double型の話

15桁で999,999,999,999,999の表示が…という話があったので自分用に整理。

環境

Windows11上でJava11利用

まずは簡単な値で確認

double型の内部

検索するとdouble型の情報が色々あると思う。double型の内部形式を見てみる。

NumericalDigitDataTest.java

```
public class NumericalDigitDataTest {

    public static void main(String[] args) {

        double doubleVal   = 1D; // double型リテラルを差し替えて確認してみる

        String prefixStr   = "0000000000000000000000000000000000000000000000000000000000000000"; //64桁の"0"
        long   d2l         = Double.doubleToLongBits(doubleVal);
        String l2bs        = prefixStr + Long.toBinaryString(d2l);
        int lastidx        = l2bs.length();

        System.out.printf("%s\n", l2bs.substring(lastidx-64, lastidx));
    }
}
```

上記コードの理解で正しければdouble型の値と対応するビット構成は以下の表になる。

doubleVal	符号	指数部	仮数部	正規化前の仮数部
0D	0	00000000000	0000000000000000000000000000000000000000000000000000	0.0000000000000000000000000000000000000000000000000000
1D	0	01111111111	0000000000000000000000000000000000000000000000000000	1.0000000000000000000000000000000000000000000000000000
2D	0	10000000000	0000000000000000000000000000000000000000000000000000	1.0000000000000000000000000000000000000000000000000000
3D	0	10000000000	1000000000000000000000000000000000000000000000000000	1.1000000000000000000000000000000000000000000000000000
4D	0	10000000001	0000000000000000000000000000000000000000000000000000	1.0000000000000000000000000000000000000000000000000000
-5D	1	10000000001	0100000000000000000000000000000000000000000000000000	1.0100000000000000000000000000000000000000000000000000
4.5D	0	10000000001	0010000000000000000000000000000000000000000000000000	1.0010000000000000000000000000000000000000000000000000
5e-1D (5×10⁻¹)	0	01111111110	0000000000000000000000000000000000000000000000000000	1.0000000000000000000000000000000000000000000000000000
5e+1D (5×10¹)	0	10000000100	1001000000000000000000000000000000000000000000000000	1.1001000000000000000000000000000000000000000000000000

指数部がゼロの時は仮数部もそのままの値として扱う。1D時の指数部は0。
指数部がゼロでなく1023以上の場合は仮数部の値から1023を引いた値になるので□2D時の指数部は 1024 - 1023 = 1□4D時の指数部は 1025 - 1023 = 2 になる。
指数部がゼロでなく1023より小さい場合は1023から仮数部の値を引いた結果になるので□5e-1D時の指数部は 1023 - 1022 = 1 になる。

doubleVal	指数部	差	符号	計算	結果(10進数)
0D	0	-		-	+0
1D	1023	0	正	1.000 の小数点位置を右に0移動 = +1.000	+1
2D	1024	1	正	1.000 の小数点位置を右に1移動 = +10.000	+2
3D	1024	1	正	1.100 の小数点位置を右に1移動 = +11.000	+3
4D	1025	2	正	1.000 の小数点位置を右に2移動 = +100.000	+4
-5D	1025	2	負	1.010 の小数点位置を右に2移動 = －101.000	-5
4.5D	1025	2	正	1.001 の小数点位置を右に2移動 = +100.100	+4.5
5e-1D	1022	1	正	1.000 の小数点位置を左に1移動 = +0.100	+0.5
5e+1D	1028	5	正	1.1001 の小数点位置を右に5移動 = +110010.000	+50.0

よし 999,999,999,999,999 で見てみる

符号	指数部	仮数部	正規化前の仮数部
0	10000110000	1100011010111111010100100110001100111111111111111000	1.1100011010111111010100100110001100111111111111111000

10000110000(2)=1072(10)なので、指数部は1072-1023=49
1.1100011010111111010100100110001100111111111111111000 の小数点位置を右に49移動 = 11100011010111111010100100110001100111111111111111.000
11100011010111111010100100110001100111111111111111(2) = +999999999999999(10)

理屈はあってそう。

化ける？

元の話は double型を15桁の文字列(実際はバイト換算で15バイトのバイト列)にしようとしたら化けたという話。

化けないよ？

先の結果から化ける要素は無さそうなんだけど確かめる。

コード	結果

NumericalDigitData2Test.java
`public class NumericalDigitDataTest {`
 	
`    public static void main(String[] args) {`
 	
`        double doubleVal = 9999999999999999D;`
`        String str       = String.format("%15.0f", doubleVal);`
 	
