String.split()で分解した時の長さゼロの文字列の話

2025/03/05
自分用。長さゼロか否かの判定は String.isEmpty() を使うのが安全ぽい。

テストコード

String.Split()の処理結果で長さゼロの配列要素の確認を行う。

Chks.java

```
public class Chks {

    public static void main(String[] args) {

        System.out.println(String.format("Version: %s",
System.getProperty("java.version")));

        String[] parts1 = "A,B,C".split(",", -1);
        String[] parts2 = "A,,C".split(",", -1);
        String[] parts3 = "A,  ,C".split(",", -1);
        String[] parts4 = "  ,  ,  ".split(",", -1);
        String[] parts5 = ",,".split(",", -1);

        check(1, parts1);
        check(2, parts2);
        check(3, parts3);
        check(4, parts4);
        check(5, parts5);
    }

    private static void check(int idx, String[] strs) {

        System.out.print(String.format("%d : strs:[%d];", idx,
strs.length));

        for(String item : strs) {
            try {
                System.out.print(String.format("var:[%s];", item));

                if (item == null) System.out.print("null;");

                if (item == "") System.out.print("zeroLen;");

                if (item.isEmpty()) System.out.print("empty;");
```

```
                System.out.print(String.format("len=%d;",
item.length()));

            } catch (Exception e) {
                //e.printStackTrace();
            }
        }

        System.out.println();
    }
}
```

実行結果

String.split()で分解した時に長さゼロの文字列(カラの要素)の扱いが違っている。出力結果2行目と5行目がそれ。
Java11までは判定に以下を使うのはよろしくない模様。

```
if (str == "") { … }
```

Java17では可能□String.isEmpty() もしくは String.length() を使う方が安全な感じ。

バージョン	結果
Java8	Version: 1.8.0_372 1 : strs:[3];var:[A];len=1;var:[B];len=1;var:[C];len=1; 2 : strs:[3];var:[A];len=1;var:[];empty;len=0;var:[C];len=1; 3 : strs:[3];var:[A];len=1;var:[ ];len=2;var:[C];len=1; 4 : strs:[3];var:[ ];len=2;var:[ ];len=2;var:[ ];len=2; 5 : strs:[3];var:[];empty;len=0;var:[];empty;len=0;var:[];empty;len=0;
Java11	Version: 11.0.19 1 : strs:[3];var:[A];len=1;var:[B];len=1;var:[C];len=1; 2 : strs:[3];var:[A];len=1;var:[];empty;len=0;var:[C];len=1; 3 : strs:[3];var:[A];len=1;var:[ ];len=2;var:[C];len=1; 4 : strs:[3];var:[ ];len=2;var:[ ];len=2;var:[ ];len=2; 5 : strs:[3];var:[];empty;len=0;var:[];empty;len=0;var:[];empty;len=0;
Java17	Version: 17.0.7 1 : strs:[3];var:[A];len=1;var:[B];len=1;var:[C];len=1; 2 : strs:[3];var:[A];len=1;var:[];zeroLen;empty;len=0;var:[C];len=1; 3 : strs:[3];var:[A];len=1;var:[ ];len=2;var:[C];len=1; 4 : strs:[3];var:[ ];len=2;var:[ ];len=2;var:[ ];len=2; 5 : strs:[3];var:[];zeroLen;empty;len=0;var:[];zeroLen;empty;len=0;var:[];zeroLen;empty;len=0;

追記

Java11までは長さゼロのStringインスタンスを個別に割り当てしていて□Java17は長さゼロのStringインスタンスを共有して割り当ててるのかな？
Stringの値が長さゼロかの確認は String.isEmpty()□String同士の値の一致確認は String.equals()を使うのが安全なのは変わらない。

追記2

System.identityHashCode()メソッドでインスタンスを確認できるようなので試してみる。条件によっては一意にならない可能性があるらしいけどこの実行時にはそれに当てはまらないものとしよう。

```
public class Chks {

    public static void main(String[] args) {

        System.out.println(String.format("Version: %s",
System.getProperty("java.version")));

        String[] parts1 = "A,B,C".split(",", -1);
        String[] parts2 = "A,,C".split(",", -1);
        String[] parts3 = "A,  ,C".split(",", -1);
        String[] parts4 = "  ,  ,  ".split(",", -1);
        String[] parts5 = ",,".split(",", -1);

        check(1, parts1);
        check(2, parts2);
        check(3, parts3);
        check(4, parts4);
        check(5, parts5);
    }

    private static void check(int idx, String[] strs) {

        System.out.print(String.format("%d : strs:[%d];", idx,
strs.length));

        for(String item : strs) {
            try {
                System.out.print(String.format("[%s]=%08x;", item,
System.identityHashCode(item)));
            } catch (Exception e) {
                //e.printStackTrace();
            }
        }
        System.out.println();
    }
}
```

