Лекция

Исполняемые файлы java

Темы

- Структура class файла.
- Средства работы с class файлами.
- Байт-код java.
- Стеганография и цифровые водяные знаки.
- Способы скрытого вложения информации class файлы.

Class файл

- Содержит байт-код, который выполняется на виртуальной машине.
- Содержит информацию о классе.
- Генерируется компилятором из исходного кода (.java)
- Jar файл – представляет собой zip архив class файлов.

Структура class файла

class файл состоит из 1,2,4 байтовых значений.

Введем обозначения:
u1 – byte (1 байт)
u2 – long (2 байта)
u4 – int (4 байта)

Структура class файла

```
ClassFile {
    u4             magic;
    u2             minor_version;
    u2             major_version;
    u2             constant_pool_count;
    cp_info        constant_pool[constant_pool_count-1];
    u2             access_flags;
    u2             this_class;
    u2             super_class;
    u2             interfaces_count;
    u2             interfaces[interfaces_count];
    u2             fields_count;
    field_info     fields[fields_count];
    u2             methods_count;
    method_info    methods[methods_count];
    u2             attributes_count;
    attribute_info attributes[attributes_count];
}
```

Структура class файла

u4 magic

Идентификатор формата файла класса.

Позволяет идентифицировать .class файл.

Всегда принимает значение: 0xCAFEBABE.

Структура class файла

u2 minor_version

u2 major_version

Вспомогательная и основная версии class файла.

Структура class файла

u2 constant_pool_count

Размер массива констант.

Структура class файла

cp_info constant_pool[constant_pool_count-1]

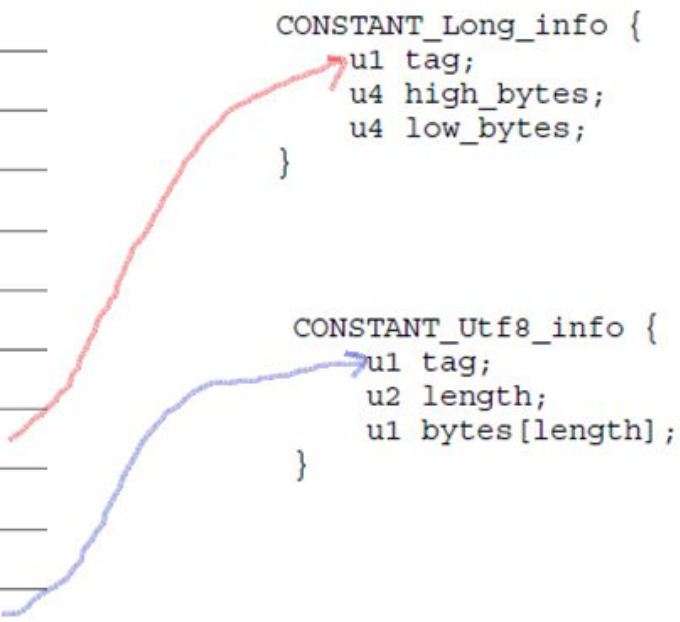
Пул констант представлен в виде массива структур в которых содержится:

строковые константы, имена классов, интерфейсов и полей …

```
cp_info {
    u1 tag;
    u1 info[];
}
```

Формат каждого элемента пула констант определяется первым байтом (**tag**).

Constant Type	Value
CONSTANT_Class	7
CONSTANT_Fieldref	9
CONSTANT_Methodref	10
CONSTANT_InterfaceMethodref	11
CONSTANT_String	8
CONSTANT_Integer	3
CONSTANT_Float	4
CONSTANT_Long	5
CONSTANT_Double	6
CONSTANT_NameAndType	12
CONSTANT_Utf8	1
CONSTANT_MethodHandle	15
CONSTANT_MethodType	16
CONSTANT_InvokeDynamic	18



Структура class файла

u2 access_flags

Значение элемента является маской флагов, используемых таким образом, чтобы обозначить права доступа и свойства этого класса.

Флаги доступа

Flag Name	Value	Interpretation
ACC_PUBLIC	0x0001	Declared public; may be accessed from outside its package.
ACC_FINAL	0x0010	Declared final; no subclasses allowed.
ACC_SUPER	0x0020	Treat superclass methods specially when invoked by the *invokespecial* instruction.
ACC_INTERFACE	0x0200	Is an interface, not a class.
ACC_ABSTRACT	0x0400	Declared abstract; must not be instantiated.
ACC_SYNTHETIC	0x1000	Declared synthetic; not present in the source code.
ACC_ANNOTATION	0x2000	Declared as an annotation type.
ACC_ENUM	0x4000	Declared as an enum type.

Структура class файла

u2 this_class

u2 super_class

Ссылки на константу с названием класса и его суперкласса.

Структура class файла

u2 interfaces_count

Размер массива интерфейсов.

Структура class файла

u2 interfaces[interfaces_count]

Массив интерфейсов. Каждый элемент массива является индексом таблицы пула констант, где указывается имя интерфейса.

Структура class файла

u2 fields_count

Размер массива полей.

Структура class файла

field_info fields[fields_count]

Массив полей.

```
field_info {
    u2             access_flags;
    u2             name_index;
    u2             descriptor_index;
    u2             attributes_count;
    attribute_info attributes[attributes_count];
}
```

Структура class файла

u2 methods_count

Размер массива методов.

Структура class файла

method_info methods[methods_count]

Массив методов.

```
method_info {
    u2             access_flags;
    u2             name_index;
    u2             descriptor_index;
    u2             attributes_count;
    attribute_info attributes[attributes_count];
}
```

```
attribute_info {
    u2 attribute_name_index;
    u4 attribute_length;
    u1 info[attribute_length];
}
```

Attribute
ConstantValue
Code
StackMapTable
Exceptions
InnerClasses
EnclosingMethod
Synthetic
Signature
SourceFile
SourceDebugExtension
LineNumberTable
LocalVariableTable
LocalVariableTypeTable
Deprecated
RuntimeVisibleAnnotations
RuntimeInvisibleAnnotations
RuntimeVisibleParameterAnnotations
RuntimeInvisibleParameterAnnotations
AnnotationDefault
BootstrapMethods

Атрибут Code

```
Code_attribute {
    u2 attribute_name_index;
    u4 attribute_length;
    u2 max_stack;
    u2 max_locals;
    u4 code_length;
    u1 code[code_length];
    u2 exception_table_length;
    {   u2 start_pc;
        u2 end_pc;
        u2 handler_pc;
        u2 catch_type;
    } exception_table[exception_table_length];
    u2 attributes_count;
    attribute_info attributes[attributes_count];
}
```

Структура class файла

u2 attributes_count

Размер массива атрибутов.

Структура class файла

attribute_info attributes[attributes_count]

Массив атрибутов.

```
attribute_info {
    u2 attribute_name_index;
    u4 attribute_length;
    u1 info[attribute_length];
}
```

Структура class файла

```
ClassFile {
    u4             magic;
    u2             minor_version;
    u2             major_version;
    u2             constant_pool_count;
    cp_info        constant_pool[constant_pool_count-1];
    u2             access_flags;
    u2             this_class;
    u2             super_class;
    u2             interfaces_count;
    u2             interfaces[interfaces_count];
    u2             fields_count;
    field_info     fields[fields_count];
    u2             methods_count;
    method_info    methods[methods_count];
    u2             attributes_count;
    attribute_info attributes[attributes_count];
}
```

Средства работы с class-файлами

JBE - Java Bytecode Editor – программа, позволяющая просматривать и редактировать class файлы.

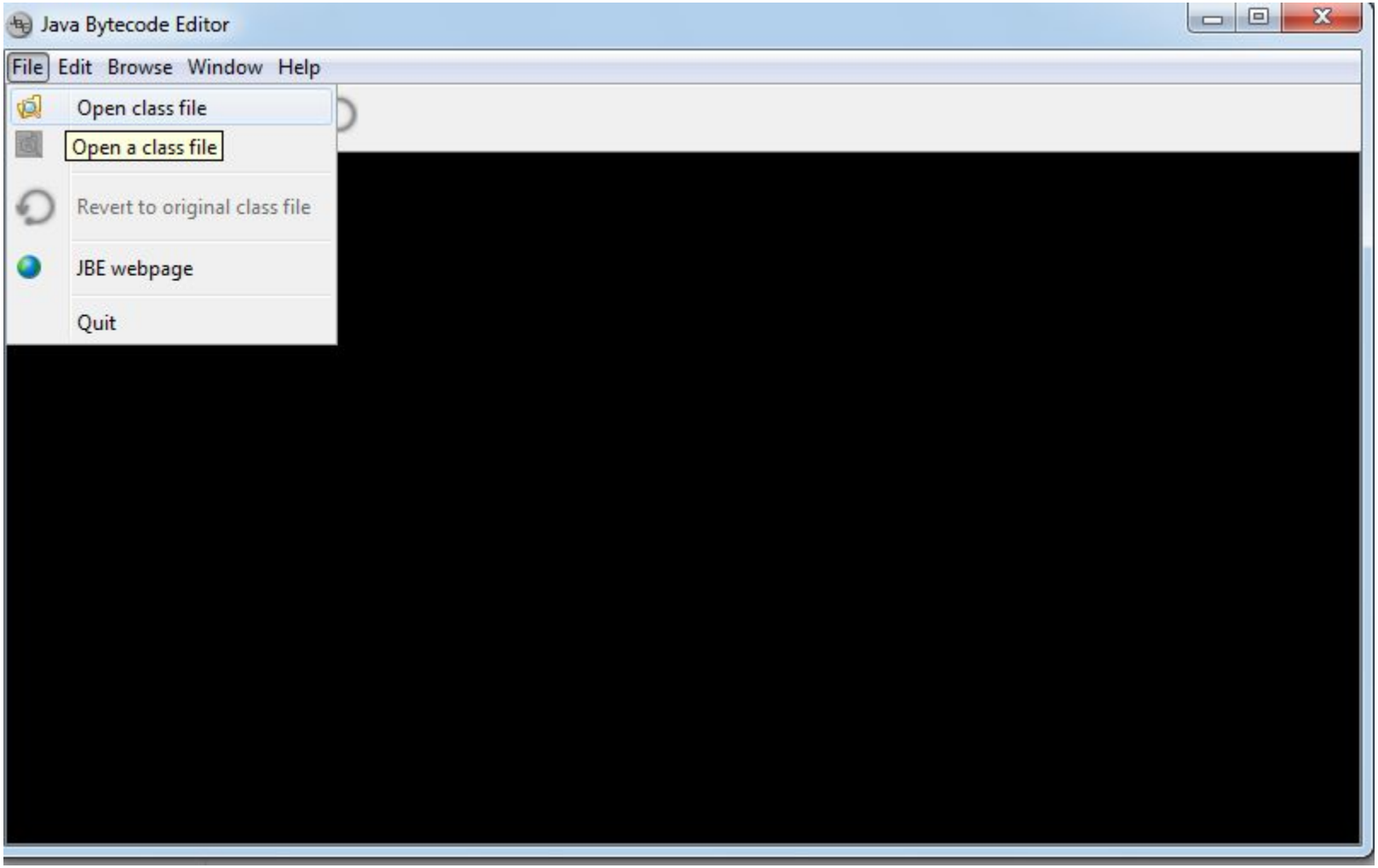
javap – декомпилятор class файлов.

