Javaプログラミング実習

24. 文字列操作

株式会社ジードライブ

今回学ぶこと

- 文字列を操作するための方法
  - Stringクラス
  - StringBuilderクラス

Stringとは

- 固定文字列を扱うためのクラス
  - 内部的には**char型**の配列を扱っている
  - new演算子を使用することなく、ダブルクォーテーションでインスタンスを生成することが可能

手順を踏んだ書き方

```
char[] letters = {'あ', 'い', 'う'};
String tx = new String(letters);
```

使用頻度が高いので、ダブルクォーテーションだけで使えるようになっている

```
String tx = "あいう";
```

複数行の文字列

- 複数行の文字列を表現する場合には ¥n または、テキストブロック """ を使用する

¥nを使用する場合

```
String tx = "こんにちは¥n元気ですか¥n私は元気です";
```

テキストブロックを使用する場合

```
String tx = """
    こんにちは
元気ですか
    私は元気です
    """;
```

場合によっては、インデントをもうけない方が良い場合もある

Stringのメソッド

- メソッドの一例

メソッド	説明
`boolean startsWith(String str)`	文字列を比較するためのメソッドで、指定された文字列で始まる場合、trueを返す。 文字列を比較するためのメソッドとしては、`equals()`, `endsWith()`, `contains()`, `matches()` といったものも存在する
`int length()`	文字数を返す
`boolean isEmpty()`	文字数が0字の場合、trueを返す。 スペースやタブといった空白文字のみの場合にtrueを返す `isBlank()` というメソッドも存在する

Stringクラス: https://docs.oracle.com/javase/jp/21/docs/api/java.base/java/lang/String.html

Stringのメソッド

- メソッドの一例

メソッド	説明
`String replace(String oldStr, String newStr)`	ある文字列を別の文字列で置き換え、新たな文字列を生成する
`String[] split(String delimiter)`	一つの文字列オブジェクトを指定されたデリミタ(区切り文字)で分割し、配列にして返す
`String trim()`	空白文字を取り除いた文字列を返す。 先頭と末尾の空白を取り除く場合は、`strip()`というメソッドを使用する
`int indexOf(String str)`	指定された文字列が最初に出現する位置のインデックスを返す 文字列の末尾から出現位置を調べるメソッドとして、`lastIndexOf()` も存在する
`String substring(int start, int end)`	指定された位置の部分文字列を返す

Stringクラス: https://docs.oracle.com/javase/jp/17/docs/api/java.base/java/lang/String.html

Stringのメソッド

例：入力された文字列のチェック

```
System.out.println("「pen」という単語を含む文章を入力して下さい");
String text = scanner.nextLine();
if(text.isBlank()) {
  System.out.println("文章が入力されていません");
}
else if(text.length() <= 3) {
  System.out.println("文章が短すぎます");
}
else if(!text.contains("pen")) {
  System.out.println("「pen」が含まれていません");
}
else {
  System.out.println("入力を受け付けました");
}
```

Stringのメソッド

例：CSV(カンマ区切り)のデータから情報を取り出す

```
String csvData = "山田太郎,25,東京都";

String[] info = csvData.split(",");

System.out.println("氏名: " + info[0]);
System.out.println("年齢: " + info[1]);
System.out.println("住所: " + info[2]);
```

```
氏名: 山田太郎
年齢: 25
住所: 東京都
```

練習

- 練習24-1
- 練習24-2

StringBuilder

- Stringは固定文字列であり、Stringで文字列操作を多用するとメモリの消費が多くなり実行時間もかかる

```
String tx = "あいう";
tx += "えお";   // tx = new String("あいうえお");
tx += "かきく"; // tx = new String("あいうえおかきく");
tx += "けこ";   // tx = new String("あいうえおかきくけこ");
```

Stringは固定文字列なので、内部的には新しくオブジェクトを生成しているため、メモリの消費が多くなってしまう

- **StringBuilder**は可変長の文字列を扱うためのクラスとして存在する
  - 文字の連結や文字列の一部削除などの文字列操作がStringを使った操作に比べて高速に行える

StringBuilderのメソッド

- メソッドの一例

メソッド	説明
`StringBuilder append(String str)`	指定された文字列を追加する(数値など他のデータ型も追加可能)
`StringBuilder insert(int offset, String str)`	指定された文字列を指定された位置に挿入する(他のデータ型も挿入可能)
`int indexOf(String str)`	指定された文字列が最初に出現する位置のインデックスを返す 文字列の末尾から出現位置を調べるメソッドとして、`lastIndexOf()` も存在する
`StringBuilder replace(int start, int end, String str)`	指定された場所の文字列を指定された文字列に置き換える
`StringBuilder reverse()`	文字を逆順に並べ替える
`String substring(int start, int end)`	指定された位置の部分文字列を返す

StringBuilderクラス: https://docs.oracle.com/javase/jp/21/docs/api/java.base/java/lang/StringBuilder.html

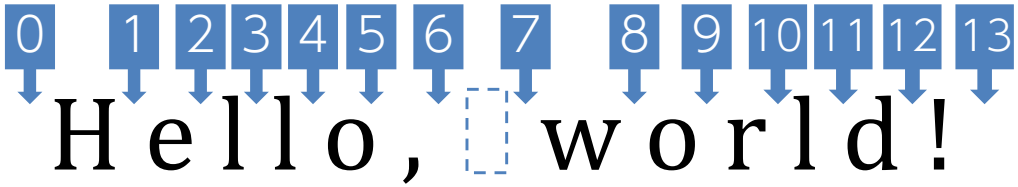
StringBuilderのメソッド

例：文字列内の単語を別の単語に置き換える

```
StringBuilder text = new StringBuilder("Hello, world!");
int start = text.indexOf("world");   // world の開始位置「7」
text.replace(start, start + 5, "Java");  // start + 5 : 5文字分を置き換え
System.out.println(text);
```

```
Hello, Java!
```

インデックスの位置



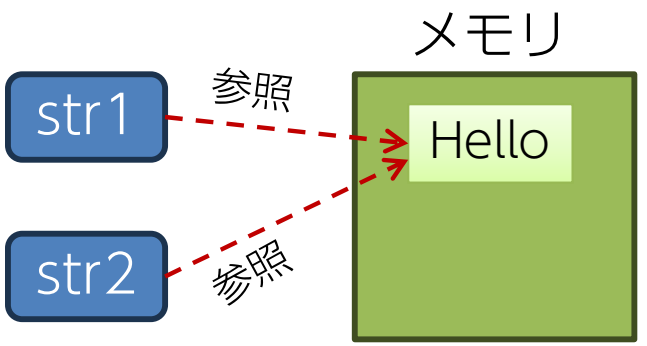
練習

- 練習24-3

イミュータブル(不変)

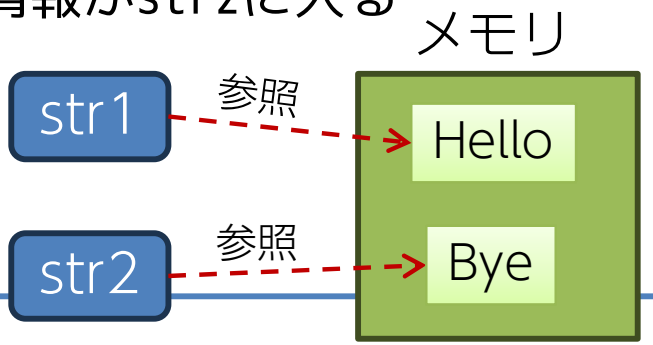
- Stringはイミュータブル(状態が変わらない)という性質をもつ

```
// メモリ上にHelloが記憶され、その参照情報がstr1に入る
String str1 = "Hello";
// str1, str2の参照先が同じになる
String str2 = str1;
```




```
// Stringはイミュータブルなので、参照先の文字列は変わらない
// メモリ上に新しくByeが記憶され、その参照情報がstr2に入る
str2 = "Bye";

// Hello が出力される
System.out.println(str1);
```



イミュータブル(不変)

- Stringはreplace()のようなメソッドを使っても、内部的に保持する文字列は変わらない、という点にい注意する

StringBuilder

```
StringBuilder sb = new StringBuilder("私はリンゴが好き");
int start = sb.indexOf("リンゴ");
sb.replace(start, start + 3, "オレンジ");
System.out.println(sb); // 私はオレンジが好き
```

String

```
String text = "私はリンゴが好き";
text.replace("リンゴ", "オレンジ");
System.out.println(text); // 私はリンゴが好き(変わっていない)

// Stringオブジェクトが保持する文字列を変えるには再代入が必要
text = text.replace("リンゴ", "オレンジ");
System.out.println(text); // 私はオレンジが好き
```

練習

- 練習24-4