1 Byte (от -128 до 127 и занимает 1 байт)

```
public class check_Byte_0 {
byte b1 = 126; # -126, 127, -127
2
           5
  0
    1
       3
         4
             6
00000020 01100001 01101110 01100111 00101111 01001111 01100010 01101010 01100101 ang/Obje
00000048 00001010 01100011 01101000 01100101 01100011 01101011 01011111 01100010 .check b
00000080 01101001 01100001 01100010 01101100 01100101 01010100 01100001 01100010 iableTab
00000090 01110011 00000001 00000000 00001100 01001100 01100011 01101000 01100101 s...Lche
00000098 01100011 01101011 01011111 01100010 01111001 01110100 01100101 00111011 ck byte;
000000b0 01100011 01101000 01100101 01100011 01101011 01011111 01100010 01111001 check by
00000120 00000000 00000010 00000000 00010011
         10000000 -128
```

10000000 -128 10000001 -127 10000010 -126 01111110 126 01111111 127

2 Short (от -32768 до 32767 и занимает 2 байта)

```
public class check_short {
  short sh1 = 32767; // 32767, -32768, -32767

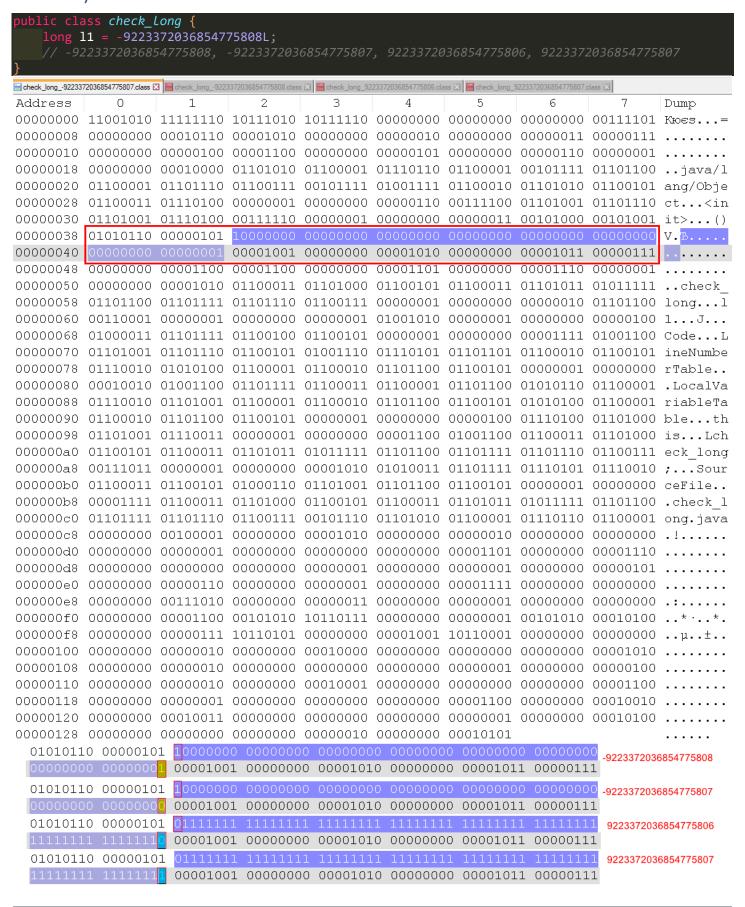
    check_short_32767.class 
    elseck_short_-32768.class 
    elseck_short_-32767.class 
    elseck_short_-32767.class 

             1
        0
                      2
                             3
                                    4
                                          5
                                                 6
00000020 01100001 01101110 01100111 00101111 01001111 01100010 01101010 01100101 ang/Obje
00000048 00001011 01100011 01101000 01100101 01100011 01101011 01011111 01110011 .check s
000000b0 00000001 00000000 00010000 01100011 01101000 01100101 01100011 01101011 ...check
000000b8 01011111 01110011 01101000 01101111 01110010 01110100 00101110 01101010 short.j
00000128 00010011
                          01111111 11111111 32767
                          10000000 00000000 -32768
                          10000000 00000001 -32767
```

```
public class check_int {
    int i1 = 2147483647; // -2147483648 -2147483647 -2147483646 2147483646 2147483647
}
```

