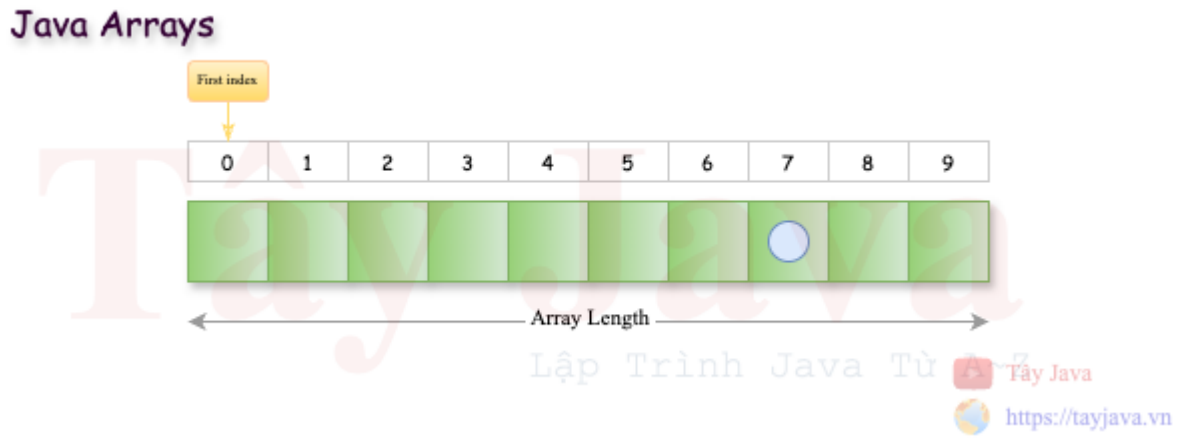
Java Arrays và Java String

1. Java Arrays

1.1 Giới thiệu về Java Arrays

Java Arrays là một đối tượng chứa các phần tử có cùng kiểu dữ liệu. Các phần tử của một mảng được lưu trữ trong một vị trí bộ nhớ liền kề. Đây là một cấu trúc dữ liệu để lưu trữ các phần tử giống nhau. Chúng ta chỉ có thể lưu trữ một tập hợp các phần tử cố định trong một mảng Java.

Chúng ta có thể lưu các giá trị nguyên thuỷ hoặc các đối tượng trong array và tất cả các phần tử đều được đánh chỉ mục.



- Ưu điểm:
    - **Tối ưu code**: Nó làm cho code được tối ưu hoá vì chúng ta dễ dàng sắp xếp và truy xuất dữ liệu.
    - **Truy xuất ngẫu nhiên**: Chúng ta có thể lấy bất kỳ dữ liệu nào theo chỉ mục.
- Nhược điểm:

    Giới hạn kích thước: Chúng ta chỉ có thể lưu trữ kích thước cố định của các phần tử trong mảng. Nó không tăng kích thước khi chạy chương trình. Để giải quyết vấn đề này Java collection có thể tự động gia tăng kích thước của array.

1.2 Các loại Arrays

Có 2 loại array:

- Mảng một chiều (Single Dimensional Array)
- Mảng đa chiều (Multidimensional Array)

1.2.1 Mảng một chiều (Single Dimensional Array)

- Cú pháp

```
// Định nghĩa array
dataType[] arr; (or)
dataType []arr; (or)
dataType arr[];

// Khởi tạo
arrayRefVa r= new datatype[size];
```

- Ví dụ

```
public static void main(String[] args) {

  int arr[] = new int[3];// định nghĩa và tạo array

  arr[0] = 13; // gán phần từ vào mảng
  arr[1] = 49;
  arr[2] = 77;

  // Duyệt mảng
  for (int i = 0; i < arr.length; i++) {
    System.out.println(arr[i]);
  }

  // Mảng string
  String arrS[] = {"A", "B", "C", "D"};
  for (String s: arrS) {
      System.out.println(s);
  }
}

--- Kết quả ---
13
49
77
---
A
B
C
D
```

- Định nghĩa, khai báo và khởi tạo array

```
public static void main(String[] args) {
    //declaration, instantiation and initialization
    int xArr[] = {1, 3, 7};

    // in mảng bằng foreach
    for (int i : xArr) {
        System.out.println(i);
    }
}

--- Kết quả ---
1
3
```

```
7
```