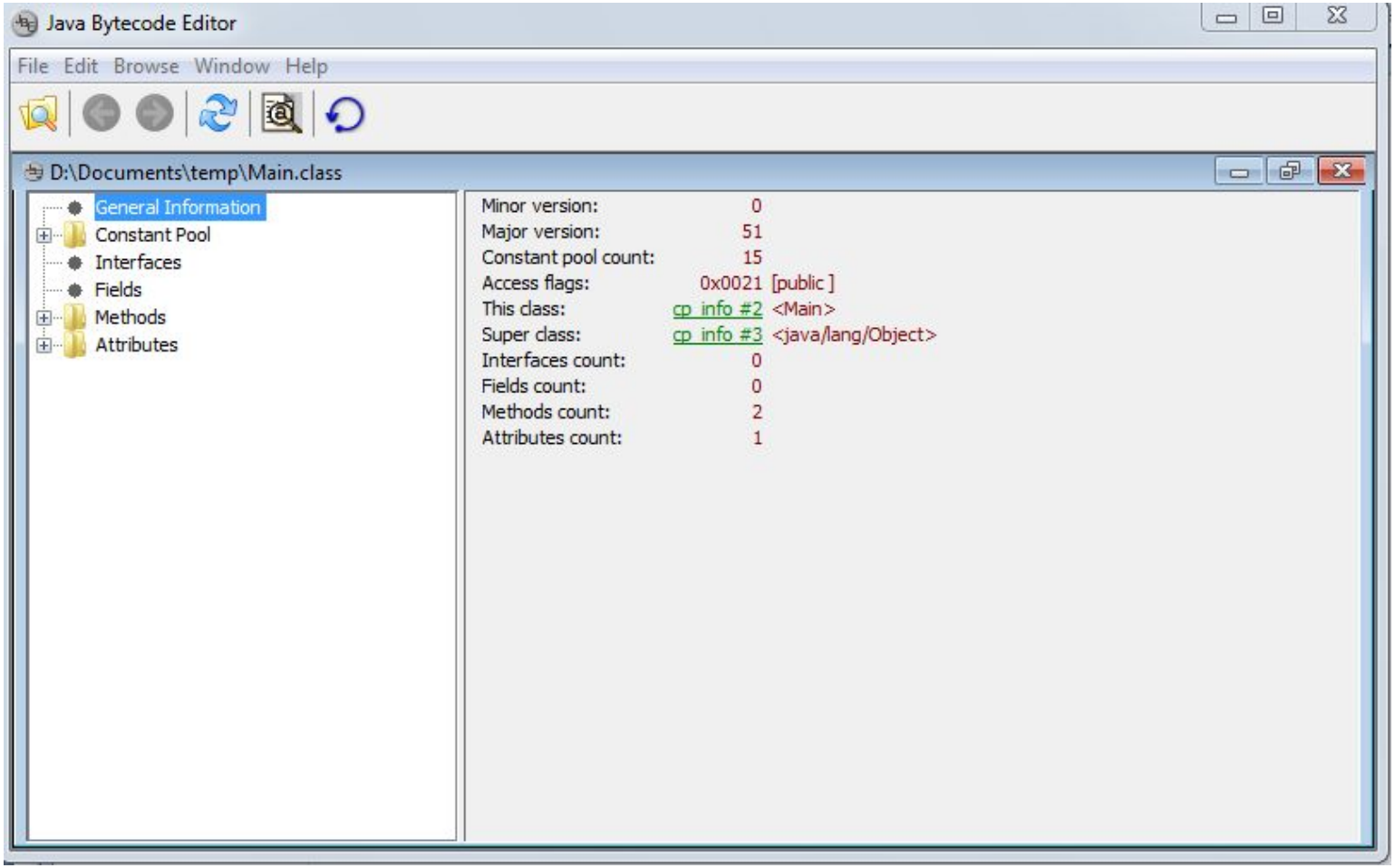
Исходный код

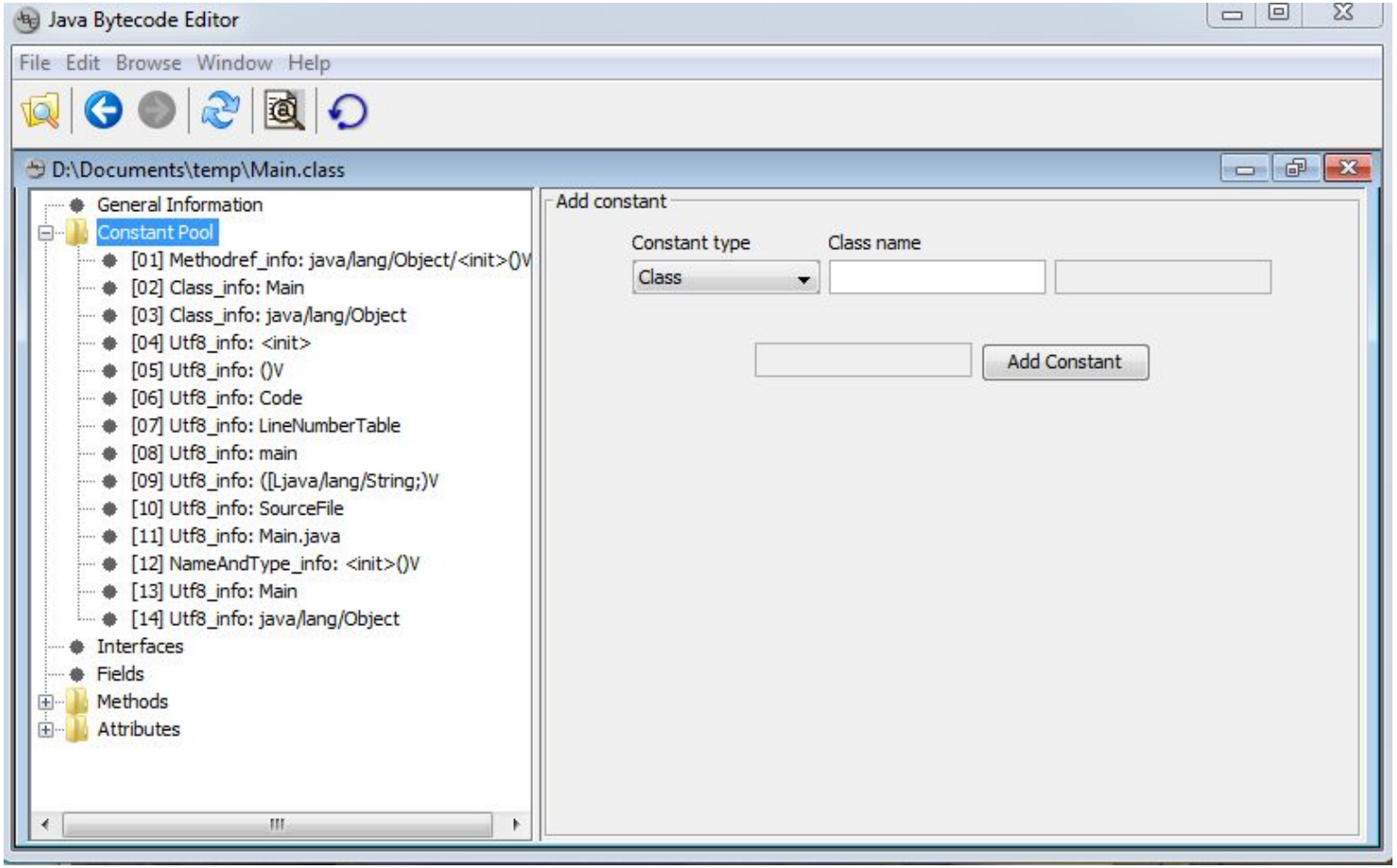
Файл Main.java

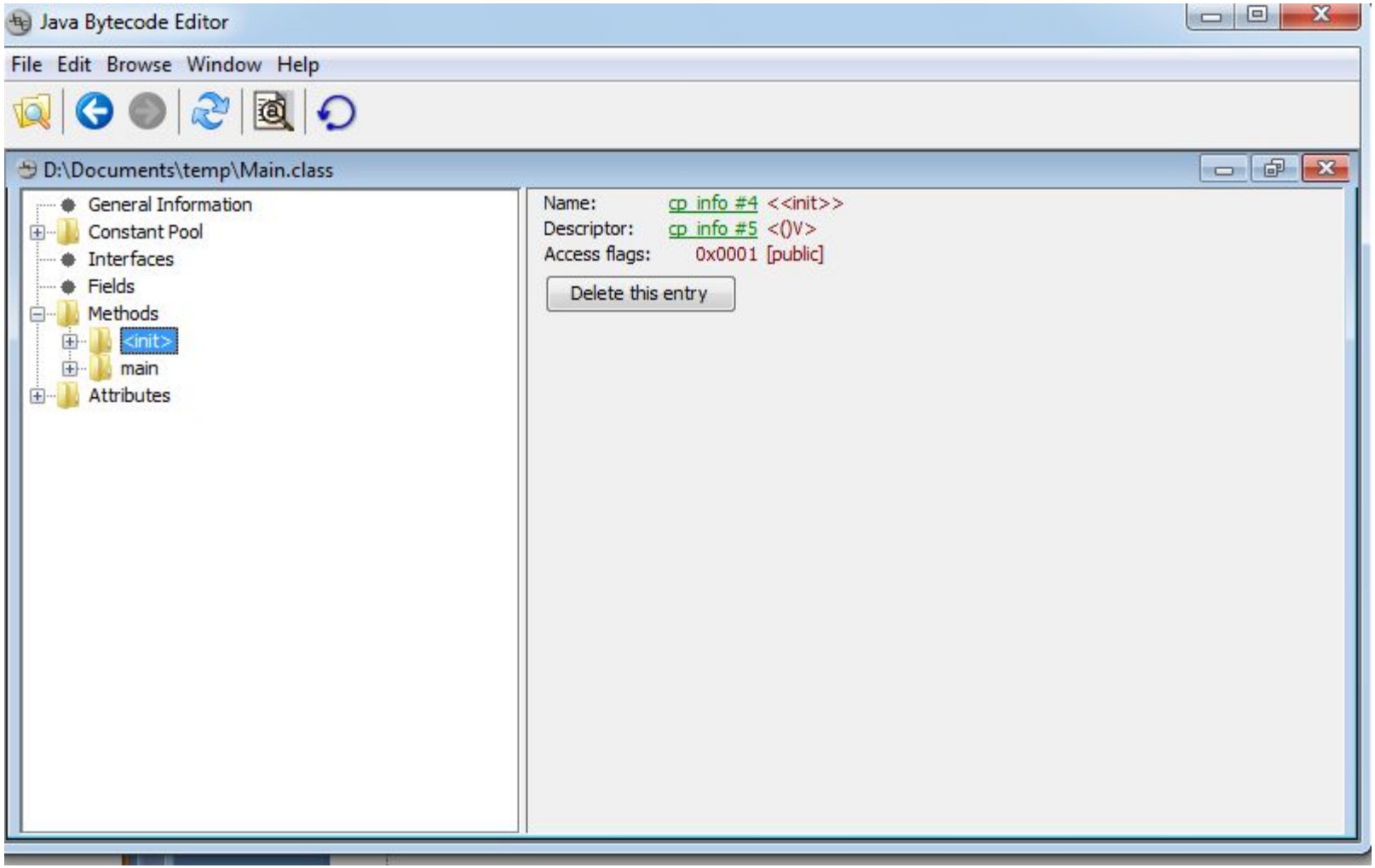
```
public class Main
{
    public static void main (String [] args )
    {
          int a = 4;
     int b = 5;
     int c = a+b;
    }
}
```

