Lập trình Java

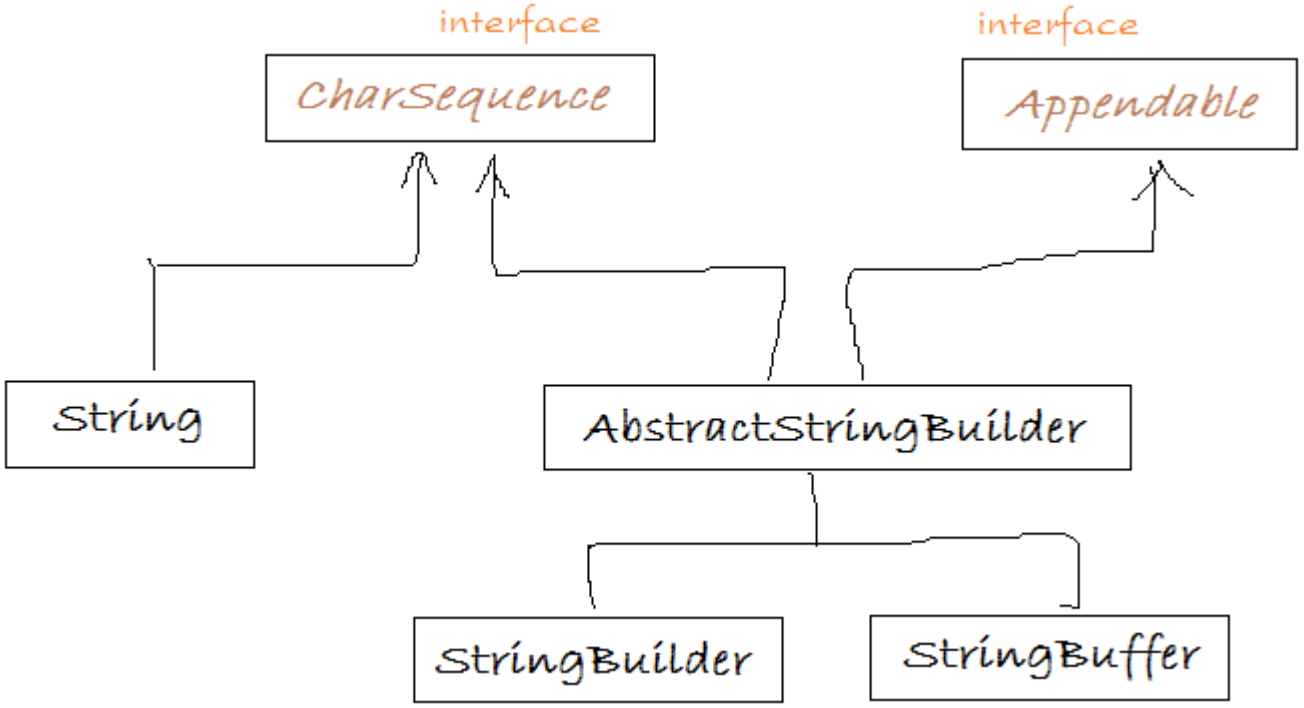
Tuần 5: String

Nội dung

- Tìm hiểu về lớp String, StringBuffer và StringBuilder

String, StringBuffer và StringBuilder

Phân cấp kế thừa



CharSequence Interface được sử dụng để biểu diễn chuỗi các ký tự.

Lớp String

- Gồm nhiều các phương thức để thực hiện các thao tác với chuỗi như: compare(), equals(), split(), length(), replace(), compareTo(),…
- Đặc điểm:

  String là không thể thay đổi (immutable), *tất cả các thay đổi trên String đều tạo ra một đối tượng String khác*, là dạng final nên không có lớp nào được kế thừa nó.

Lớp String

- Immutable và mutable
- Ví dụ:

```
public class ImmutableClassExample  {
    private int value;
    private String name;
    public ImmutableClassExample(String name, int value)  {
          this.value = value;
          this.name= name;
    }
    public String getName()  {
          return name;
    }
    public int getValue()  {
         return value;
    }
}
```

Ví dụ: String có nhiều thuộc tính (trường), ví dụ length,... nhưng các giá trị đó là không thể thay đổi

Lớp String

- String vừa có tính đối tượng và vừa có tính nguyên thủy (primitive). Thể hiện thông qua cách khởi tạo: String Literal và String Object.

