kiếm sẽ bắt đầu từ cuối chuỗi gốc, VD:
  - `System.out.println(str.lastIndexOf("l")); // output: 3`

replace(char old, char new)

- Phương thức `replace()` thay thế ký tự xuất hiện có sẵn trong chuỗi gốc bằng ký tự mới.
- Nếu không tồn tại ký tự, chuỗi gốc sẽ được trả về, VD:
  - `System.out.println(str.replace('e','a'));`
    `// output: 'Hallo'`

substring(int beginIndex, int endIndex)

- Phương thức `substring()` được dùng để ngắt một phần của chuỗi.
- Chuỗi được ngắt theo tham số bắt đầu và kết thúc, VD:
  - `System.out.println(str.substring(2,5)); // output: 'llo'`

String

toString()

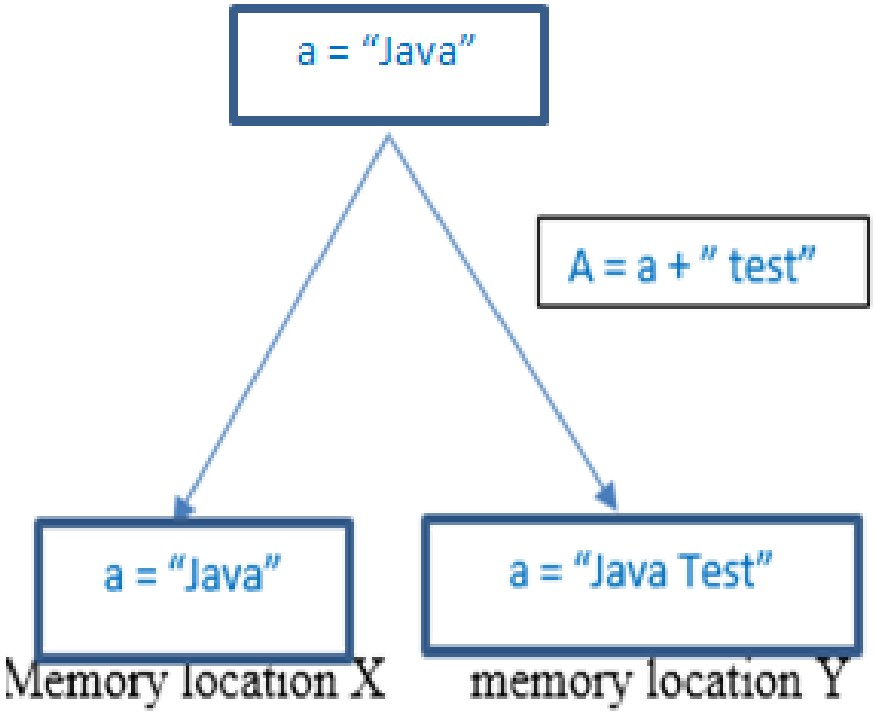
- Phương thức `toString()` trả về đối tượng String.
- Nó được sử dụng để chuyển đổi kiểu dữ liệu khác về kiểu String, VD:
  - `Integer length = 5;`
  - `System.out.println(length.toString()); // output: 5`

trim()

- Phương thức `trim()` trả về chuỗi mới sau khi xóa bỏ đi các khoảng trắng ở đầu và cuối chuỗi, VD:
  - `String str1 = " Hello ";`
  - `System.out.println(str1.trim()); // output: 'Hello'`

String

StringBuilder là lớp sử dụng thao tác với chuỗi tương tự như String nhưng mềm dẻo hơn.



StringBuilder có các phương thức sau:

append()

- Thêm chuỗi mới vào cuối chuỗi gốc.
- VD:
  - `StringBuilder str = new StringBuilder("JAVA ");`
  - `System.out.println(str.append("SE ");`
    `// output: JAVA SE`
  - `System.out.println(str.append(7); // output: JAVA SE 7`

DEVMASTER.VN

StringBuilder có các phương thức sau:

insert()

- Thêm chuỗi với vào chuỗi gốc tại vị trí chỉ định.
- Các hàm insert:
  - `StringBuilder insert(int insertPosition, String str)`
  - `StringBuilder insert(int insertPosition, char ch)`
  - `StringBuilder insert(int insertPosition, float f)`
- VD:

```
StringBuilder str = new StringBuilder("JAVA 7 ");
System.out.println(str.insert(5, "SE");
  // output: JAVA SE 7
```

String

delete()

- Xóa một số ký tự trong chuỗi.
- VD:
  - `StringBuilder str = new StringBuilder("JAVA SE 7");`
  - `System.out.println(str.delete(4,7); // output: JAVA 7`

reverse()

- Đảo ký tự trong chuỗi.
- VD:
  - `StringBuilder str = new StringBuilder("JAVA SE 7");`
  - `System.out.println(str.reverse());`
    `// output: 7 ES AVAJ`

Tóm tắt bài học

- ✓ **Mảng** là vùng lưu trữ dữ liệu đặc biệt có thể cố định **một số giá trị đơn** liên tục trên vùng nhớ.
- ✓ **Mảng đơn** chỉ có một chiều được hiểu như một cột với nhiều dòng.
- ✓ **Mảng đa chiều** là mảng có các phần tử chính là mảng (mảng của mảng).
- ✓ **String** là hằng số và không thay đổi, giá trị của nó không thể thay đổi khi chúng được tạo ra.
- ✓ **StringBuilder** tương tự như String ngoại trừ việc chúng có thể thay đổi.

Thank for watching!