- ArrayIndexOutOfBoundsException

- Ngoại lệ ArrayIndexOutOfBoundsException xảy ra khi chúng ta cố gắng truy xuất phần tử ngoài độ dài của mảng.

```
public static void main(String[] args) {

  int yArr[] = {2,4,6};
  System.out.println("Độ dài của yArr = " + yArr.length);
  System.out.println(yArr[0]);
  System.out.println(yArr[1]);
  System.out.println(yArr[2]);
  // dòng này bị lỗi ArrayIndexOutOfBoundsException do mảng chỉ có 3
phần tử và index = 3 là không tồn tại.
  System.out.println(yArr[3]);
}

--- Kết quả ---
Độ dài của yArr = 3
2
4
6
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:
Index 3 out of bounds for length 3
at src.Main.main(Main.java:32)
```

1.2.2 Mảng đa chiều (Multidimensional Array)

- Cú pháp

```
dataType[][] arr; (or)
dataType [][]arr; (or)
dataType arr[][]; (or)
dataType []arr[];
```

- Ví dụ

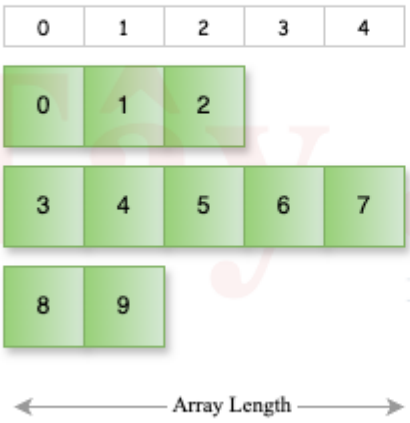
```
int[][] xArr = new int[3][3]; // 3 = row and 3 = column

xArr[0][0] = 1;
xArr[0][1] = 2;
xArr[0][2] = 3;

xArr[1][0] = 4;
xArr[1][1] = 5;
xArr[1][2] = 6;

xArr[2][0] = 7;
xArr[2][1] = 8;
xArr[2][2] = 9;

System.out.println("--- xArr ---");
```

```
for (int i = 0; i < 3; i++) {
  for (int j = 0; j < 3; j++) {
    System.out.println(xArr[i][j]);
  }
  System.out.println();
}

int[][] yArr = {{10, 20, 30}, {40, 50, 60}, {70, 80, 90}};
System.out.println("--- yArr ---");
for (int i = 0; i < 3; i++) {
  for (int j = 0; j < 3; j++) {
    System.out.println(yArr[i][j]);
  }
    System.out.println();
}
```

```
--- Kết quả --
--- xArr ---
1
2
3

4
5
6

7
8
9

--- yArr ---
10
20
30

40
50
60

70
80
90
```

1.2.3 Jagged Array (Mảng răng cưa)

Jagged Arrays




```
public static void main(String[] args) {
  // Mảng 2 chiều
  int arr[][] = new int[3][];
  arr[0] = new int[3];
  arr[1] = new int[5];
  arr[2] = new int[2];

  // Tạo mảng răng cưa
  int count = 0;
  for (int i = 0; i < arr.length; i++)
      for (int j = 0; j < arr[i].length; j++)
          arr[i][j] = count++;

  // in ra mảng
  for (int i = 0; i < arr.length; i++) {
      for (int j = 0; j < arr[i].length; j++) {
          System.out.print(arr[i][j] + " ");
      }
      System.out.println();//new line
  }
}

--- Kết quả ---

0 1 2
3 4 5 6 7
8 9
```

1.2.4 Clone Array

```
public static void main(String[] args) {

    System.out.println("--- zArr ---");
    int[] zArr = yArr[0].clone(); // Clone array
```

```
    for (int i : zArr) {
        System.out.println(i);
    }
}

--- Kết quả ---

10
20
30
```

1.2.5 Copy Array

- Cú pháp

```
public static void arraycopy(
  Object src, int srcPos,Object dest, int destPos, int length
)
```

- Ví dụ

```
public static void main(String[] args) {
    char[] fromArr = {'T', 'a', 'y', 'J', 'a', 'v', 'a'};
    char[] toArr = new char[4];

    // copy Java từ fromArr tới toArr
    System.arraycopy(fromArr, 3, toArr, 0, 4);

    System.out.println(String.copyValueOf(toArr));
}

--- Kết quả ---

Java
```

2. Java String

2.1 Giới thiệu về Java String

Trong Java, String về cơ bản là một đối tượng biểu diễn chuỗi các giá trị char. Một mảng các ký tự hoạt động giống như Java string