Lớp String

+String Literal (chuỗi chữ): được lưu trữ trong ngăn xếp

VD: String str1 = "Hello";

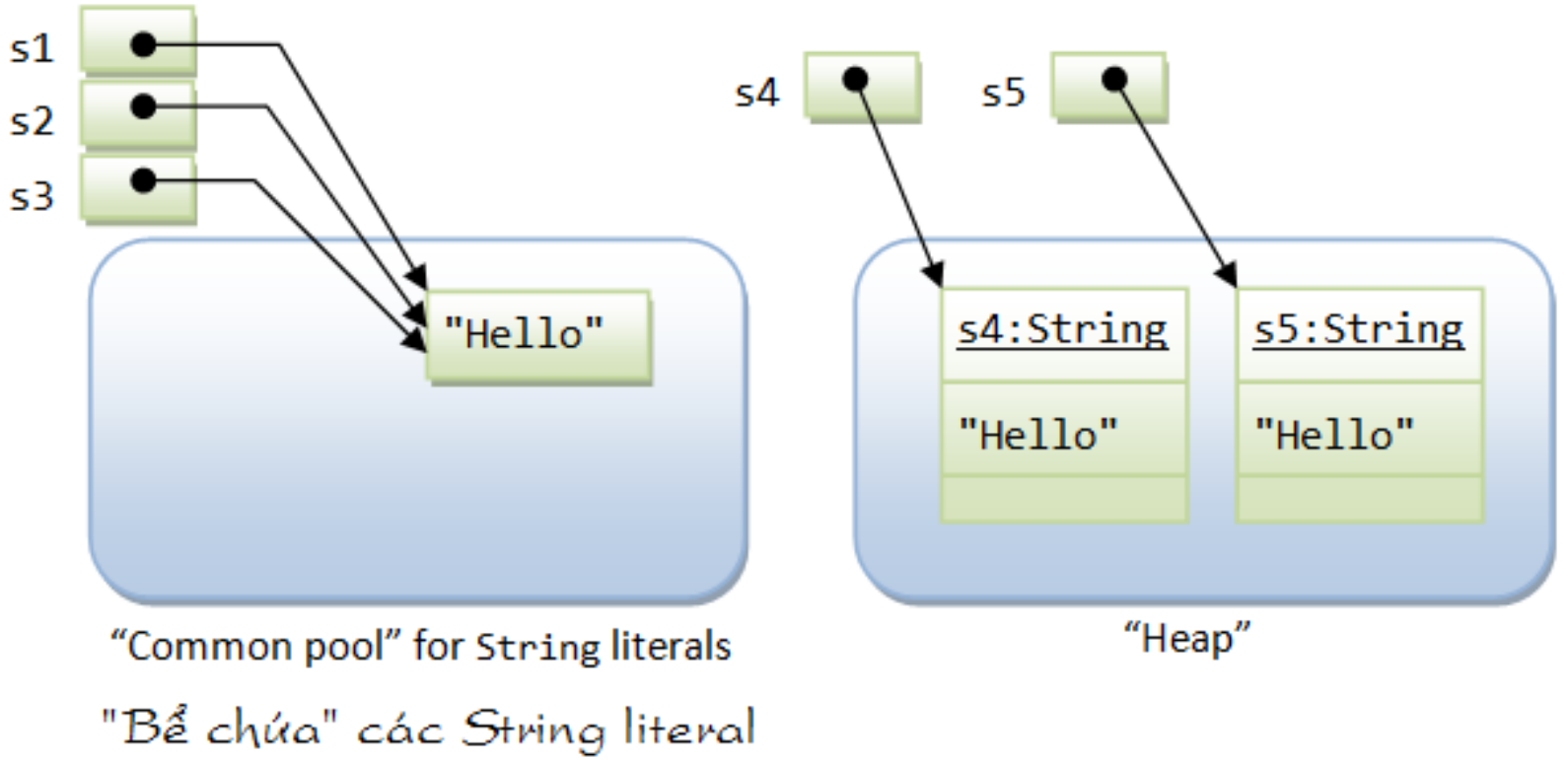
Các string literal giống nội dung, sẽ chia sẻ cùng một vị trí lưu trữ trong một bể chứa (common pool).