	3648.class 🖾 🔚 checl	k int -2147483647.class	s 🗵 🔚 check int -214	47483646.class 🔀 🔠	check int 2147483646.	class 🔀 🔚 check int	2147483647.class 🔀	Check int 21474836	646.class 🗷 📙 check int
Address	0	1	2	3	4	5	6	7	Dump
00000000	11001010	11111110	10111010	10111110	00000000	00000000	00000000	00111101	Кюєз=
00000008	00000000	00010101	00001010	00000000	00000010	00000000	00000011	00000111	
00000010	00000000	00000100	00001100	00000000	00000101	00000000	00000110	00000001	
00000018	00000000	00010000	01101010	01100001	01110110	01100001	00101111	01101100	java/l
00000020	01100001	01101110	01100111	00101111	01001111	01100010	01101010	01100101	ang/Obje
00000028	01100011	01110100	00000001	00000000	00000110	00111100	01101001	01101110	ct <in< td=""></in<>
00000030	01101001	01110100	00111110	00000001	00000000	00000011	00101000	00101001	it>()
00000038	01010110	00000011	10000000	00000000	00000000	00000000	00001001	00000000	V. B
00000040	00001001	00000000	00001010	00000111	00000000	00001011	00001100	00000000	
00000048	00001100	00000000	00001101	00000001	00000000	00001001	01100011	01101000	ch
00000050	01100101	01100011	01101011	01011111	01101001	01101110	01110100	00000001	eck_int.
00000058	00000000	00000010	01101001	00110001	00000001	00000000	00000001	01001001	i1I
00000060	00000001	00000000	00000100	01000011	01101111	01100100	01100101	00000001	Code.
00000068	00000000	00001111	01001100	01101001	01101110	01100101			
00000070	01101101	01100010	01100101	01110010	01010100	01100001	01100010	01101100	mberTabl
00000078	01100101	00000001	00000000	00010010	01001100	01101111	01100011	01100001	eLoca
00000080	01101100	01010110	01100001	01110010	01101001	01100001	01100010	01101100	lVariabl
00000088	01100101	01010100	01100001	01100010	01101100	01100101	00000001	00000000	eTable
00000090		01110100							
00000098		01100011							_
000000a0		01110100							•
000000a8		01110010							
000000000		00000000							
8d000000		01101001							
00000000		00000000							
000000c8		00000000			00000000				
000000000		00000000		00000000		00000000			
000000d8		00000000						00000000	
000000e0		00000000							
000000e8		00000000							
000000f0		00000111							
000000f8		00000010							
00000100		00000010							
		00000010							
		00000001							
		00010010					00000000	00010011	
		00000000				00010100			
10000000 00000000 00000000 00000000 -2147483648 10000000 00000000 00000000 00000001 -2147483647 10000000 00000000 00000000 00000010 -2147483646 01111111 11111111 11111111 11111110 2147483646 01111111 11111111 11111111 11111111 2147483647									

