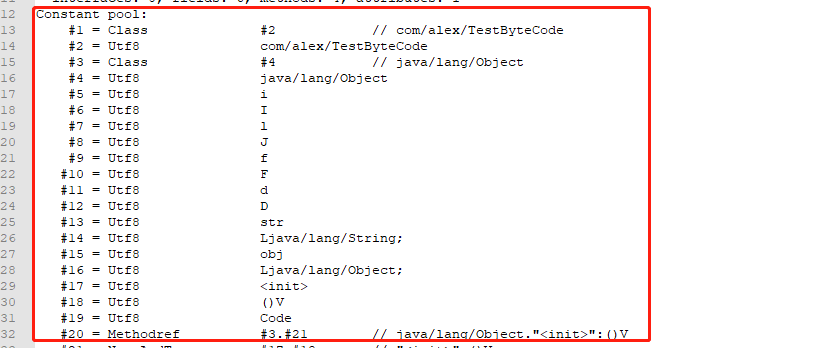
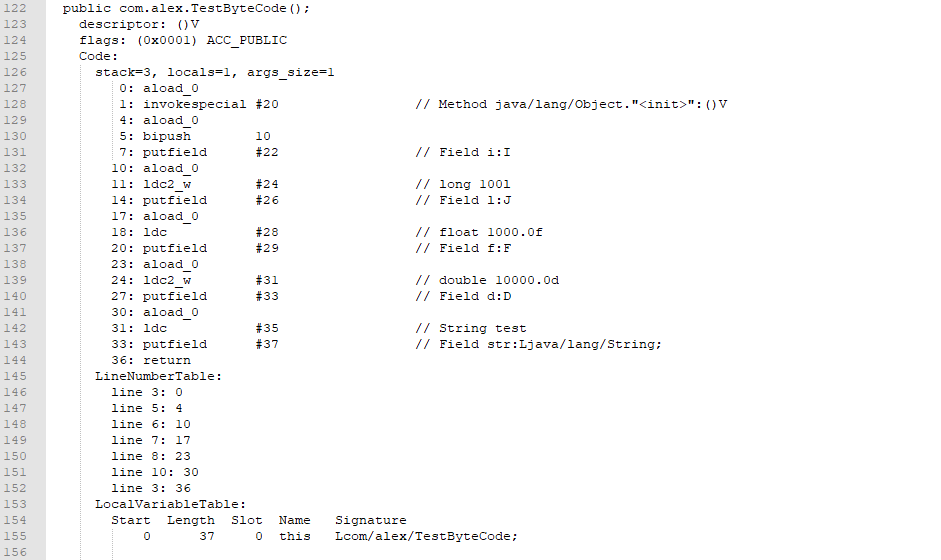


字节码修改时间，大小，版本号等基本信息；



常量池信息



TestByteCode类默认构造方法对应的字节码；

aload\_0：加载局部变量表中的第一个数据到操作数栈中；

非静态方法的局部变量表中的第一个数据是this（调用方法的对象）；

invokespecial #20：调用常量池#20位置对应的方法，即父类Object的构造方法；

#20位置的常量由其他的常量组成，如下所示：

#3 = Class #4 // java/lang/Object

#4 = Utf8 java/lang/Object

#17 = Utf8 <init>

#18 = Utf8 ()V

#20 = Methodref #3.#21 // java/lang/Object."<init>":()V

#21 = NameAndType #17:#18 // "<init>":()V

invokestatic：用于调用静态方法。

invokespecial：用于调用私有实例方法、构造器，以及使用 super 关键字调用父类的实例方法或构造器，和所实现接口的默认方法。

invokevirtual：用于调用非私有实例方法。

invokeinterface：用于调用接口方法。

invokedynamic：用于调用动态方法。

aload\_0：invokespecial #20指令的方法调用会消耗由第一条指令aload\_0加载到操作数栈中的this，如果后续指令操作需要用到this，需要重新aload\_0，从局部变量表中加载this到操作数栈中；

bipush 10：bipush加载一字节所代表的int常数到操作数栈中；

putfield #22：把bipush 10指令加载的10赋值到aload\_0加载的this对象的常量池#22对应的i字段上；

#22 = Fieldref #1.#23 // com/alex/TestByteCode.i:I

4: aload\_0

5: bipush 10

7: putfield #22 // Field i:I

对应的代码：

**private** **int** i = 10;

后续指令：

10: aload\_0

11: ldc2\_w #24 // long 100l

14: putfield #26 // Field l:J

17: aload\_0

18: ldc #28 // float 1000.0f

20: putfield #29 // Field f:F

23: aload\_0

24: ldc2\_w #31 // double 10000.0d

27: putfield #33 // Field d:D

30: aload\_0

31: ldc #35 // String test

33: putfield #37 // Field str:Ljava/lang/String;