+String Object: vì nó là một class nên được khởi tạo bằng từ khóa new

VD: String str2 = new String(“Hello");

Các đối tượng String lưu trữ trong Heap, và không chia sẻ vị trí lưu trữ kể cả 2 đối tượng string này có nội dung giống nhau.

Lớp String



Thông thường nên dùng dạng String Literal để tiết kiệm bộ nhớ.

Lớp String

Các phương thức của String

char charAt(int index)	Trả về giá trị char cho chỉ số cụ thể.
int length()	Trả về độ dài chuỗi.
String substring(int beginIndex)	Trả về chuỗi con bắt đầu từ chỉ số index.
String substring(int beginIndex, int endIndex)	Trả về chuỗi con từ chỉ số bắt đầu đến chỉ số kết thúc.
boolean isEmpty()	Kiểm tra chuỗi rỗng.
boolean equals(Object another)	kiểm tra sự tương đương của chuỗi với đối tượng.
String replace(char old, char new)	Thay thế tất cả giá trị char cụ thể bằng một giá trị char mới.

Lớp String

Các phương thức của String

int indexOf(int ch)	Trả về vị trí của ký tự ch cụ thể.
int indexOf(int ch, int fromIndex)	Trả về vị trí của ký tự ch tính từ từ vị trí fromIndex.
int indexOf(String substring)	Trả về vị trí của chuỗi con substring.
String toUpperCase()	Trả về chuỗi chữ hoa.
String toLowerCase()	Trả về chuỗi chữ thường.
String trim()	Xóa khoảng trắng ở đầu và cuối của chuỗi.
static String valueOf(int value)	Chuyển đổi giá trị kiểu dữ liệu đã cho thành chuỗi.

Lớp String

Chú ý khi dùng phương thức equals() và dấu “==” để so sánh String.

+ equals() so sánh về nội dung.

+ dấu == so sánh về vùng nhớ.

Ví dụ:

```
String s1 = "VLTH";
String s2 = s1;
String s3 = new String("VLTH");
String s4 = new String("VLTH");
```

```
s1==s2; // true
s1==s3; // false
s3==s4; // false
s1.equals(s2); //true
s1.equals(s3); //true
s3.equals(s4); //true
```

Phương thức toString

- Phương thức toString() trả về chuỗi đại diện của đối tượng. Nếu không viết lại thì mặc định sẽ hiện thị dang:

```
public String toString(){
        return getClass().getName() + '@' +
Integer.toHexString(hashCode());
    }
```

Phương thức toString

```
public class Person {
        private name;
        private age;

        public Person(name, age){this.name = name, this.age = age}

         public static void main(String... a) {
             Person person = new Person ("Đỗ Trung Quân", "25");
             System.out.println("Person.Tostring : " + person .toString());
         }
     }
```

Output: Person.Tostring : Person@15db9742

Phương thức toString

```
public class Person{
        private name;
        private age;

        public Person(name, age){this.name = name, this.age = age}
        @Override
        public String toString(){
            return "Name:" + this.name+ "\nAge: " + this.age;
        }
        public static void main(String... a) {
            Person person = new Person ("Đỗ Trung Quân", "25");
            System.out.println("Person.Tostring : " + person .toString());
        }
 }
```

Output: Person.Tostring :
Name: Đỗ Trung Quân
Age : 25

StringBuffer và StringBuilder

StringBuffer, StringBuilder có thể thay đổi, được dùng khi dữ liệu nhiều.

Khi xử lý văn bản sử dụng nhiều luồng (Thread) nên sử dụng StringBuffer để tránh tranh chấp giữa các luồng.

Khi xử lý văn bản sử dụng 1 luồng (Thread) nên sử dụng StringBuilder.

Tốc độ xử lý StringBuilder là tốt nhất, sau đó StringBuffer và cuối cùng mới là String.

Bài tập trên lớp tuần 5

Cho chuỗi số: 012345abc67d89

Xử lý chuỗi trên sao cho kết quả trả về: 0123456789