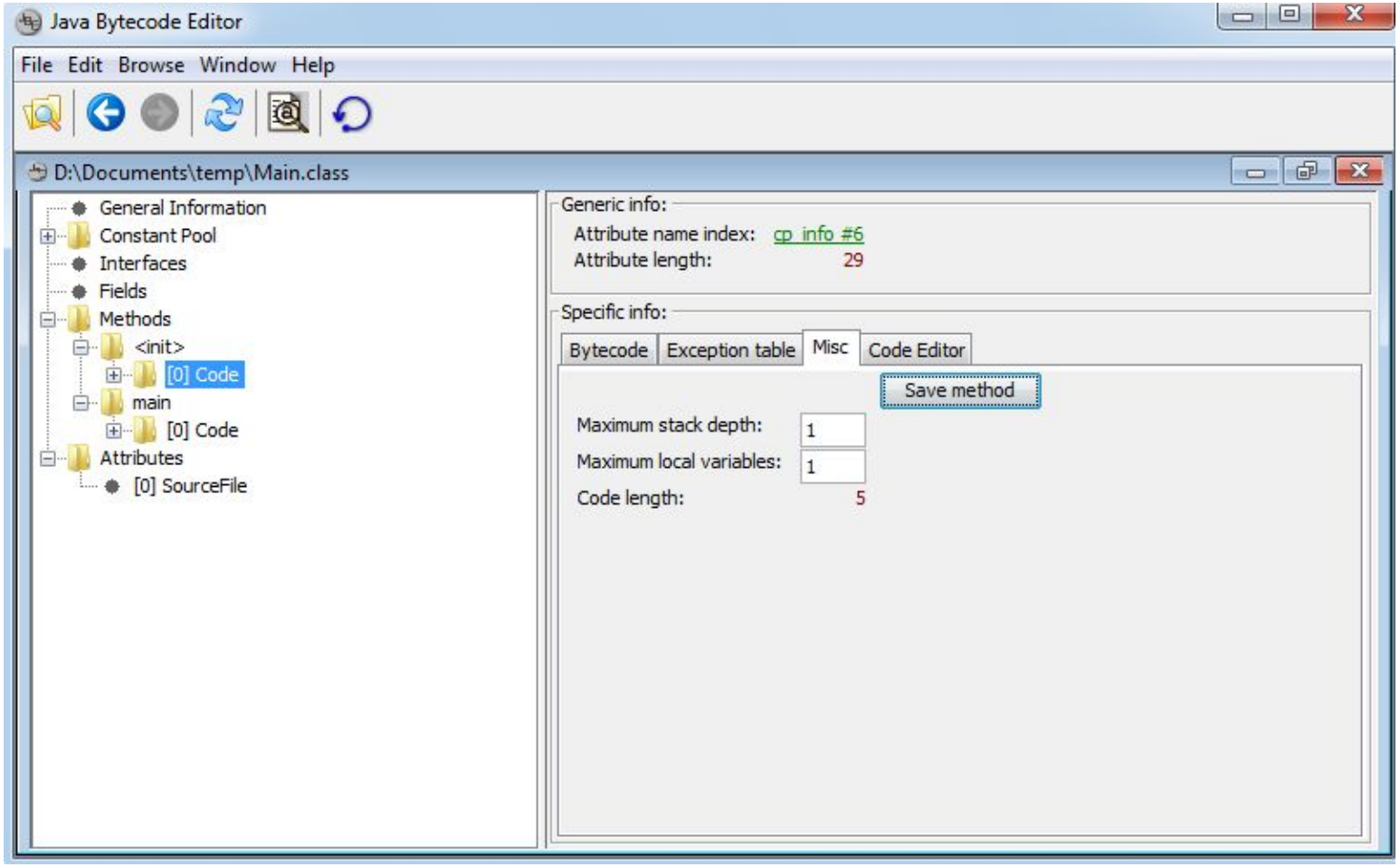
> **javac Main.java**

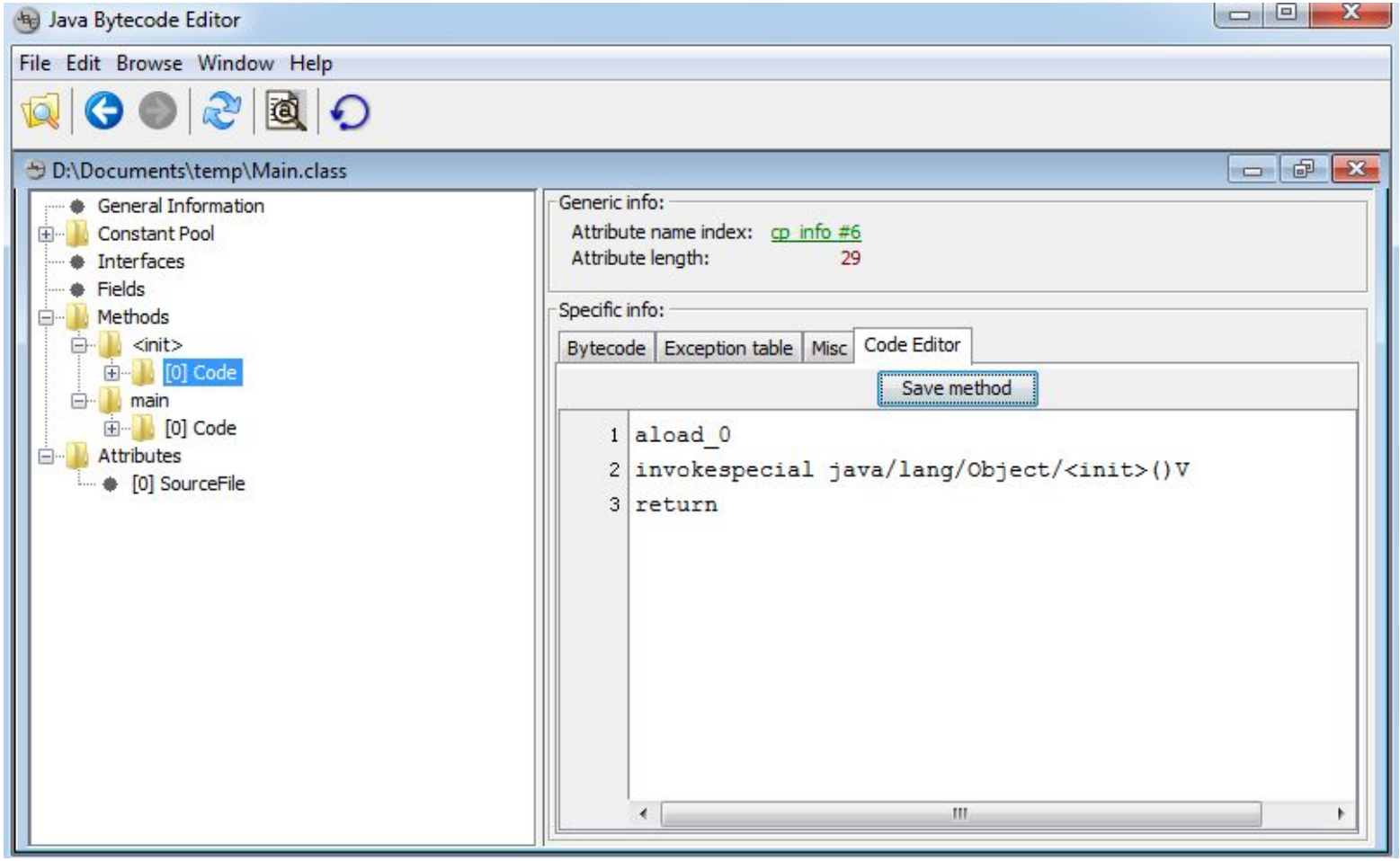

Java Bytecode Editor
File
Edit
Browse
Window
Help
Open class file
Open a class file
Revert to original class file
JBE webpage
Quit

