7-11.メソッド

この章では、JavaSilver試験において頻出の、また注意すべきメソッド(API)について説明します。

これらの詳細な仕様や、他に気になるメソッドがあった場合は、
公式のAPIリファレンスで検索して確かめてみましょう！

Java SE11 公式APIリファレンス

charAt()

```
//記述形式
文字列.charAt(戻したい文字列インデックス番号)

//処理内容
文字列から、指定のインデックス番号の1文字を戻す

//戻り値
char型

//記述例
public class Main {
  public static void main(String[] args) {
    String str = "Java Silver";
    System.out.println(str.charAt(0));
    System.out.println(str.charAt(3));
    System.out.println(str.charAt(5));
  }
}

//出力結果
J
a
S
```

文字列に対して、インデックス番号は0から始まっているので引数の指定に注意です。

文字列	J	a	v	a		S	i	l	v	e	r
インデックス番号	0	1	2	3	4	5	6	7	8	9	10

?

isAlphabetic()、isDigit()

```
//記述形式
isAlphabetic(char型の1文字)

//処理内容
char型の引数がアルファベットかどうか調べる

//戻り値
Boolean型
アルファベットならtrue、それ以外ならfalseを返す


//記述形式
isDigit(char型の1文字)

//処理内容
char型の引数が数字かどうか調べる

//戻り値
Boolean型
数字ならtrue、それ以外ならfalseを返す

//記述例
public class Main {
    public static void main(String[] args) {
        char a = 'A';
        char b = 'あ';
        char c = '山';
        char d = '3';

        System.out.println("---isAlphabetic---");
        System.out.println(Character.isAlphabetic(a));
        System.out.println(Character.isAlphabetic(b));
        System.out.println(Character.isAlphabetic(c));
        System.out.println(Character.isAlphabetic(d));

        System.out.println("---isDigit---");
        System.out.println(Character.isDigit(a));
        System.out.println(Character.isDigit(b));
        System.out.println(Character.isDigit(c));
        System.out.println(Character.isDigit(d));
    }
}

//出力結果
---isAlphabetic---
true
true
true
false
---isDigit---
false
false
false
true
```

`isDigit()`の、数字かどうかの判断はイメージしやすいですね。
(ちなみに全角でも判別できます)

しかし、`isAlphabetic()`は少し注意が必要で、
アルファベットというともちろんA,B,Cといった英語のものが思い浮かびますが、
実際には日本語のひらがなや漢字、ギリシャ文字などもtrueで返します。
数字や記号以外の、人間の言語に関する文字は大体trueの判別がされると押さえておきましょう。

startsWith()、 endsWith()

```
//記述形式
文字列.startsWith(検索する文字列)

//処理内容
文字列の始まりが指定の文字列と一致するかを調べる

//戻り値
Boolean型


//記述形式
文字列.endsWith(検索する文字列)

//処理内容
文字列の末尾が指定の文字列と一致するかを調べる

//戻り値
Boolean型

//記述例
public class Main {
    public static void main(String[] args) {
        String str = "JavaSilver";

        System.out.println("---startsWith---");
        System.out.println(str.startsWith("Java"));
        System.out.println(str.startsWith("Javaaaa"));

        System.out.println("---endsWith---");
        System.out.println(str.endsWith("ver"));
        System.out.println(str.endsWith("Sver"));
    }
}

//出力結果
---startsWith---
true
false
---endsWith---
true
false
```

substring()

```
//記述形式
文字列.substring(開始位置のインデックス番号[, 終了位置のインデックス番号])
※カッコ [ ]内は省略可の意味です。

//処理内容
指定の範囲の文字を切り取る。
第二引数は省略可能。その場合は第一引数から最後の文字までを切り取る。

//戻り値
Stringクラス型

//記述例
public class Main {
    public static void main(String[] args) {
        String str = "JavaSilver";

        System.out.println(str.substring(0));
        System.out.println(str.substring(4));
        System.out.println(str.substring(2,6));

        //第二引数の設定は注意。
        String str2 = "0123456";
        System.out.println(str2.substring(2,6));
    }
}

//出力結果
JavaSilver
Silver
vaSi
2345
```

切り取る範囲は、文字列のインデックス番号を元にすると考えやすいです。

しかし、第二引数の捉え方は少し注意しましょう。

例として、`substring(2,6)`は、インデックス番号2から5までの4文字が切り取られます。

第二引数の手前までとなってますね。

このようなインデックス番号による範囲指定は、下の図のように文字の間に区切り線をイメージすると捉えやすいです。

replace()

```
//記述形式
java.lang.Stringクラス
文字列.replace(検索する文字列, 置換する文字列) //引数2つ

java.lang.StringBuilderクラス
文字列.replace(指定インデックス1, 指定インデックス2, 置換する文字列) //引数3つ

//処理内容
指定した文字列、またはインデックスの範囲を、別の文字列に置換する

//戻り値
Stringクラス

//記述例
public class Main {
    public static void main(String[] args) {

        String str1 = "JavaSilver";
        System.out.println(str1.replace("Silver", "Bronze"));

        StringBuilder str2 = new StringBuilder("JavaSilver");
        System.out.println(str2.replace(4, 10, "Gold"));

    }
}

//出力結果
JavaBronze
JavaGold
```

