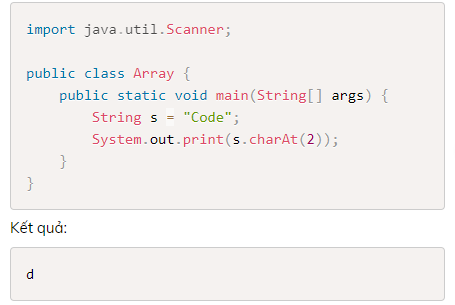
**Xử lý chuỗi**

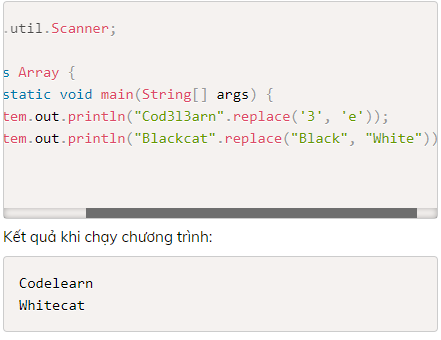
**charAt():** trả về một ký tự trong string

**VD:**



**Replace():** thay thế chuỗi/ký tự được tìm thấy thành chuỗi/ký tự khác.

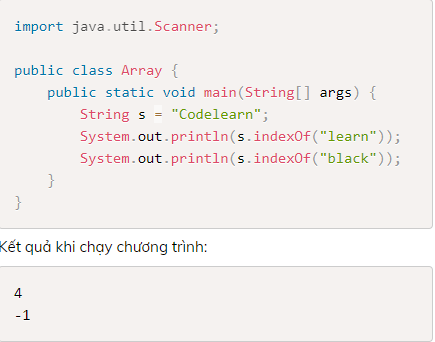
VD:



**toUpperCase/toLowerCase()**: chuyển các ký tự từ in thường thành in hoa và ngược lại.

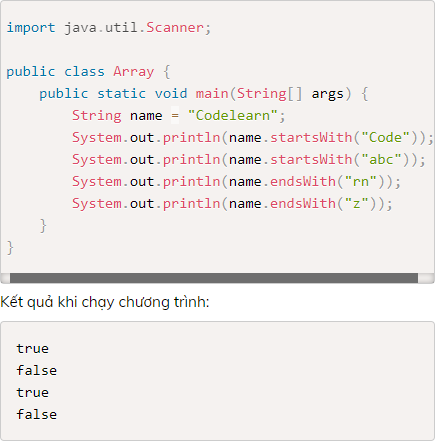
**indexOf():** trả về vị trí xuất hiện đầu tiên của 1 chuỗi trong 1 chuỗi khác, nếu không tìm thấy trả về -1

VD:



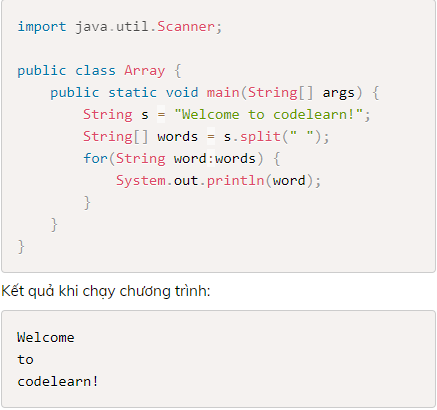
**startsWith / endsWith :** dùng để kiểm tra một xâu có bắt đầu hoặc kết thúc bằng một chuỗi khác hay không

VD:



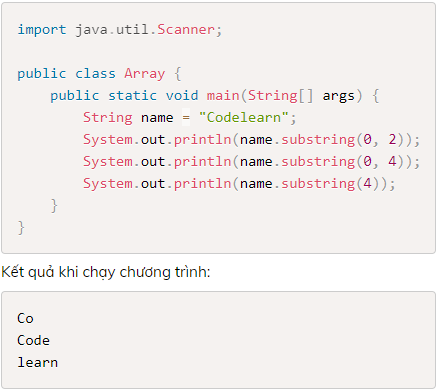
**Split():** tách 1 xâu ra thành mảng các xâu dựa trên 1 xâu cho trước

VD:



**Substring():** lấy ra 1 xâu con dựa trên chỉ số bắt đầu và chỉ số kết thúc của 1 xâu khác

VD:



**Overloading**(nạp chồng phương thức): là 1 phương thức cùng tên nhưng khác tham số, kiểu tham số

**Overide** (ghi đè): Nếu lớp con có cùng phương thức như đã được khai báo trong lớp cha, thì đó gọi là Ghi đè phương thức

Qui tắc cho ghi đè phương thức

Phương thức phải có cùng tên như trong lớp cha.