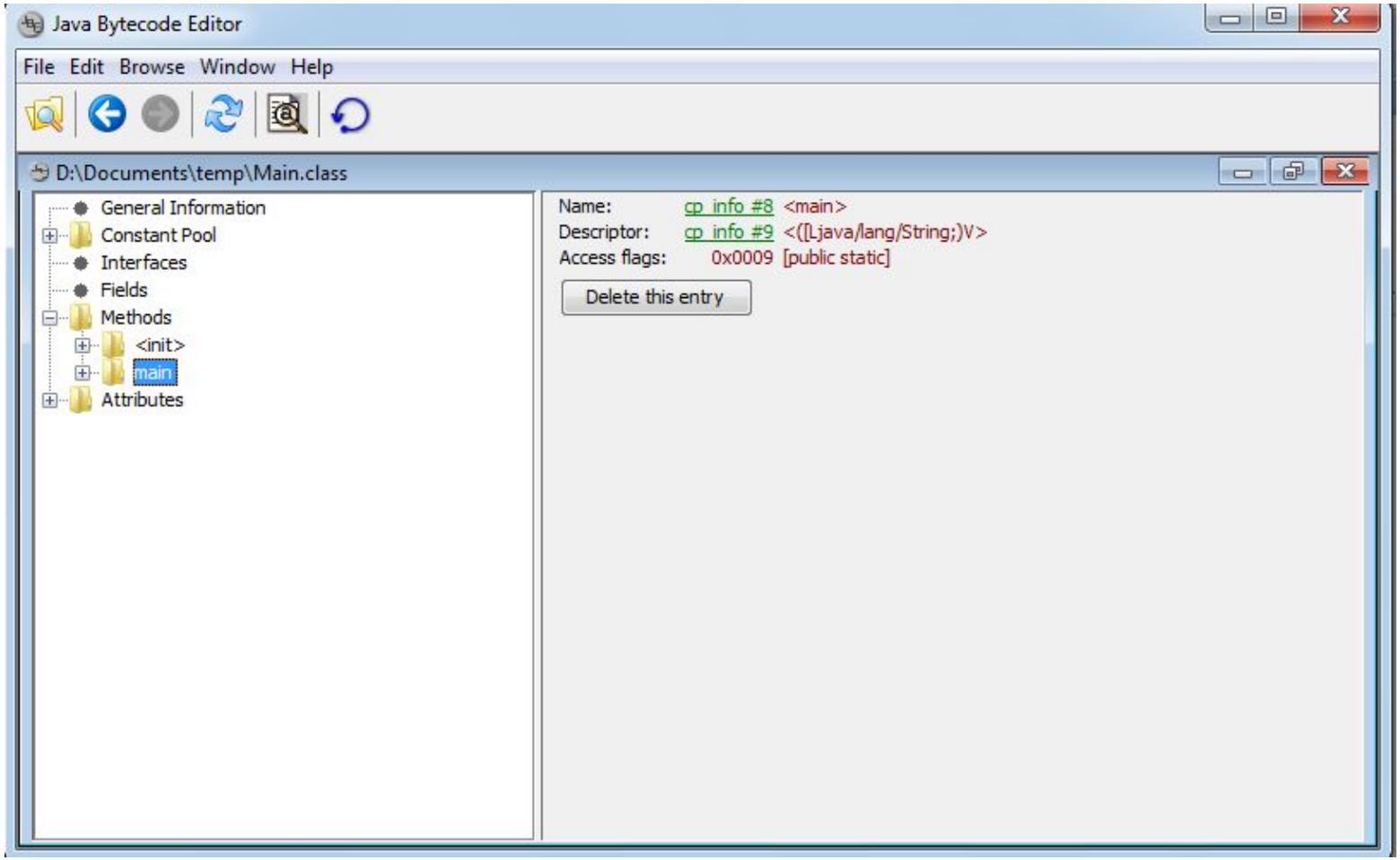

Java Bytecode Editor
File Edit Browse Window Help
D:\Documents\temp\Main.class
General Information
Constant Pool
Interfaces
Fields
Methods
Attributes
Minor version: 0
Major version: 51
Constant pool count: 15
Access flags: 0x0021 [public]
This class: cp_info #2 <Main>
Super class: cp_info #3 <java/lang/Object>
Interfaces count: 0
Fields count: 0
Methods count: 2
Attributes count: 1

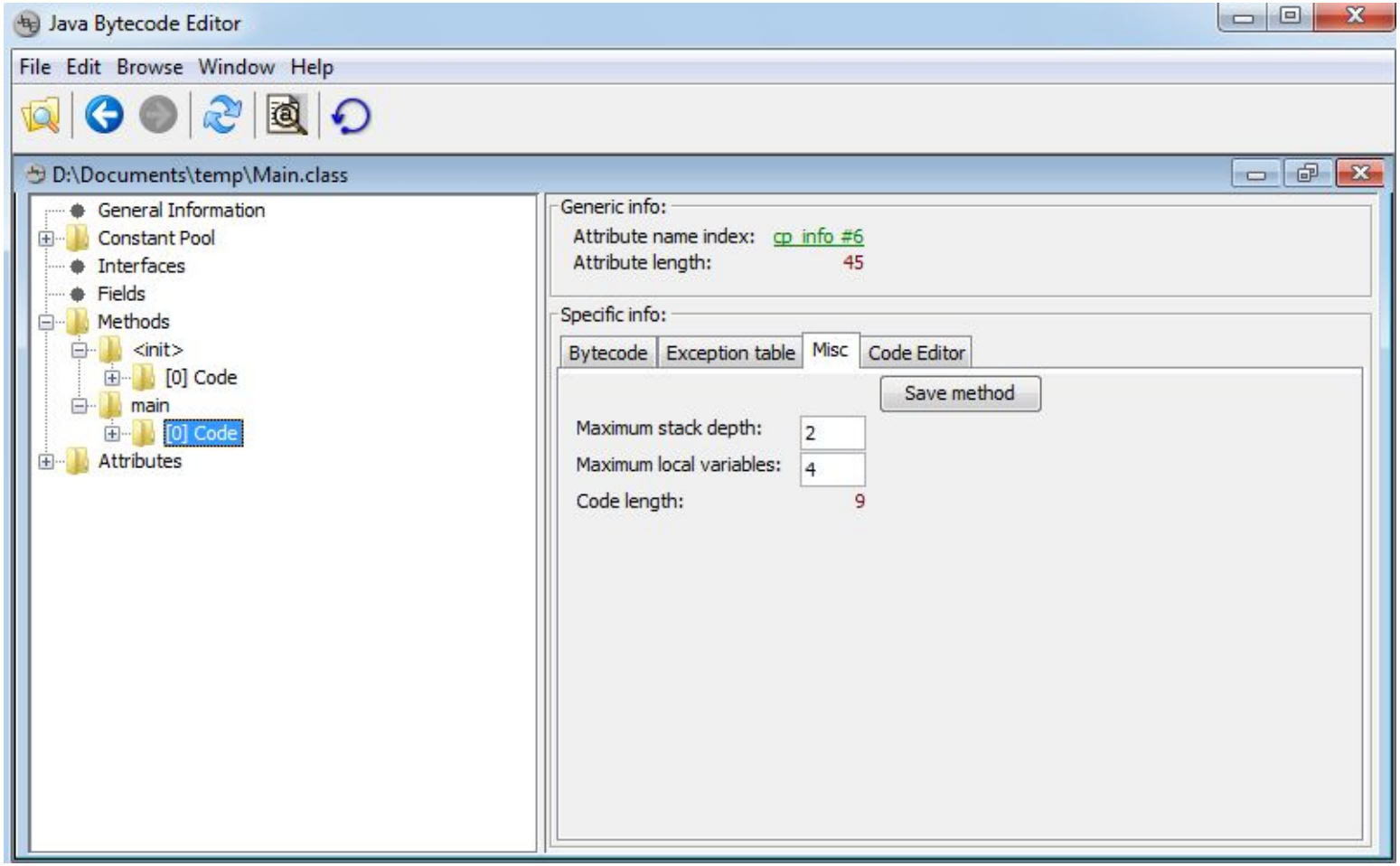

Java Bytecode Editor
File Edit Browse Window Help
D:\Documents\temp\Main.class
General Information
Constant Pool
[01] Methodref_info: java/lang/Object/<init>()V
[02] Class_info: Main
[03] Class_info: java/lang/Object
[04] Utf8_info: <init>
[05] Utf8_info: ()V
[06] Utf8_info: Code
[07] Utf8_info: LineNumberTable
[08] Utf8_info: main
[09] Utf8_info: ([Ljava/lang/String;)V
[10] Utf8_info: SourceFile
[11] Utf8_info: Main.java
[12] NameAndType_info: <init>()V
[13] Utf8_info: Main
[14] Utf8_info: java/lang/Object
Interfaces
Fields
Methods
Attributes
Add constant
Constant type
Class name
Class
Add Constant

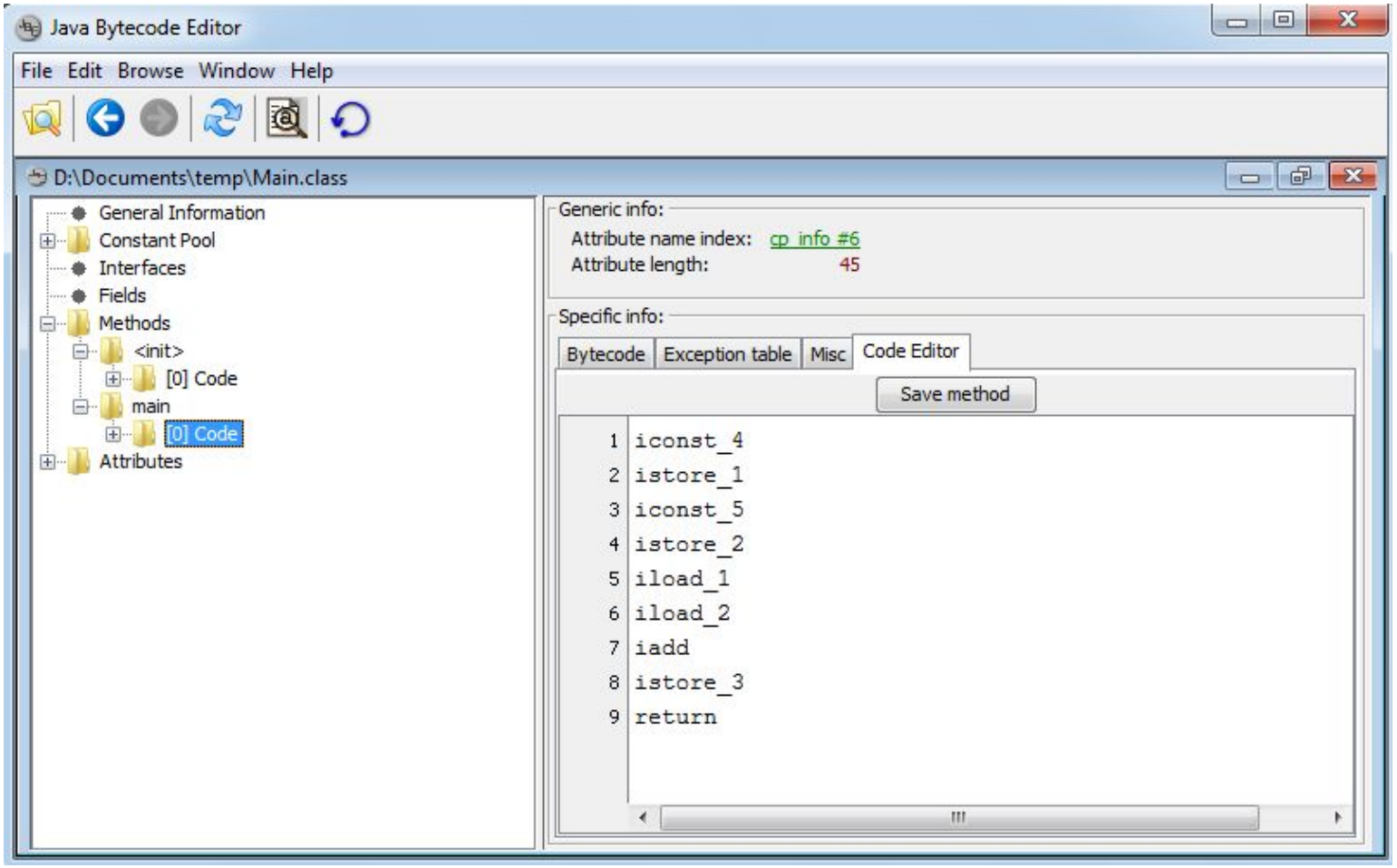

Java Bytecode Editor
File Edit Browse Window Help
D:\Documents\temp\Main.class
General Information
Constant Pool
Interfaces
Fields
Methods
<init>
main
Attributes
Name: cp_info #4 <<init>>
Descriptor: cp_info #5 <()V>
Access flags: 0x0001 [public]
Delete this entry