`        System.out.printf("toStr[%s]\n", str);`
`    }`
`}`
	`toStr[9999999999999999]`

桁数も合ってる。

小数点以下が含まれてると…

想定の出力にならないのは元の値に小数点以下が含まれてるケースが多かったです。

コード	結果

NumericalDigitData3Test.java
`public class NumericalDigitDataTest {`
 	
`    public static void main(String[] args) {`
 	
`        double doubleVal1  = 9999999999999999.0D;`
`        double doubleVal2  = 9999999999999999.4D;`
`        double doubleVal3  = 9999999999999999.5D;`
`        double doubleVal4  = 9999999999999999.9D;`
 	
`        String str1        = String.format("%15.0f", doubleVal1);`
`        String str2        = String.format("%15.0f", doubleVal2);`
`        String str3        = String.format("%15.0f", doubleVal3);`
`        String str4        = String.format("%15.0f", doubleVal4);`
 	
`        System.out.printf("str1 [%s]\n", str1);`
`        System.out.printf("str2 [%s]\n", str2);`
`        System.out.printf("str3 [%s]\n", str3);`
`        System.out.printf("str4 [%s]\n", str4);`
`    }`
`}`
	`str1 [9999999999999999]`
	`str2 [9999999999999999]`
	`str3 [10000000000000000]`
	`str4 [10000000000000000]`

str3,str4の結果もstr1,str2と同様にしたいなら整数部分だけを明示的に与える必要があります。

コード	結果

NumericalDigitData4Test.java
`public class NumericalDigitDataTest {`
 	
`    public static void main(String[] args) {`
 	
`        double doubleVal1  = 999999999999999.0D;`
`        double doubleVal2  = 999999999999999.4D;`
`        double doubleVal3  = 999999999999999.5D;`
`        double doubleVal4  = 999999999999999.9D;`
 	
`        String str1        = String.format("%15.0f", Math.floor(doubleVal1));`
`        String str2        = String.format("%15.0f", Math.floor(doubleVal2));`
`        String str3        = String.format("%15.0f", Math.floor(doubleVal3));`
`        String str4        = String.format("%15.0f", Math.floor(doubleVal4));`
 	
`        System.out.printf("str1 [%s]\n", str1);`
`        System.out.printf("str2 [%s]\n", str2);`
`        System.out.printf("str3 [%s]\n", str3);`
`        System.out.printf("str4 [%s]\n", str4);`
`    }`
`}`
	`str1 [999999999999999]`
	`str2 [999999999999999]`
	`str3 [999999999999999]`
	`str4 [999999999999999]`

桁数が16桁なのでは？

入力の桁数が多いんじゃないの？というコメントが来たので確かめよう。

コード	結果

NumericalDigitData5Test.java
`public class NumericalDigitDataTest {`
 	
`    public static void main(String[] args) {`
 	
`        double doubleVal = 9999999999999999D; // 9を16桁`
`        System.out.printf("[%15.0f]\n", doubleVal);`
`    }`
`}`
	`[10000000000000000]`

NumericalDigitData6Test.java
`public class NumericalDigitDataTest {`
 	
`    public static void main(String[] args) {`
 	
`        double doubleVal = 9999999999999999D; // 9を16桁`
`        System.out.printf("[%17.0f]\n", doubleVal);  // %17.0fにして確認する`
`    }`
`}`
	`[10000000000000000]`

見ての通り、値を9,999,999,999,999,999にしても表示値は10,000,000,000,000,000になり桁数も17桁になっていておかしい。

どうせなので

- (A) 9,999,999,999,999,998
- (B) 9,999,999,999,999,998.9
- (C) 9,999,999,999,999,999
- (C)10,000,000,000,000,000

の内部も見てみる

値	符号	指数部	仮数部	正規化前の仮数部
(A)	0	10000110100	0001110000110111100100110111111000000111111111111111	1.0001110000110111100100110111111000000111111111111111
(B)	0	10000110100	0001110000110111100100110111111000000111111111111111	1.0001110000110111100100110111111000000111111111111111
(C)	0	10000110100	0001110000110111100100110111111000001000000000000000	1.0001110000110111100100110111111000001000000000000000
(D)	0	10000110100	0001110000110111100100110111111000001000000000000000	1.0001110000110111100100110111111000001000000000000000

内部形式が重複してしまっていますね。
そして(A)(D)のいずれも指数部は 10000110100(2)=1076(10)で、1076 - 1023=53 ※仮数部52桁を1桁オーバーしている
仮数部の有効桁を使い切ってしまっているのでもう無理という事かな。

技術資料, double型, java