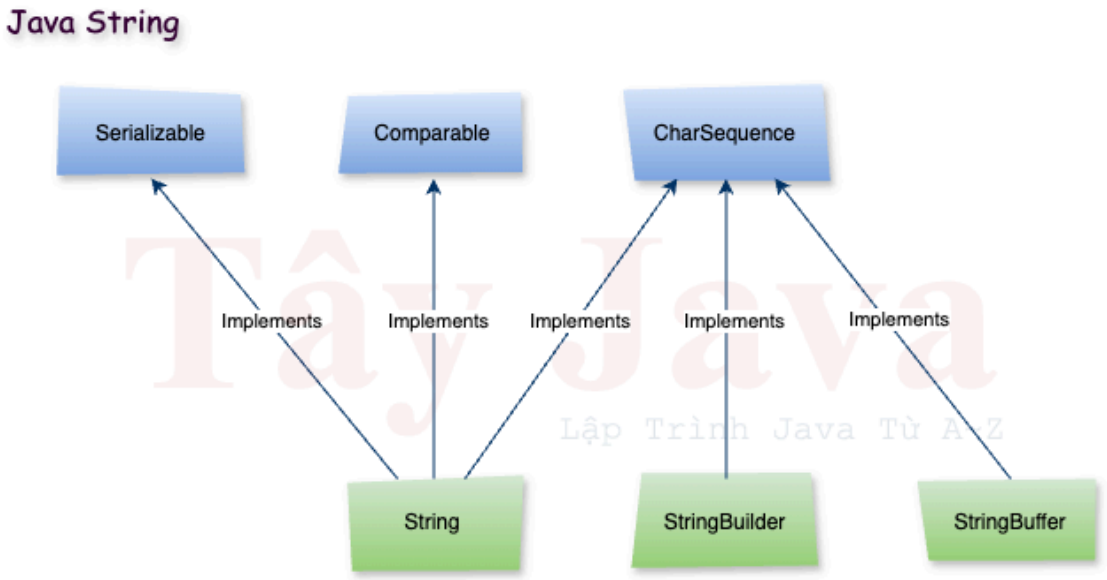
```
char[] ch = {'T', 'a', 'y', 'J', 'a', 'v', 'a'};
String s = new String(ch);
```

→

```
String s = "TayJava";
```

Lớp Java String cung cấp nhiều phương thức để thực hiện các để xử lý String như compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring(), v.v.

Interface CharSequence được sử dụng để biểu diễn chuỗi ký tự. Các lớp String, StringBuffer và StringBuilder đều implements nó. Điều đó có nghĩa là chúng ta có thể tạo một string trong Java bằng cách sử dụng ba lớp này.



Java String là bất biến, nghĩa là không thể thay đổi kích thước của nó. Bất cứ khi nào chúng ta thay đổi bất kỳ string nào đó thì một instance mới sẽ được tạo ra. Nếu bạn muốn sử dụng các chuỗi có thể thay đổi thì có thể dùng StringBuffer và StringBuilder.