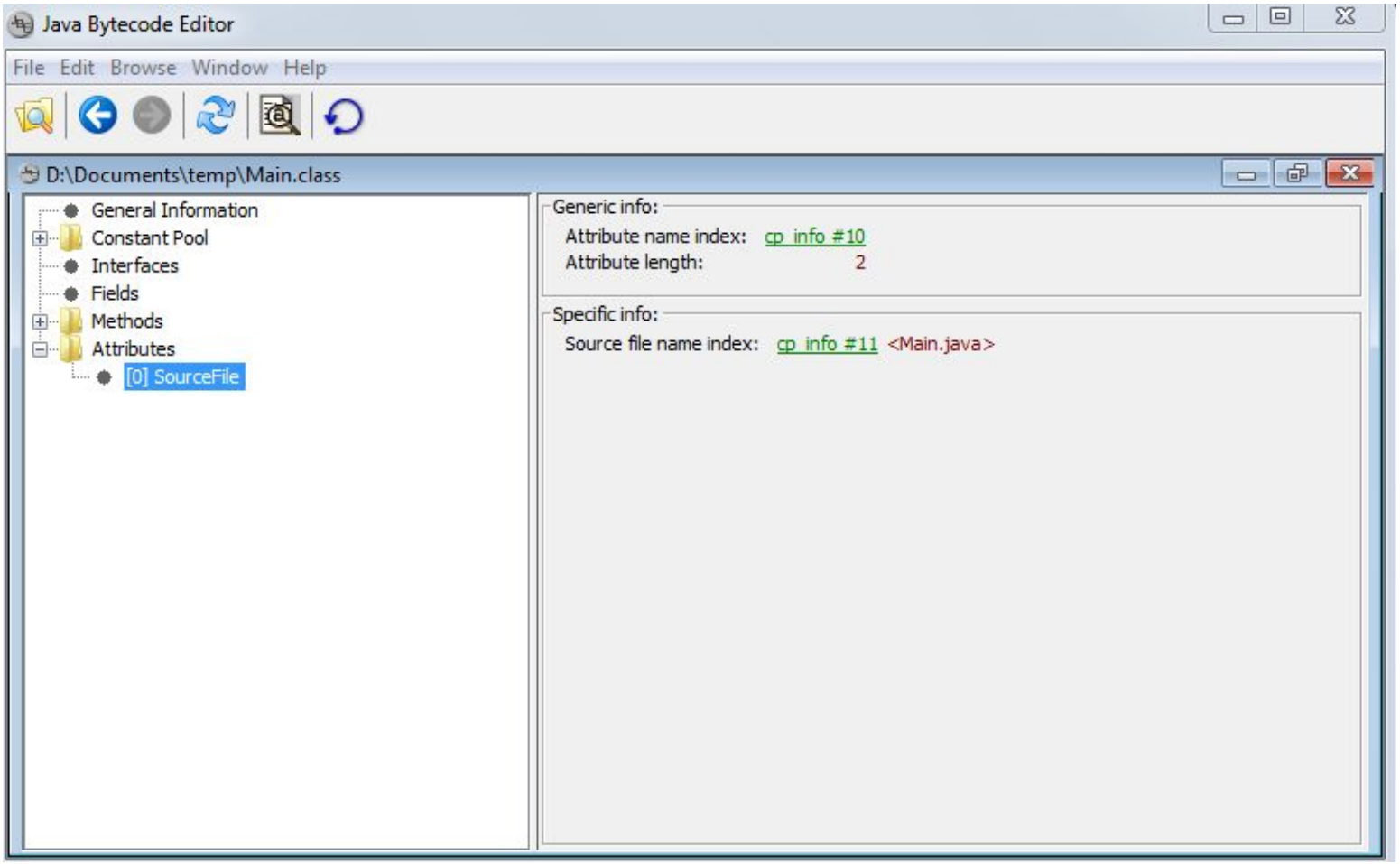

Java Bytecode Editor
File Edit Browse Window Help
D:\Documents\temp\Main.class
General Information
Constant Pool
Interfaces
Fields
Methods
<init>
[0] Code
main
[0] Code
Attributes
[0] SourceFile
Generic info:
Attribute name index: cp_info #6
Attribute length: 29
Specific info:
Bytecode
Exception table
Misc
Code Editor
Save method
Maximum stack depth: 1
Maximum local variables: 1
Code length: 5


Java Bytecode Editor
File Edit Browse Window Help
D:\Documents\temp\Main.class
General Information
Constant Pool
Interfaces
Fields
Methods
<init>
[0] Code
main
[0] Code
Attributes
[0] SourceFile
Generic info:
Attribute name index: cp_info #6
Attribute length: 29
Specific info:
Bytecode
Exception table
Misc
Code Editor
Save method
1 aload_0
2 invokespecial java/lang/Object/<init>()V
3 return


Java Bytecode Editor
File Edit Browse Window Help
D:\Documents\temp\Main.class
General Information
Constant Pool
Interfaces
Fields
Methods
<init>
main
Attributes
Name: cp_info #8 <main>
Descriptor: cp_info #9 <([Ljava/lang/String;)V>
Access flags: 0x0009 [public static]
Delete this entry


Java Bytecode Editor
File Edit Browse Window Help
D:\Documents\temp\Main.class
General Information
Constant Pool
Interfaces
Fields
Methods
<init>
[0] Code
main
[0] Code
Attributes
Generic info:
Attribute name index: cp_info #6
Attribute length: 45
Specific info:
Bytecode
Exception table
Misc
Code Editor
Save method
Maximum stack depth: 2
Maximum local variables: 4
Code length: 9


Java Bytecode Editor
File Edit Browse Window Help
D:\Documents\temp\Main.class
General Information
Constant Pool
Interfaces
Fields
Methods
<init>
[0] Code
main
[0] Code
Attributes
Generic info:
Attribute name index: cp_info #6
Attribute length: 45
Specific info:
Bytecode
Exception table
Misc
Code Editor
Save method
1 iconst_4
2 istore_1
3 iconst_5
4 istore_2
5 iload_1
6 iload_2
7 iadd
8 istore_3
9 return


Java Bytecode Editor
File Edit Browse Window Help
D:\Documents\temp\Main.class
General Information
Constant Pool
Interfaces
Fields
Methods
Attributes
[0] SourceFile
Generic info:
Attribute name index: cp_info #10
Attribute length: 2
Specific info:
Source file name index: cp_info #11 <Main.java>

Исходный код

Файл Main.java

```
public class Main
{
    public static void main (String [] args )
    {
          int a = 4;
     int b = 5;
     int c = a+b;
    }
}
```

> **javac Main.java**

Просмотр байт-кода

> javap -c Main.class

```
Compiled from "Main.java"
public class Main {
  public Main();
    Code:
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."<init>":
()V
       4: return

  public static void main(java.lang.String[]);
    Code:
       0: iconst_4
       1: istore_1
       2: iconst_5
       3: istore_2
       4: iload_1
       5: iload_2
       6: iadd
       7: istore_3
       8: return
}
```

Байт-код Java

Байт-код Java — набор инструкций, исполняемых виртуальной машиной Java. Каждый код операции байт-кода имеет размер один байт.

Виды инструкций

- загрузка и сохранение;
- арифметические и логические операции;
- преобразование типов;
- создание и преобразование объекта;
- управление стеком;
- операторы перехода;
- вызовы методов и возврат;

Формат инструкций

Инструкции загрузки

Загрузка типа int из локальной переменной:

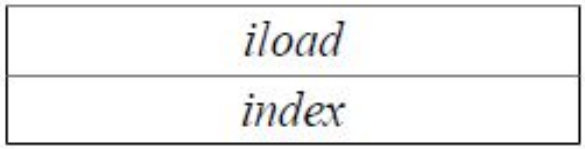
iload_<n>

- *iload_0 = 26 (0x1a)*
- *iload_1 = 27 (0x1b)*
- *iload_2 = 28 (0x1c)*
- *iload_3 = 29 (0x1d)*

Не имеют аргументов.

Инструкции загрузки

Загрузка типа int из локальной переменной:



iload = 21 (0x15)

Имеет один аргумент.

Инструкции загрузки

Загрузка из локальной переменной

i*load* — для типа **int**

l*load* — для типа **long**

f*load* — для типа **float**

d*load* — для типа **double**

Инструкции сохранения

Сохранение локальной переменно типа int:

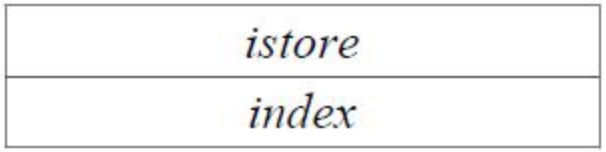
istore_<n>

- *istore_0 = 59 (0x3b)*
- *istore_1 = 60 (0x3c)*
- *istore_2 = 61 (0x3d)*
- *istore_3 = 62 (0x3e)*

Не имеют аргументов

Инструкции сохранения

Сохранение локальной переменной типа int:



istore = 54 (0x36)

Имеет один аргумент

Инструкции загрузки

Сохранение в локальной переменной

***i**store* — для типа **int**

***l**store* — для типа **long**

***f**store* — для типа **float**

***d**store* — для типа **double**

Математические операции

***i**add = 96 (0x60) – сложение типа* **int**

***l**add = 97 (0x61) – сложение типа* **long**

***f**add = 98 (0x62) – сложение типа* **float**

***d**add = 99 (0x63) – сложение типа* **double**

***i**mul = 104 (0x68) – умножение типа int*

Примеры кода

Исходный код	Байт-код

int a = 0;
int b = 1;
int c = a + b;
	iconst_0
	istore_1
	iconst_1
	istore_2
	iload_1
	iload_2
	iadd
	istore_3

Исходный код	Байт-код

long a = 0;
long b = 1;
long c = a + b;
	lconst_0
	lstore_1
	lconst_1
	lstore_3
	lload_1
	lload_3
	ladd
	lstore 5

Примеры кода

Исходный код	Байт-код

int a = 2;
int b = 3;
int c = 1;
int d = a * b + c;
	iconst_2
	istore_1
	iconst_3
	istore_2
	iconst_1
	istore_3
	iload_1
	iload_2
	imul
	iload_3
	iadd
	istore 4

Java Virtual Machine

- Виртуальная машина java реализована в виде стековой машины.
- Команды делятся на два вида: аргументы и операции.
- Аргументы расположены перед операциями.
- Аргументы - добавляют элементы в стек.
- Операции – извлекают элементы из стека.