indexOf()

```
//記述形式
文字列.indexOf(探したい文字列)

//処理内容
文字列の中に指定の文字列があるかを調べ、あれば最初に現れる場所のインデックス番号を返す

//戻り値
int型
文字列が見つかればそのインデックス番号、
見つからなければ負の値(-1)を戻す

//記述例
public class Main {
    public static void main(String[] args) {
        String str = "JavaSilver";

        System.out.println(str.indexOf("v"));
        System.out.println(str.indexOf("va")); //引数は1文字以上であれば可
        System.out.println(str.indexOf("vaaa")); //このまとまりの文字は見つからない
        System.out.println(str.indexOf("Silver"));
    }
}

//出力結果
2
2
-1
4
```

sort()

```
//記述形式
Arrays.sort(配列);

//処理内容
配列内容をソートする

//戻り値
無し
戻り値を返すのではなく、変数の中にある配列の中身を直接並び替えて書き換えます。

//記述例

import java.util.Arrays; //インポートが必要

public class Main {
    public static void main(String[] args) {
        int[] a = {5, 1, 333, 15, 9};

        //Arraysクラス指定の記述が必要
        Arrays.sort(a);

        System.out.println("---数値ソート---");
        System.out.println(Arrays.toString(a));

        String[] b = {"ABC", "CBA", "あああ","B", "CA", "AAA"};

        Arrays.sort(b);

        System.out.println("---文字列ソート---");
        System.out.println(Arrays.toString(b));
    }
}

//出力結果
---数値ソート---
[1, 5, 9, 15, 333]
---数値ソート---
[AAA, ABC, B, CA, CBA, あああ]
```

List.of()

```
//記述形式
List.of(リストに入れる複数の要素)

//処理内容
引数の値を格納した、不変なリストを作成する。
中身の変更や削除が出来ない。

//戻り値
Listクラス型

//記述例
import java.util.List;

public class Main {
    public static void main(String[] args) {
        List<String> arr = List.of("A", "B", "C");
        arr.add("D"); //リストに追加しようとする要素
    }
}

//出力結果
Exception in thread "main" java.lang.UnsupportedOperationException
```

不変なリストを操作しようとすると、
UnsupportedOperationExceptionという例外が発生します。

pow()

```
//記述形式
Math.pow(基準の数値, 累乗する数値)

//処理内容
第二引数の数だけ、第一引数の数を累乗する

//戻り値
double型

//記述例
public class Main {
    public static void main(String[] args) {
        System.out.println(Math.pow(3, 3)); //3の3乗 (3×3×3)
        System.out.println(Math.pow(2, 4));
        System.out.println(Math.pow(10, 3));
    }
}

//出力結果
27.0
16.0
1000.0
```

equals()、mismatch()、compare()

```
//記述形式
Arrays.equals(配列、配列)

//処理内容
2つの配列を比較し、等しいかどうかを調べる

//戻り値
boolean型
等しければtrue、等しくなければfalse


//記述形式
Arrays.mismatch(配列、配列)

//処理内容
2つの配列を比較し、最初に不一致な箇所を調べ、そのインデックス番号を返す。
不一致の箇所が無ければ、-1を返す。

//戻り値
int型
インデックス番号または-1


//記述形式
Arrays.compare(配列、配列)

//処理内容
2つの配列を比較し、辞書的にどちらが大きいかを調べる。
2つの配列が等しければ0を返す。
2つの配列が違う場合、第一引数<第二引数であれば-1、第一引数>第二引数であれば1を返す。

//戻り値
int型
0,1,-1

//記述例
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        int[] a = {1,2,3,4};
        int[] b = {1,2,3,4};
        int[] c = {1,2,8,3};
        int[] d = {1,2,2,8};

        System.out.println("---equals---");
        System.out.println(Arrays.equals(a,b)); //2つが等しい
        System.out.println(Arrays.equals(a,c)); //2つが異なる

        System.out.println("---mismatch---");
        System.out.println(Arrays.mismatch(a,b)); //2つが等しい
        System.out.println(Arrays.mismatch(a,c)); //2つがインデックス2の場所で異なる

        System.out.println("---compare---");
        System.out.println(Arrays.compare(a,b)); //2つが等しい
        System.out.println(Arrays.compare(a,c)); //2つがインデックス2の場所で異なり、配列aの"3"は、配列cの"8"よりも小さい
        System.out.println(Arrays.compare(a,d)); //2つがインデックス2の場所で異なり、配列aの"3"は、配列dの"2"よりも大きい
    }
}


//出力結果
---equals---
true
false
---mismatch---
-1
2
---compare---
0
-1
1
```

それぞれ配列の比較ですが、どのような基準で比較しているか、戻り値はどのような形かに注意しましょう。

課題

満点を取れるまで小テストを受験して下さい。

制限時間: 30 分

評定方法: 最高評点

合格点: 100 / 100

受験件数: 77