2.2 Làm thế nào để khởi tạo một Java String ?

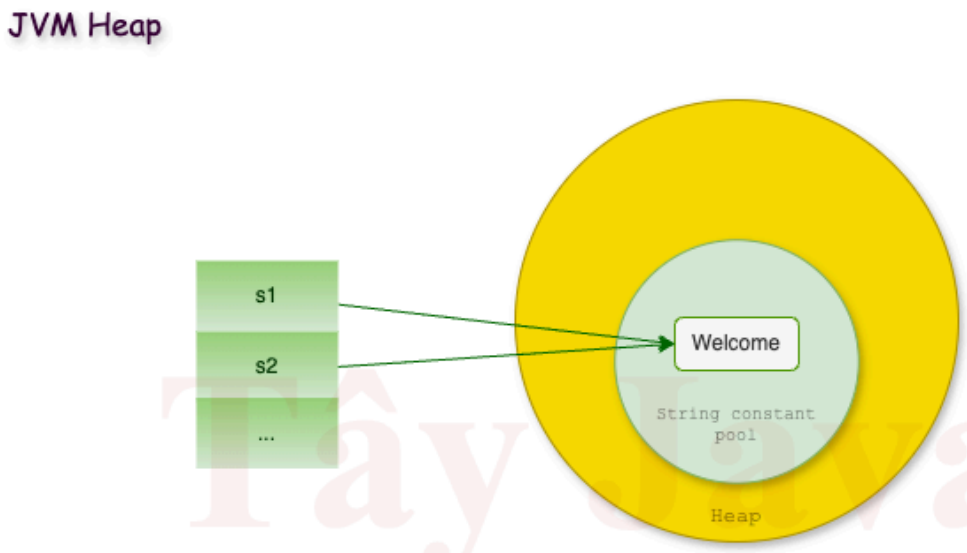
2.2.1 Bằng string literal

String được tạo ra với dấu ""

```
String s1 = "Welcome to Tay Java";
```

Mỗi lần bạn tạo ra một String, JVM sẽ kiểm tra "string constant pool" trước. Nếu String đã tồn tại trong pool thì một tham chiếu đến instance đó sẽ được tạo ra. Nếu String đó chưa tồn tại thì JVM sẽ tạo string mới.

```
String s1 = "Welcome to Tay Java";
String s2 = "Welcome to Tay Java"; // JVM không tạo một string mới
```



Tại sao Java sử dụng khái niệm String literal?

Để làm cho Java sử dụng bộ nhớ hiệu quả hơn bởi vì không có đối tượng mới nào được tạo nếu nó đã tồn tại trong Spring constant pool.

2.2.2 Bằng từ khóa new

```
String s1 = new String("Welcome to Tay Java");
```

Trong trường hợp như vậy, JVM sẽ tạo một đối tượng String mới trong bộ nhớ heap bình thường (không phải pool) và giá trị "Welcome" theo nghĩa đen sẽ được đặt trong **Spring constant pool.** Biến s sẽ tham chiếu đến đối tượng trong heap (không phải pool).

2.2.3 Các method trong class String

#	Method	Mô tả
1	char charAt(int index)	Trả về giá trị char cho index cụ thể
2	int length()	Trả về độ dài của string
3	static String format(String format, Object... args)	Trả về một String đã được định dạng
4	static String format(Locale locale, String format, Object... args)	Trả về một String đã được định dạng theo ngôn ngữ đã cho

5	String substring(int beginIndex)	Trả về substring từ vị trí index cho trước
6	String substring(int beginIndex, int endIndex)	Trả về substring từ index bắt đầu đến index kết thúc
7	boolean contains(CharSequence s)	Nó trả về giá trị đúng hoặc sai sau khi khớp với string giá trị char
8	static String join(CharSequence delimiter, CharSequence... elements)	Trả về một string đã nối
9	static String join(CharSequence delimiter, Iterable<? extends CharSequence> elements)	Trả về một string đã nối
10	boolean equals(Object another)	Kiểm tra xem string có giống với đối tượng đã cho hay không
11	boolean isEmpty()	Kiểm tra string có bị rỗng không (null hoặc blank) không ?
12	String concat(String str)	Nối string đã chỉ định
13	String replace(char old, char new)	Thay thế tất cả các lần xuất hiện của giá trị char được chỉ định
14	String replace(CharSequence old, CharSequence new)	Thay thế tất cả các lần xuất hiện của CharSequence đã chỉ định
15	static String equalsIgnoreCase(String another)	So sánh với một string khác không kiểm tra chữ hoa chữ thường
16	String[] split(String regex)	Trả về một string phân tách khớp với regex
17	String[] split(String regex, int limit)	Trả về một string phân tách khớp với regex và limit
18	String intern()	Trả về một string đã được đồng bộ trong String constant pool
19	int indexOf(int ch)	Trả về giá trị char được chỉ định

20	int indexOf(int ch, int fromIndex)	Trả về giá trị char được chỉ định bắt đầu theo index đã cho
21	int indexOf(String substring)	Trả về chỉ mục string con được chỉ định
22	int indexOf(String substring, int fromIndex)	Trả về chỉ mục string con được chỉ định bắt đầu theo index đã cho
23	String toLowerCase()	Trả về một string thường
24	String toLowerCase(Locale locale)	Trả về một string thường theo ngôn ngữ được chỉ định
25	String toUpperCase()	Trả về một STRING IN HOA
26	String toUpperCase(Locale locale)	Trả về một STRING IN HOA theo ngôn ngữ được chỉ định
27	String trim()	Loại bỏ khoảng trắng ở trước và sau String
28	static String valueOf(int value)	Chuyển đổi kiểu từ dữ liệu đã cho thành string. Đây là overloading method

3. Câu hỏi phỏng vấn

1. So sánh string s1 và s2

```
String s1 = "Tay Java";
String s2 = new String("Tay Java");

if (s1.equals(s2)) {
  System.out.println("giống nhau");
} else {
  System.out.println("khác nhau");
}

if (s1 == s2) {
  System.out.println("giống nhau");
} else {
  System.out.println("khác nhau");
}
```

Kết quả in sẽ là gì và tại sao ?