Phương thức phải có cùng tham số như trong lớp cha.

Phải là quan hệ IS-A (kế thừa).

**Đa hình:** Khi một tác vụ được thực hiện theo nhiều cách khác nhau được gọi là tính đa hình. Chúng ta có thể thực hiện tính đa hình trong Java bởi nạp chồng phương thức và ghi đè phương thức.

**Trừu tượng (abstract):**

Phương thức: khai báo bằng từ khóa abstract phía trước tên phương thức và phương thức không có hành động (chỉ khai báo tên thôi).

**Kế thừa**(Java chỉ cho kế thừa đơn – chỉ kế thừa từ 1 lớp cha duy nhất): extends

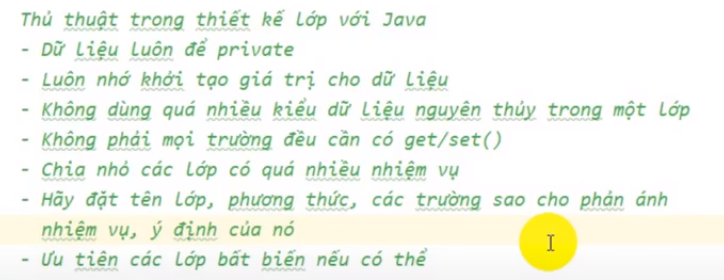
**STATIC (bộ nhớ tạm thời)**

Tạo ra dùng để sử dụng chung 1 giá trị. Là thành phần dùng chung của class

Các phương thức có cùng **static** với nhau thì mới thao tác được với nhau

Từ khóa **super :** gọi đến phương thức khởi tạo của class cha (phải gọi ở đầu phương thức), cũng có thể gọi đến phương thức của class cha bằng **super.**

**Qui chuẩn thiết kế class:**



**Regular Expression**

|  |  |  |
| --- | --- | --- |
| TT | Biểu thức chính quy | Mô tả |
| 1 | . | Khớp (match) với bất kỳ ký tự nào |
| 2 | ^regex | Biểu thức chính quy phải  khớp tại điểm bắt đầu |
| 3 | regex$ | Biểu thức chính quy phải khớp ở cuối dòng. |
| 4 | [abc] | Thiết lập định nghĩa, có thể khớp với a hoặc b hoặc c. |
| 5 | [abc][vz] | Thiết lập định nghĩa, có thể khớp với a hoặc b hoặc c theo sau là v hay z. |
| 6 | [^abc] | Khi dấu ^ xuất hiện như là nhân vật đầu tiên trong dấu ngoặc vuông, nó phủ nhận mô hình. Điều này có thể khớp với bất kỳ ký tự nào ngoại trừ a hoặc b hoặc c. |
| 7 | [a-d1-7] | Phạm vi: phù hợp với một chuỗi giữa a và điểm d và con số từ 1 đến 7. |
| 8 | X|Z | Tìm X hoặc Z. |
| 9 | XZ | Tìm X và theo sau là Z. |
| 10 | $ | Kiểm tra kết thúc dòng. |
|  | | |
| 11 | \d | Số bất kỳ, viết ngắn gọn cho [0-9] |
| 12 | \D | Ký tự không phải là số, viết ngắn gon cho [^0-9] |
| 13 | \s | Ký tự khoảng trắng, viết ngắn gọn cho [ \t\n\x0b\r\f] |
| 14 | \S | Ký tự không phải khoản trắng, viết ngắn gọn cho [^\s] |
| 15 | \w | Ký tự chữ, viết ngắn gọn cho [a-zA-Z\_0-9] |
| 16 | \W | Ký tự không phải chữ, viết ngắn gọn cho [^\w] |
| 17 | \S+ | Một số ký tự không phải khoảng trắng (Một hoặc nhiều) |
| 18 | \b | Ký tự thuộc a-z hoặc A-Z hoặc 0-9 hoặc \_, viết ngắn gọn cho [a-zA-Z0-9\_]. |
|  | | |
| 19 | \* | Xuất hiện 0 hoặc nhiều lần, viết ngắn gọn cho {0,} |
| 20 | + | Xuất hiện 1 hoặc nhiều lần, viết ngắn gọn cho {1,} |
| 21 | ? | Xuất hiện 0 hoặc 1 lần, ? viết ngắn gọn cho {0,1}. |
| 22 | {X} | Xuất hiện X lần, {} |
| 23 | {X,Y} | Xuất hiện trong khoảng X tới Y lần. |
| 24 | \*? | \* có nghĩa là xuất hiện 0 hoặc nhiều lần, thêm ? phía sau nghĩa là tìm kiếm khớp nhỏ nhất. |