結果、長さゼロの文字列のハッシュコードから□Java8,Java11では別インスタンスが割り当てられ□Java17では同じインスタンスを割り当てていることが分かった。
おそらくString.split()は実行するまで結果が分からないから個別にインスタンスを作るけど、長さゼロの文字列についてはインスタンスを共有する事になる模様。

バージョン	結果
Java8	Version: 1.8.0_372 1 : strs:[3];[A]=4554617c;[B]=74a14482;[C]=1540e19d; 2 : strs:[3];[A]=677327b6;[]=014ae5a5;[C]=7f31245a; 3 : strs:[3];[A]=6d6f6e28;[ ]=135fbaa4;[C]=45ee12a7; 4 : strs:[3];[ ]=330bedb4;[ ]=2503dbd3;[ ]=4b67cf4d; 5 : strs:[3];[]=7ea987ac;[]=12a3a380;[]=29453f44;
Java11	Version: 11.0.19 1 : strs:[3];[A]=1888ff2c;[B]=35851384;[C]=649d209a; 2 : strs:[3];[A]=6adca536;[]=357246de;[C]=28f67ac7; 3 : strs:[3];[A]=256216b3;[ ]=2a18f23c;[C]=0d7b1517; 4 : strs:[3];[ ]=16c0663d;[ ]=23223dd8;[ ]=4ec6a292; 5 : strs:[3];[]=1b40d5f0;[]=0ea4a92b;[]=3c5a99da;
Java17	Version: 17.0.7 1 : strs:[3];[A]=682a0b20;[B]=3d075dc0;[C]=214c265e; 2 : strs:[3];[A]=448139f0;[]=7cca494b;[C]=7ba4f24f; 3 : strs:[3];[A]=3b9a45b3;[ ]=7699a589;[C]=58372a00; 4 : strs:[3];[ ]=04dd8dc3;[ ]=6d03e736;[ ]=568db2f2; 5 : strs:[3];[]=7cca494b;[]=7cca494b;[]=7cca494b;

追記3

String.split()の結果**ではない**場合**はどうなのか**String.split()の結果と同じ配列を書いて確かめる。

```
public class Chks {

    public static void main(String[] args) {

        System.out.println(String.format("Version: %s",
System.getProperty("java.version")));

        String[] parts1 = {"A","B","C"};
        String[] parts2 = {"A","" ,"C"};
        String[] parts3 = {"A","  ","C"};
        String[] parts4 = {"  ","  ","  "};
        String[] parts5 = {"","",""};

        check(1, parts1);
        check(2, parts2);
        check(3, parts3);
        check(4, parts4);
        check(5, parts5);
    }

    private static void check(int idx, String[] strs) {

        System.out.print(String.format("%d : strs:[%d];", idx,
strs.length));

        for(String item : strs) {
            try {
```

```
                System.out.print(String.format("[%s]=%08x;", item,
System.identityHashCode(item)));
            } catch (Exception e) {
                //e.printStackTrace();
            }
        }
        System.out.println();

    }
}
```

結果は、さすがにコンパイルの時点で同じ定数値インスタンスを参照する内容になっている。

バージョン	結果
Java8	Version: 1.8.0_372 1 : strs:[3];[A]=4554617c;[B]=74a14482;[C]=1540e19d; 2 : strs:[3];[A]=4554617c;[]=677327b6;[C]=1540e19d; 3 : strs:[3];[A]=4554617c;[ ]=014ae5a5;[C]=1540e19d; 4 : strs:[3];[ ]=014ae5a5;[ ]=014ae5a5;[ ]=014ae5a5; 5 : strs:[3];[]=677327b6;[]=677327b6;[]=677327b6;
Java11	Version: 11.0.19 1 : strs:[3];[A]=6e3c1e69;[B]=1888ff2c;[C]=35851384; 2 : strs:[3];[A]=6e3c1e69;[]=649d209a;[C]=35851384; 3 : strs:[3];[A]=6e3c1e69;[ ]=6adca536;[C]=35851384; 4 : strs:[3];[ ]=6adca536;[ ]=6adca536;[ ]=6adca536; 5 : strs:[3];[]=649d209a;[]=649d209a;[]=649d209a;
Java17	Version: 17.0.7 1 : strs:[3];[A]=682a0b20;[B]=3d075dc0;[C]=214c265e; 2 : strs:[3];[A]=682a0b20;[]=448139f0;[C]=214c265e; 3 : strs:[3];[A]=682a0b20;[ ]=7cca494b;[C]=214c265e; 4 : strs:[3];[ ]=7cca494b;[ ]=7cca494b;[ ]=7cca494b; 5 : strs:[3];[]=448139f0;[]=448139f0;[]=448139f0;

java, 技術資料, String.split(), 長さゼロ判定