对应的代码

**private** **long** l = 100;

**private** **float** f = 1000.00f;

**private** **double** d = 10000.00;

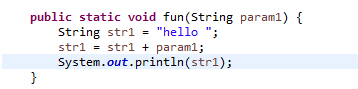
**private** String str = "test";

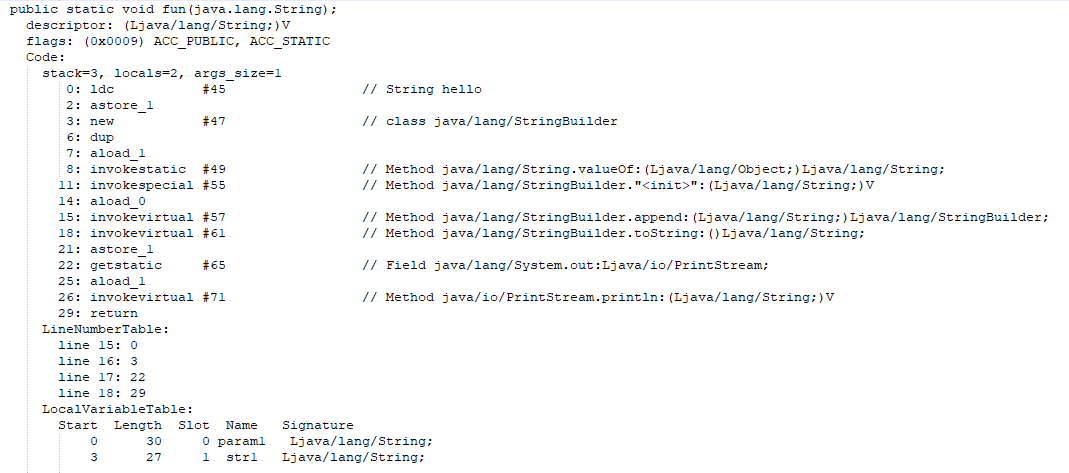
**private** Object obj;

ldc：加载常量池中的数据到操作数栈

ldc、ldc\_w、ldc2\_w，都是从运行时常池将目标对象压入操作数栈，前两个指令功能一致，主要针对int、char、float、String，区别是ldc\_w是宽索引，最后一个针对long、double







ldc #45：从常量池中加载hello到操作数栈中；

#45 = String #46 // hello

#46 = Utf8 hello

astore\_1：把hello存储到本地变量表中1的位置，赋值给str1；

new #47 // class java/lang/StringBuilder

new一个StringBuilder对象；

执行new指令，Java 虚拟机将指向一块已分配的、未初始化的内存的引用压入操作数栈中。

dup：new 指令所生成的未经初始化的引用，后续的操作会消耗对象引用，所以要先复制一份；

aload\_1：加载本地变量表中位置1的数据到操作数栈中，即str1；

invokestatic #49 // Method java/lang/String.valueOf:(Ljava/lang/Object;)Ljava/lang/String;

invokespecial #55 // Method java/lang/StringBuilder."<init>":(Ljava/lang/String;)V

调用str1的valueOf方法，获取str1中的值用来构建StringBuilder；

aload\_0：加载本地变量表中的位置0出的数据到操作数栈中，即param1参数；

invokevirtual #57 // Method java/lang/StringBuilder.append:(Ljava/lang/String;)Ljava/lang/StringBuilder;

invokevirtual #61 // Method java/lang/StringBuilder.toString:()Ljava/lang/String;

调用StringBuilder.append方法，把str1的内容和param1的内容加起来，调用toString方法，把StringBuilder转化成字符串

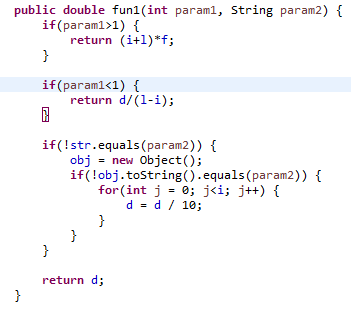
astore\_1：把str1和param1相加的字符串写入局部变量表的位置1处，即复制给str1；

getstatic #65 // Field java/lang/System.out:Ljava/io/PrintStream;

aload\_1

invokevirtual #71 // Method java/io/PrintStream.println:(Ljava/lang/String;)V

调用System.out静态字段，获取Printstream对象，加载局部变量表的位置1处的数据，即str1，调用Printstream的print方法打印str1；



加载：本地变量表->操作数栈

存储：操作数栈->本地变量表

0: iload\_1：加载本地变量表位置1的数据到操作数栈，即param1；

1: iconst\_1：加载常量1到操作数栈

2: if\_icmple 23：比较param1和1的大小，如果param不大于1，跳转到指令23处；

5: aload\_0：加载this到操作数栈

6: getfield #22 // Field i:I

获取this.i;