4 Long (от –9223372036854775808 до 9223372036854775807 и занимает 8 байт)



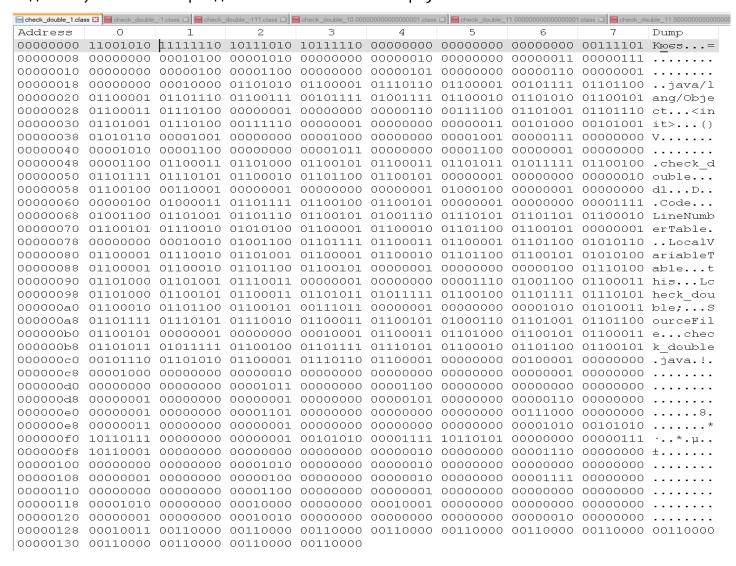
5 Float (число с плавающей точкой от -224*10¹⁰⁴ до 224*10¹⁰⁴ и занимает 4 байта). Максимум — 38 цифр в значении

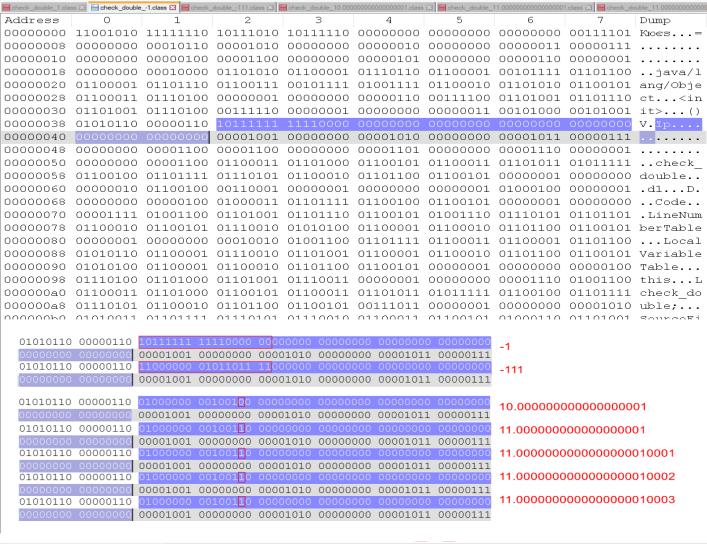


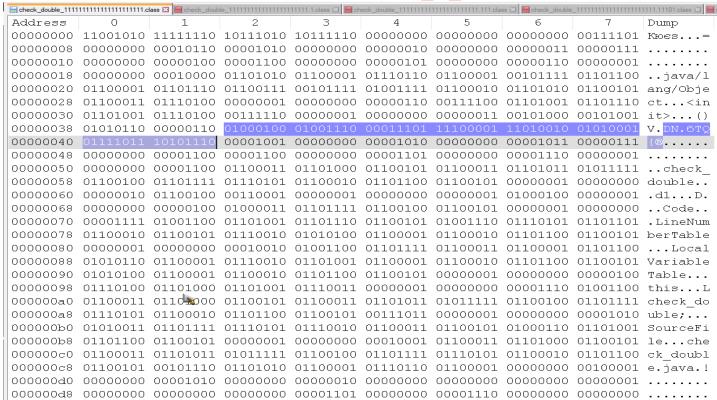


6 Double (число с плавающей точкой от ±4.9*10-324 до ±1.7976931348623157*10308 и занимает 8 байт)

Для аргумента = 1 класс выглядит не похожим на все остальные, отличается существенно во многих местах, выделить какую-то одну группу байтов невозможно. Классы при других аргументах выглядят совсем по другому и между собой изменяются только в одном месте (выделено синим) либо вообще не меняются (это обозначено отдельно)вот так и при дальнейшем изменении аргумента.







Все абсолютно одинаково для: