Giới thiệu về xử lý chuỗi và mảng

Buổi học 3

Kiểu dữ liệu chuỗi và các hàm thông dụng

Trong ngôn ngữ lập trình Java, chuỗi được coi là 1 dữ liệu dạng đối tượng (tức là nó có các thuộc tính và phương thức)

- Các hàm được update từ phiên bản 9 trở đi:

    + Repeat(solan_lap): Hàm nhân số lượng chuỗi lên n lần.

    + Text Block trong chuỗi, khi nào lên sử dụng text block.

    String xmlString = """

        <language> <no>%s</no>

        <name>%s</name> </language> """.formatted("12345", "Java");

- Dùng text block khi cần xử lý text cho cấu trúc như html hoặc câu truy vấn sql

```
@NamedQuery(name = "findByCustomerNumberAndCity", query =
""" from Customer c
        where c.customerNo = :customerNo
        and c.city = :city
""")
```

- indent(inInsertOrRemove): Hàm insert hoặc remove khoảng trắng trong chuỗi.
- transform(Functions -> {}): Hàm lambda xử lý các chuỗi bằng lambda.

```
public class TransformExample {
	public static void main(String[] args) {
		String str = "1000";
		Integer integer = str.transform(Integer::parseInt);
		System.out.println(integer);
	}
 }
```

- strip(): Hàm remove các khoảng trắng trong chuỗi.

Hàm strim() cũng là hàm remove các khoảng trắng trong chuỗi vậy cần strip để làm gì .?

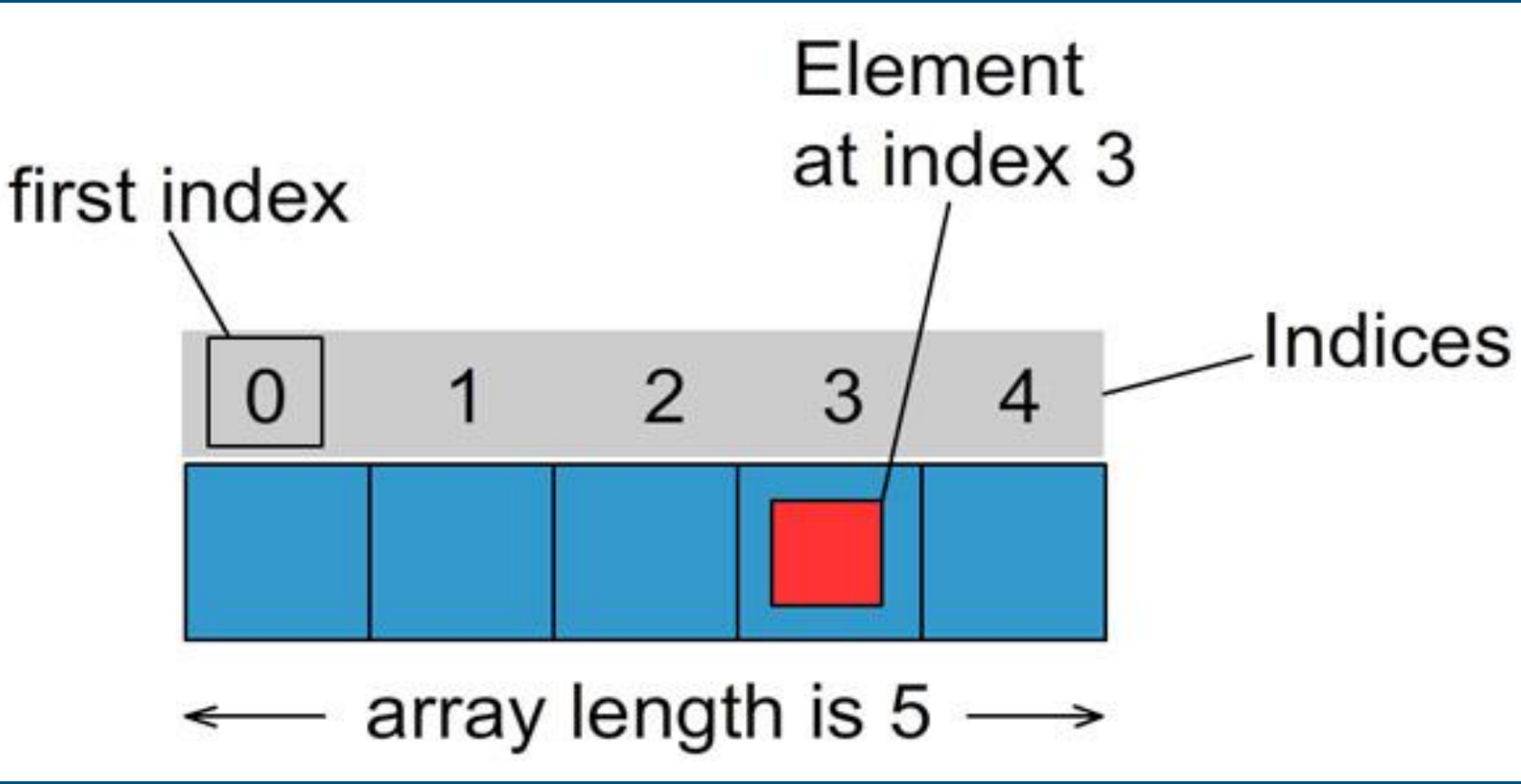
- isBlank, lines: kiểm tra chuỗi rỗng hoặc chỉ có khoảng trắng và số dòng của chuỗi hiện tại.

- String là bất biến (immutable) tức là không thể thay đổi. Có nghĩa là khi nào bạn thay đổi giá trị của bất kỳ chuỗi nào thì một instance mới được tạo ra.
- Đối với chuỗi có thể thay đổi, bạn có thể sử dụng các lớp StringBuffer và StringBuilder.

Giới thiệu về mảng tĩnh

Mảng (array) là một tập hợp các phần tử có cùng kiểu dữ liệu, có địa chỉ tiếp nhau trên bộ nhớ (memory). Mảng có số phần tử cố định và bạn không thể thay đổi kích thước của nó.

Các phần tử của mảng được đánh chỉ số (index), bắt đầu từ chỉ số 0. Bạn có thể truy cập vào các phần tử của nó thông qua chỉ số.


Element
at index 3
first index
Indices
0
1
2
3
4
array length is 5

Mảng hai chiều là tập hợp các một chiều được sắp xếp theo chiều ngang hoặc dọc tạo thành hàng và cột.

Index	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9	10
1	11	12	13	14	15	16	17	18	19	20
2	21	22	23	24	25	26	27	28	29	30
3	31	32	33	34	35	36	37	38	39	40
4	41	42	43	44	45	46	47	48	49	50

Thao tác với Arrays

Các hàm để thao tác với mảng như:

create, compare, sort, search, stream, and transform arrays.

CopyOf and copyOfRange

```
String[] intro = new String[] { "once", "upon", "a", "time" };

String[] abridgement = Arrays.copyOfRange(storyIntro, 0, 3);

assertArrayEquals(new String[] { "once", "upon", "a" }, abridgement);

assertFalse(Arrays.equals(intro, abridgement));
```

CopyOf

```
String[] revised = Arrays.copyOf(intro, 3);

String[] expanded = Arrays.copyOf(intro, 5);

assertArrayEquals(Arrays.copyOfRange(intro, 0, 3), revised);

assertNull(expanded[4]);
```

fill

```
String[] stutter = new String[3];

Arrays.fill(stutter, "once");

assertTrue(Stream.of(stutter) .allMatch(el -> "once".equals(el));
```

Comparing

Object[] story = new Object[] { intro, new String[] { "chapter one", "chapter two" }, end };

Object[] copy = new Object[] { intro, new String[] { "chapter one", "chapter two" }, end };

assertTrue(Arrays.deepEquals(story, copy)); assertFalse(Arrays.equals(story, copy));

Sorting and Searching

```
String[] sorted = Arrays.copyOf(intro, 4);

Arrays.sort(sorted);

assertArrayEquals( new String[]{ "a", "once", "time", "upon" }, sorted);
```

BinarySearch

```
int exact = Arrays.binarySearch(sorted, "time");

int caseInsensitive = Arrays.binarySearch(sorted, "TiMe",
String::compareToIgnoreCase);

assertEquals("time", sorted[exact]);

assertEquals(2, exact); assertEquals(exact, caseInsensitive);
```

Streaming

```
Assert.assertEquals(Arrays.stream(intro).count(), 4);

exception.expect(ArrayIndexOutOfBoundsException.class);

Arrays.stream(intro, 2, 1).count();
```

asList

```
List<String> rets = Arrays.asList(storyIntro);

assertTrue(rets.contains("upon"));

assertTrue(rets.contains("time"));

assertEquals(rets.size(), 4);
```

setAll

```
String[] longAgo = new String[4];

Arrays.setAll(longAgo, i -> this.getWord(i));

assertArrayEquals(longAgo, new String[]{"a","long","time","ago"});
```