Пример вычисления

Исходный код	Байт-код

int a = 2;
int b = 3;
int c = 1;
int d = a * b + c;
	……
	1) iload_1
	2) iload_2
	3) imul
	4) iload_3
	5) iadd
	6) istore 4

Пример вычисления

Исходный код	Байт-код

int a = 2;
int b = 3;
int c = 1;
int d = a * b + c;
	……
	1) iload_1
	2) iload_2
	3) imul
	4) iload_3
	5) iadd
	6) istore 4

1) **2**

Пример вычисления

Исходный код	Байт-код

int a = 2;
int b = 3;
int c = 1;
int d = a * b + c;
	……
	1) iload_1
	2) iload_2
	3) imul
	4) iload_3
	5) iadd
	6) istore 4

1)

2)

Пример вычисления

Исходный код	Байт-код

int a = 2;
int b = 3;
int c = 1;
int d = a * b + c;
	……
	1) iload_1
	2) iload_2
	3) imul
	4) iload_3
	5) iadd
	6) istore 4

1)

2)

3)

Пример вычисления

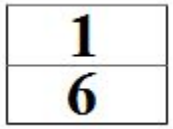
Исходный код	Байт-код

int a = 2;
int b = 3;
int c = 1;
int d = a * b + c;
	……
	1) iload_1
	2) iload_2
	3) imul
	4) iload_3
	5) iadd
	6) istore 4

1)

2)

3)

4) 

Пример вычисления

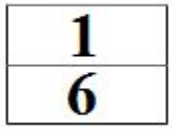
Исходный код	Байт-код

int a = 2;
int b = 3;
int c = 1;
int d = a * b + c;
	……
	1) iload_1
	2) iload_2
	3) imul
	4) iload_3
	5) iadd
	6) istore 4

1)

2)

3)

4) 

5)

Пример вычисления

Исходный код	Байт-код

int a = 2;
int b = 3;
int c = 1;
int d = a * b + c;
	……
	1) iload_1
	2) iload_2
	3) imul
	4) iload_3
	5) iadd
	6) istore 4

1) **2**

2) **3**
2

3) **6**

4) **1**
6

5) **7**

6) **стек пуст**

СТЕГАНОГРАФИЯ (В широком смысле).
Англоязычный термин"Скрытие информации" (Information hiding (IH)).

Определение. IH - это семейство методов, при помощи которых некоторое дополнительное сведение погружается в основное
(покрывающее сообщение (ПС)) при сохранении хорошего качества ПС.

Две основные части IH:
1. Собственно стеганография (стеганография).
2. Цифровые "водяные знаки" (ЦВЗ).

Задачи стеганографии:
Погрузить дополнительное сообщение в ПС так, чтобы сам факт его присутствия в нем нельзя было бы обнаружить нелегитимным пользователям.

Задача ЦВЗ:
Погрузить дополнительные сведения (обычно идентификационный код автора) в ПС так, чтобы его нельзя было бы удалить, не ухудшив существенно качество ПС.
(Факт такого вложения может и обнаруживаться нелегитимными пользователями.)

Типичные ПС

- неподвижное изображение
- подвижное изображение (видео)
- аудио файлы
- речь
- печатный смысловой текст
- интернет - протоколы
- программы для компьютеров(исполняемые файлы).

Вложение в class файл

- Применяется для защиты авторских прав.
- Не должно нарушать работу программы.
- Не должно быть обнаружимо.

Методы вложения в исполняемый class файл

- **Модификация class файла**

Перестановка элементов массивов:

1. Интерфейсов
2. Полей
3. Методов
4. Атрибутов

- **Модификация байт-кода**

1. Вставка операций
2. Замена конструкций на эквивалентные
3. Перестановка операций

Количество бит, которые можно вложить за счет перестановки массивов.

Имеется массив из N элементов.

Количество комбинаций равно N!

$$2^b = N!$$

Число бит которые можно будет вложить равно

$$b = [log_2 (N!)]$$

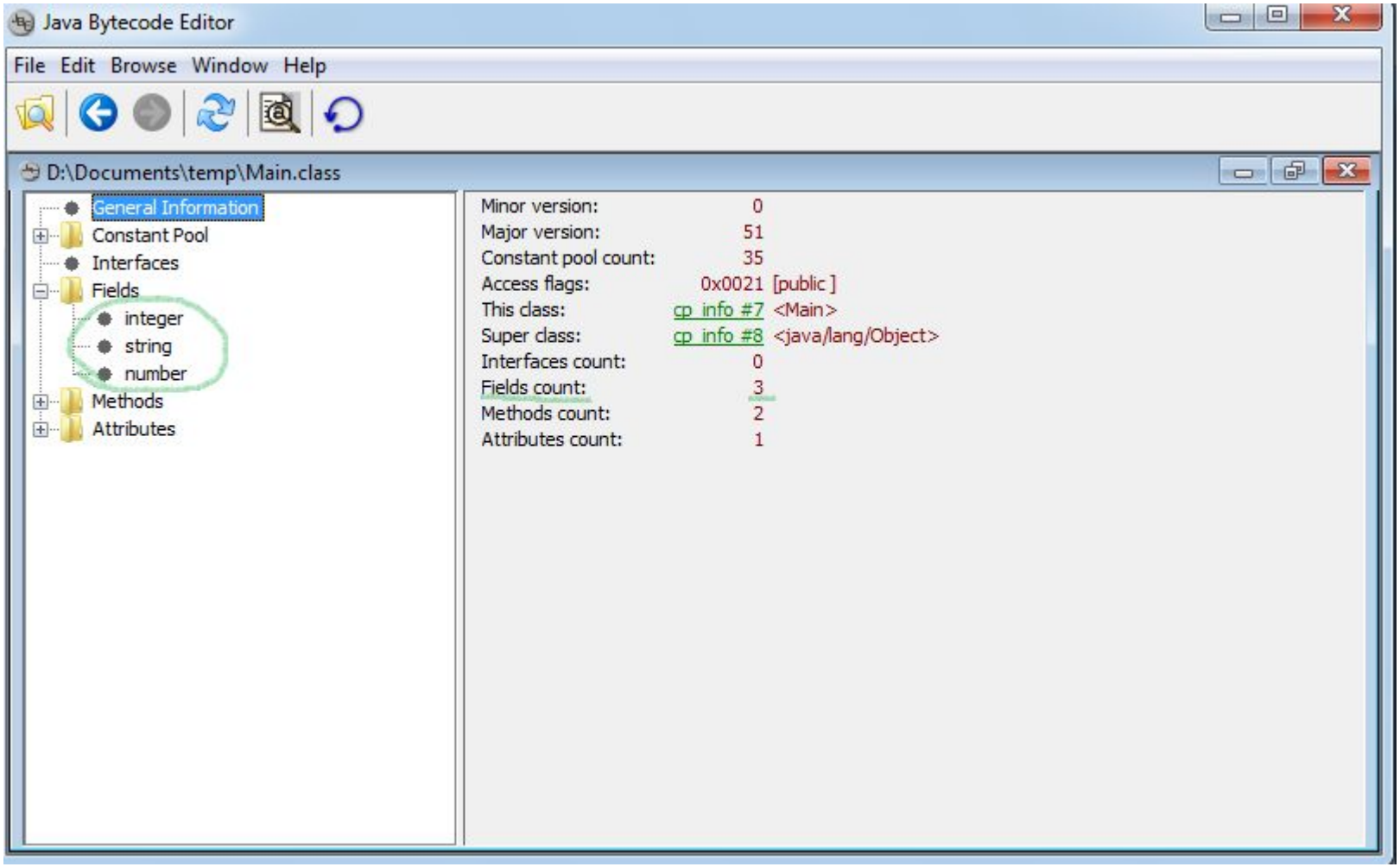
N	2	3	4	5	6	7	8
b	1	2	4	6	9	12	15

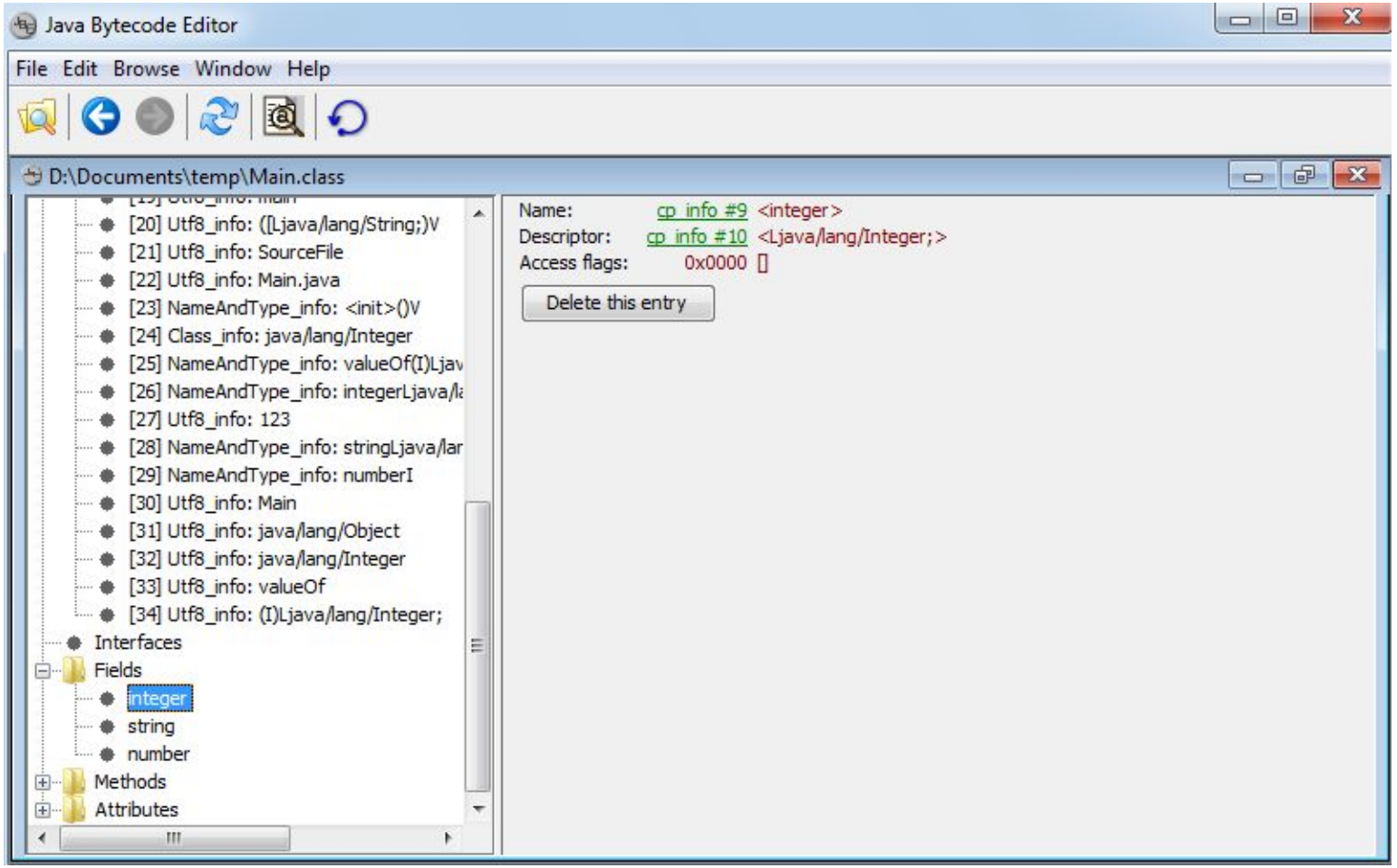
Файл Main.java

```
public class Main
{
	Integer integer = 4;
	String string = "123";
	int number = 5;
	public static void main (String [] args )
	{
	}

}
```

> javac Main.java


Java Bytecode Editor
File Edit Browse Window Help
D:\Documents\temp\Main.class
General Information
Constant Pool
Interfaces
Fields
integer
string
number
Methods
Attributes
Minor version: 0
Major version: 51
Constant pool count: 35
Access flags: 0x0021 [public]
This class: cp_info #7 <Main>
Super class: cp_info #8 <java/lang/Object>
Interfaces count: 0
Fields count: 3
Methods count: 2
Attributes count: 1


Java Bytecode Editor
File Edit Browse Window Help
D:\Documents\temp\Main.class
[20] Utf8_info: ([Ljava/lang/String;)V
[21] Utf8_info: SourceFile
[22] Utf8_info: Main.java
[23] NameAndType_info: <init>()V
[24] Class_info: java/lang/Integer
[25] NameAndType_info: valueOf(I)Ljav
[26] NameAndType_info: integerLjava/l
[27] Utf8_info: 123
[28] NameAndType_info: stringLjava/lar
[29] NameAndType_info: numberI
[30] Utf8_info: Main
[31] Utf8_info: java/lang/Object
[32] Utf8_info: java/lang/Integer
[33] Utf8_info: valueOf
[34] Utf8_info: (I)Ljava/lang/Integer;
Interfaces
Fields
integer
string
number
Methods
Attributes
Name: cp_info #9 <integer>
Descriptor: cp_info #10 <Ljava/lang/Integer;>
Access flags: 0x0000 []
Delete this entry

Перестановка элементов массивов полей

Количество бит, которые мы можем вложить равно

$$[log_2 (\text{fields_count}!)]$$

Для массива интерфейсов и атрибутов все аналогично.

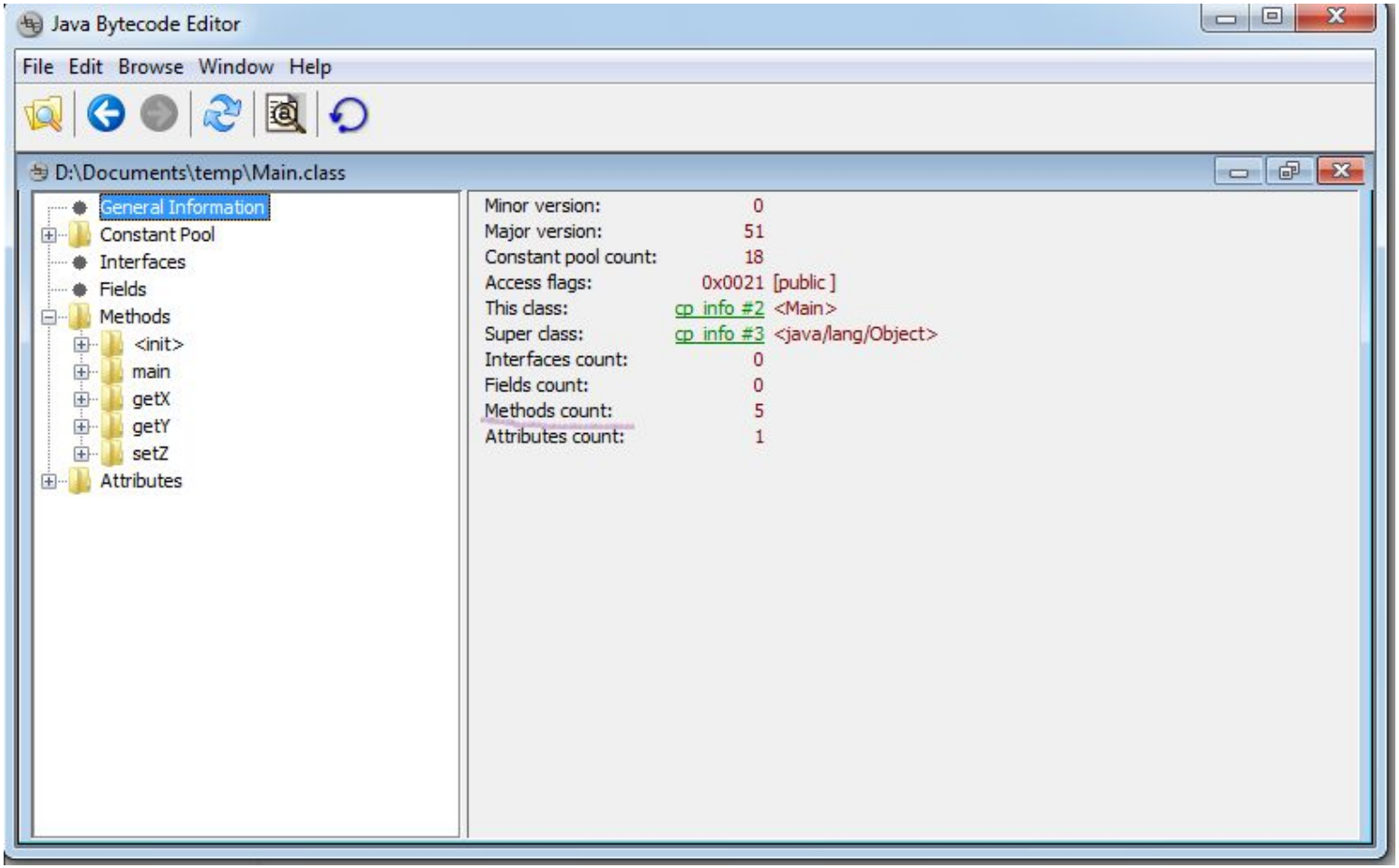
Перестановка элементов массивов методов

Файл Main.java

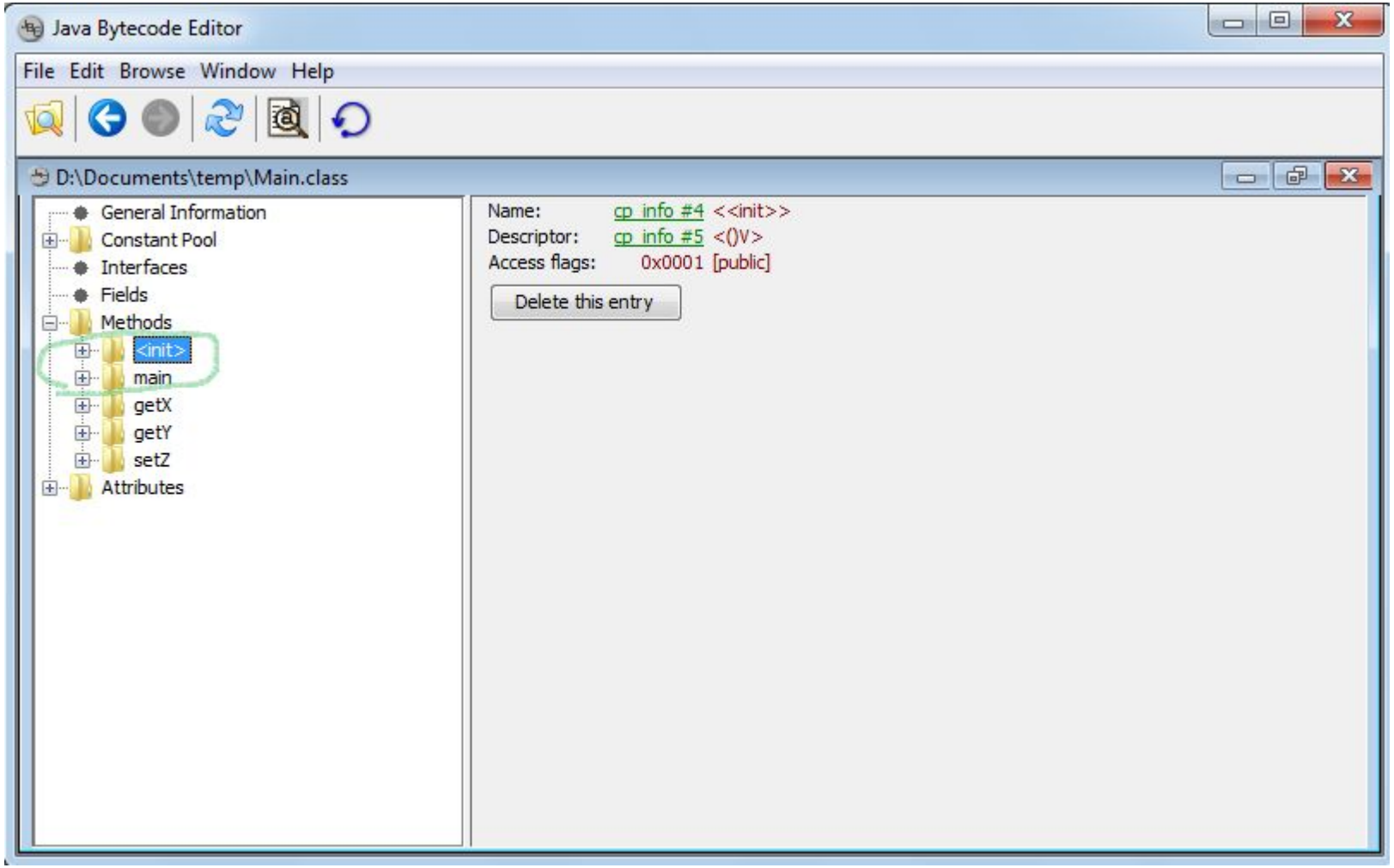
```
public class Main
{
   public static void main (String [] args )
   {
     }
   void getX()
     {}
     void getY()
     {}
     void setZ()
     {}
}
```