9: i2l：int转long

10: aload\_0：加载this到操作数栈

11: getfield #26 // Field l:J

获取this.l;

14: ladd

long型加法，i+l；

15: l2f

long转float

16: aload\_0：加载this到操作数栈

17: getfield #29 // Field f:F

加载this.f

20: fmul

float乘法

21: f2d

float转double，方法的返回值是double；

22: dreturn：返回double数据；

23: iload\_1：加载param1；

24: iconst\_1：加载常量1；

25: if\_icmpge 45：比较param1和1的大小，param1不小于1就跳转到指令45处

28: aload\_0：加载this；

29: getfield #33 // Field d:D

加载this.d

32: aload\_0：加载this

33: getfield #26 // Field l:J

加载this.l

36: aload\_0：加载this

37: getfield #22 // Field i:I

加载this.i

40: i2l： int转long

41: lsub： long减法

42: l2d： long转double

43: ddiv： double除法

44: dreturn：返回double类型数据

45: aload\_0

46: getfield #37 // Field str:Ljava/lang/String;

加载this

加载this.str

49: aload\_2

加载param2

50: invokevirtual #80 // Method java/lang/String.equals:(Ljava/lang/Object;)Z

53: ifne 109

调用equals方法比较str和param2，如果不相等，就跳到指令109处；

56: aload\_0：加载this

57: new #3 // class java/lang/Object

分配内存，返回一块为初始化的内存引用到操作数栈

60: dup：复制上述引用；

61: invokespecial #20 // Method java/lang/Object."<init>":()V

调用Object的构造方法；

64: putfield #84 // Field obj:Ljava/lang/Object;

把dup赋值的引用赋值给this.obj

67: aload\_0

68: getfield #84 // Field obj:Ljava/lang/Object;

71: invokevirtual #86 // Method java/lang/Object.toString:()Ljava/lang/String;

加载this

加载this.obj

调用this.obj.toString方法

74: aload\_2

75: invokevirtual #80 // Method java/lang/String.equals:(Ljava/lang/Object;)Z

78: ifne 109

加载param2，调用equals方法比较obj.toString()和param2内容，如果不相等就跳转到指令109处；

81: iconst\_0：加载常量0

82: istore\_3：把0存储到本地变量表3处，即复制给j

83: goto 101：跳转到指令101处

86: aload\_0：加载this

87: dup：复制this

88: getfield #33 // Field d:D

加载this.d

91: ldc2\_w #87 // double 10.0d

加载常量10，double和float双精度所以用ldc2\_w

94: ddiv：double除法

95: putfield #33 // Field d:D

赋值d/10的结果给d

98: iinc 3, 1

本地变量表3处加1，**iinc是直接操作的本地变量表**，不是操作的操作数栈

101: iload\_3：加载j

102: aload\_0：加载this

103: getfield #22 // Field i:I

加载this.i

106: if\_icmplt 86

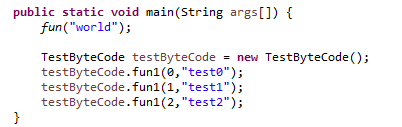
比较i和j，j小于i跳转到指令86处；

109: aload\_0：加载this

110: getfield #33 // Field d:D

加载this.d

113: dreturn：返回double类型数据



0: ldc #94 // String world

加载常量world

2: invokestatic #96 // Method fun:(Ljava/lang/String;)V

调用静态方法fun

5: new #1 // class com/alex/TestByteCode

分配为初始化的内存，并返回内存引用

8: dup

复制上述内存引用

9: invokespecial #98 // Method "<init>":()V

调用构造函数

12: astore\_1

构造函数返回对象存储到testByteCode

13: aload\_1

加载testByteCode

14: iconst\_0

加载常量0

15: ldc #99 // String test0

加载常量test0

17: invokevirtual #101 // Method fun1:(ILjava/lang/String;)D

调用testByteCode.fun1，操作数为0和test0

20: pop2

方法出栈；

21: aload\_1

22: iconst\_1

23: ldc #103 // String test1

25: invokevirtual #101 // Method fun1:(ILjava/lang/String;)D

28: pop2

加载testByteCode，常量1，test1；

调用testByteCode.fun1，操作数为1和test1；

方法出栈；

29: aload\_1

30: iconst\_2

31: ldc #105 // String test2

33: invokevirtual #101 // Method fun1:(ILjava/lang/String;)D

36: pop2

加载testByteCode，常量2，test2；

调用testByteCode.fun1，操作数为2和test2；

方法出栈；

37: return：返回