1. Đặc điểm và tính chất của String

* Tính bất biến

String s = "Hello";

s.concat(" World");

System.out.println(s);

Output: Hello

* String pool: lưu trữ các chuỗi theo cách tối ưu bằng cách sử dụng String Pool để giảm thiểu việc tạo ra các đối tượng String trùng lặp.

String s1 = "Hello";

String s2 = "Hello";

System.out.println(s1 == s2);

Output: true

* Hỗ trợ các phương thức như length(), charAt(), substring(), indexOf(), toLowerCase(), toUpperCase(), trim()…

String s = "Hello World";

System.out.println(s.length());

Output: 11

* Tương thích vs mảng: String có thể được chuyển đổi qua lại với mảng ký tự (char[]).

char[] charArray = {'H', 'e', 'l', 'l', 'o'};

String s = new String(charArray);

System.out.println(s);

Output: Hello

* Ngoài String bất biến, còn cung cấp StringBuilder và StringBuffer là các lớp có thể thay đổi

StringBuilder sb = new StringBuilder("Hello");

sb.append(" World");

System.out.println(sb.toString());

Output: Hello World

1. Các cách tạo String

* Tạo trực tiếp(literals)

String str = "Hello, World!";

* Tạo bằng từ khóa new

String str= new String(“Hello”);

1. String pool

* là một khu vực đặc biệt trong bộ nhớ heap của Java, nơi mà các đối tượng String được lưu trữ và quản lý để tối ưu hóa bộ nhớ. Khi một chuỗi được tạo ra, Java sẽ kiểm tra String Pool trước để xem chuỗi đó đã tồn tại hay chưa. Nếu chuỗi đã tồn tại, Java sẽ sử dụng lại đối tượng đó thay vì tạo một đối tượng mới.
* Cách hoạt động

Tạo bằng literals:

String s1 = "Hello";

String s2 = "Hello";

System.out.println(s1 == s2);

Output: true

* Vì cùng tham chiếu đến đối tượng String pool

Tạo bằng từ khóa new:

String s3 = new String("Hello");

String s4 = new String("Hello");

System.out.println(s3 == s4);

Output: false

System.out.println(s3.equals(s4));

Output: true

* 2 biến trên là 2 đối tượng khác nhau dù cùng nội dung

Dùng phương thức intern:

String s5 = new String("Hello").intern();

String s6 = "Hello";

System.out.println(s5 == s6);

Output: true

* S5 đc gọi intern nó sẽ tham chiếu đến đối tượng chuỗi trong intern như trên là s6 thay vì tạo chuỗi mới
* Lợi ích của String pool: tiết kiệm bộ nhớ , tăng hiệu suất
* Hạn chế: việc quản lí và kiểm tra String pool gây tốn thời gian

1. So sánh chuỗi:

Có 2 cách:

* Sử dụng equal(): so sánh giá trị của chuỗi

public boolean equals(Object another) so sánh chuỗi này với đối tượng được chỉ định.

String s1="string";

String s2="string";

String s3=new String("string");

System.out.println(s1.equals(s2));//true

System.out.println(s1.equals(s3));//true

public boolean equalsIgnoreCase(String another) so sánh chuỗi này với chuỗi khác, bỏ qua chữ hoa chữ thường.

String s1="string";

   String s2="STRING";

System.out.println(s1.equals(s2));//false

   System.out.println(s1.equalsIgnoreCase(s2));//true

* Sử dụng toán tử ==

Toán tử == so sánh các tham chiếu chứ không phải giá trị.

String s1="string";

String s2="string";

String s3=new String("string");

System.out.println(s1==s2);

System.out.println(s1==s3);

1. So sánh String, String Builder, String Buffer

|  |  |  |  |
| --- | --- | --- | --- |
| Đặc điểm | String | StringBuffer | StringBuilder |
| Có thể thay đổi | không | có | Có |
| Hiệu suất | Chậm khi có nhiều thay đổi | Nhanh | Nhanh |
| An toàn trong đa luồng | Kh cần | không | Có |
| Khi nào nên sd | Khi chuỗi ít thay đổi | Khi chuỗi nhiều thay đổi và đơn luồng | Khi chuỗi nhiều thay đổi và đa luồng |

// String

String str = "Hello";

str.concat(“World”);

System.out.println("String: " + str);

// StringBuilder

StringBuilder sb = new StringBuilder("Hello");

sb.append(" World"); // Thay đổi trực tiếp trên đối tượng sb

System.out.println("StringBuilder: " + sb.toString());

// StringBuffer

StringBuffer sbf = new StringBuffer("Hello");

sbf.append(" World"); // Thay đổi trực tiếp trên đối tượng sbf

System.out.println("StringBuffer: " + sbf.toString());

1. Static

* Khái niệm: là từ khóa được sử dụng để khai báo các thành phần của class thuộc về class thay vì các đối tượng của class.
* Khi nào nên sd static

Sử dụng thuộc tính static: khi muốn tạo 1 biến dùng chung cho tất cả đối tượng của class và Khi giá trị của biến không phụ thuộc vào các đối tượng cụ thể của class mà thuộc về class nói chung.

Sử dụng phương thức static: Khi phương thức không cần truy cập hoặc sửa đổi các thuộc tính hoặc phương thức không static của class và Khi phương thức thực hiện các thao tác không phụ thuộc vào trạng thái của các đối tượng của class

* Truy cập tới phương thức và thuộc tính static bằng 2 cách : thông qua tên class và thông qua đối tượng của class

Truy cập tới thuộc tính:

public class Main {  
 static class MyClass {  
 static int *staticVariable* = 10; // Khai báo thuộc tính static  
 }  
 public static void main(String[] args) {  
 // Truy cập thuộc tính static thông qua tên lớp  
 System.*out*.println(MyClass.*staticVariable*);   
  
 // Truy cập thuộc tính static thông qua đối tượng của lớp  
 MyClass obj = new MyClass();  
 System.*out*.println(obj.*staticVariable*); }  
}

Truy cập tới phương thức static:

public class Main {  
 static class MyClass {  
 static void staticMethod() {  
 System.*out*.println("This is a static method.");  
 }  
 }  
 public static void main(String[] args) {  
 // Truy cập phương thức static thông qua tên lớp  
 MyClass.*staticMethod*();  
  
 // Truy cập phương thức static thông qua đối tượng của lớp  
 MyClass obj= new MyClass();  
 obj.*staticMethod*();  
  
 }  
}

1. Final

* Final đc sử dụng để chỉ định rằng một biến, phương thức, hoặc lớp không thể thay đổi sau khi được khởi tạo hoặc khai báo.
* Khai báo 1 biến final khác gì với static, biến khai báo bằng final có thể chỉnh sửa được không?

Final: đảm bảo biến hoặc phương thức kh bị thay đổi sau khi khởi tạo

Static: xác định 1 biến hoặc phương thức thuộc về class chứ kh thuộc về đối tượng

Biến khai báo bằng final kh chính sửa đc

* Ví dụ:
* public class Main {  
   static class MyClass {  
   final int myfinal1=10;  
   final int myfinal2;  
   MyClass(int value){  
   this.myfinal2=value;  
   }  
   }  
   public static void main(String[] args) {  
   MyClass obj=new MyClass(50);  
   System.*out*.println(obj.myfinal1);  
   System.*out*.println(obj.myfinal2);  
    
   }  
  }