> javac Main.java

Перестановка элементов массивов методов



Перестановка элементов массивов методов



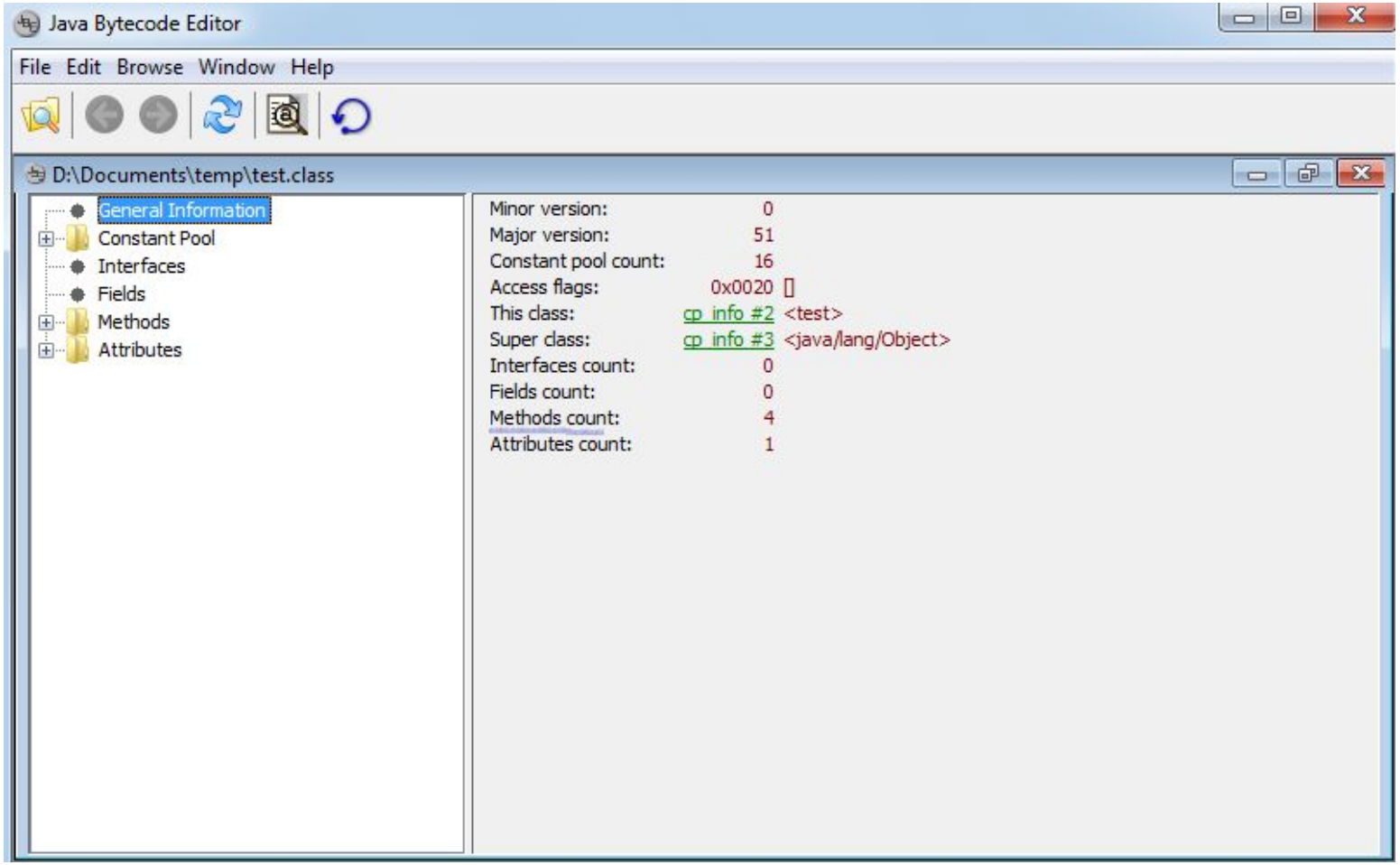
Перестановка элементов массивов методов

Файл Main.java

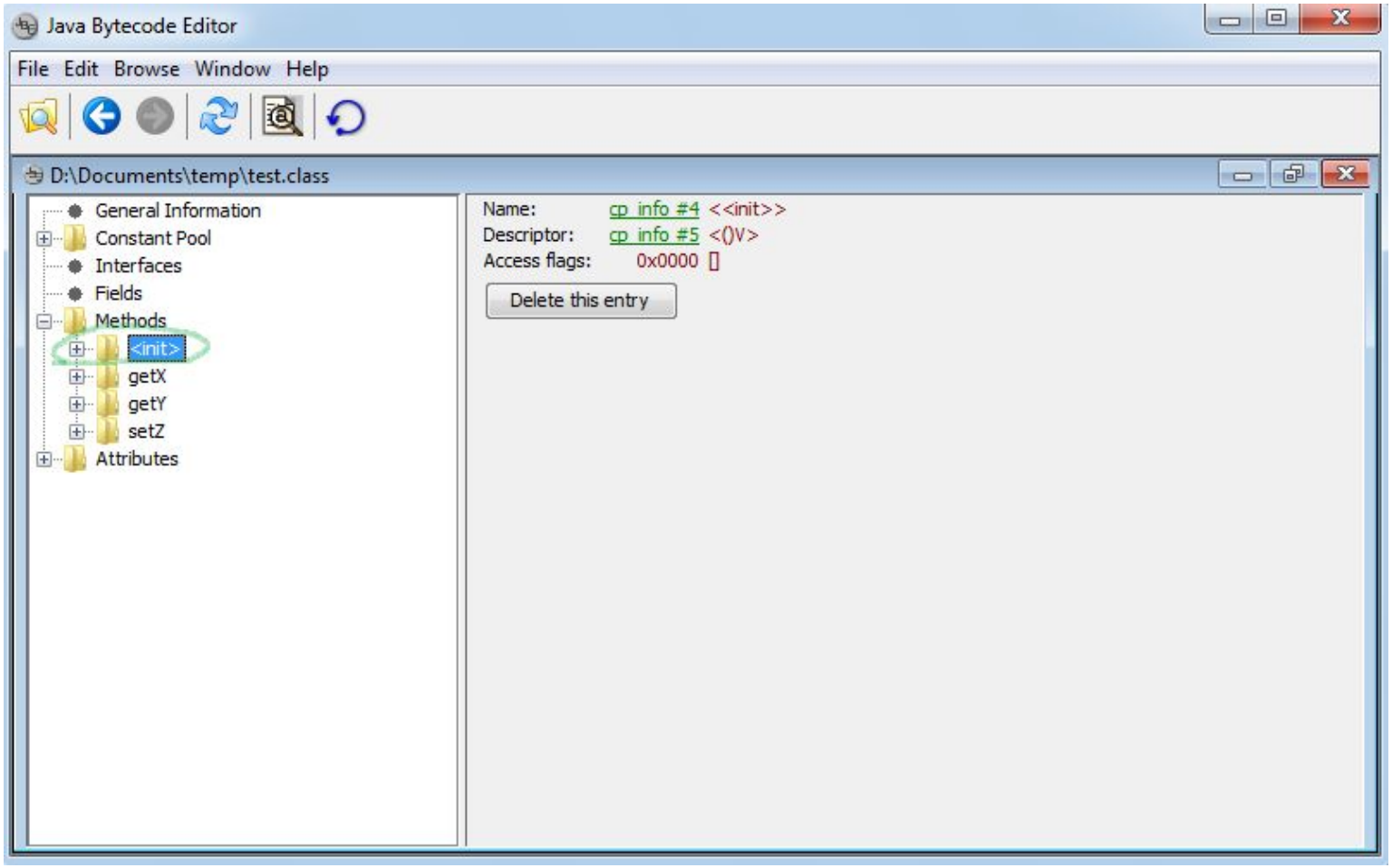
```
public class test
{
   test()
   {
    }
   void getX()
     {}
     void getY()
     {}
     void setZ()
     {}
}
```

> javac test.java

Перестановка элементов массивов методов



Перестановка элементов массивов методов



Перестановка элементов массивов методов

- Количество бит которые можно вложить за счет перестановки массивов методов равно:

$$[log_2((methods_count - k)!)]$$

Если класс содержит функцию **main**, то k = 2 , если нет то k = 1.

Количество бит которые можно вложить за счет перестановки массивов в class файле равно:

$$[log_2\,(\text{interfaces_count!})] + [log_2\,(\text{fields_count!})] +$$
$$[log_2\,(\text{attributes_count!})] + [log_2\,((\text{methods_count} - \text{k})!)]$$

Если класс содержит функцию **main**, то k = 2 , если нет то k = 1.

Перестановка элементов массивов методов

- Не меняют размер исполняемого файла.
- Объем вложения ограничен.
- Не обнаруживается

Модификация байт-кода

1. Вставка операций
2. Замена конструкций на эквивалентные
3. Перестановка операций

Вставка операций

Исходный код	Байт код
int a; a = 2;	iconst_2 istore_1

Исходный код (измененный)	Байт код (измененный)

int a;
a = 123;
a = 2423;
a = 4234;
a = 2;
	bipush 123
	istore_1
	sipush 2423
	istore_1
	sipush 4234
	istore_1
	iconst_2
	istore_1

Размер вложенной информации 4*3 = 12 байт

Вставка операций

- Увеличивает размер исполняемого файла.
- Объем вложения неограничен.
- Обнаруживается с помощью статистического анализа.

Замена конструкций на эквивалентные

- Замена математических операций на эквивалентные
- Замена ветвлений кода в условиях

Замена математических операций на эквивалентные

- a+b ⇔ a-(-b)
- a-b ⇔ a+(-b)
- a*b ⇔ (-a)*(-b)
- a*(-b) ⇔ (-a)*(b)

Замена математических операций на эквивалентные

Исходный код	Байт код	Вкладываемый бит

int a = 2;
 	
int b = 4;
 	
int c = a + b;
	iconst_2
	istore_1
	iconst_4
	istore_2
	iload_1
	iload_2
	iadd
	istore_3
		0

int a = 2;
 	
int b = 4;
 	
int c = a -(-b);
	iconst_2
	istore_1
	iconst_4
	istore_2
	iload_1
	iload_2
	ineg
	isub
	istore_3
		1

Замена математических операций на эквивалентные

Исходный код	Байт код	Вкладываемый бит

int a = 2;
int b = 4;
int c = a-b;
	iconst_2
	istore_1
	iconst_4
	istore_2
	iload_1
	iload_2
	isub
	istore_3
		0

int a = 2;
int b = 4;
int c = a+(-b);
	iconst_2
	istore_1
	iconst_4
	istore_2
	iload_1
	iload_2
	ineg
	iadd
	istore_3
		1

Замена математических операций на эквивалентные

Исходный код	Байт код	Вкладываемый бит

int a = 2;
int b = 4;
int c = a*b;
	iconst_2
	istore_1
	iconst_4
	istore_2
	iload_1
	iload_2
	imul
	istore_3
		0

int a = 2;
int b = 4;
int c = (-a)*(-b);
	iconst_2
	istore_1
	iconst_4
	istore_2
	iload_1
	ineg
	iload_2
	ineg
	imul
	istore_3
		1

Замена математических операций на эквивалентные

Исходный код	Байт код	Вкладываемый бит

int a = 2;
int b = 4;
int c = a*(-b);
	iconst_2
	istore_1
	iconst_4
	istore_2
	iload_1
	iload_2
	ineg
	imul
	istore_3
		0

int a = 2;
int b = 4;
int c = (-a)*b;
	iconst_2
	istore_1
	iconst_4
	istore_2
	iload_1
	ineg
	iload_2
	imul
	istore_3
	return
		1

Замена математических операций на эквивалентные

- Могут увеличить объём исполняемого файла но не значительно.
- Объем вложения ограничен.
- Обнаружение зависит от вложенного объёма информации.

Замена ветвлений кода в условиях

```
ЕСЛИ a < b
ТОГДА команды1
ИНАЧЕ команды2
```

```
ЕСЛИ a > b
ТОГДА команды2
ИНАЧЕ команды1
```

Замена ветвлений кода в условиях

Исходный код	Байт код	Вкладываемый бит

if(a > b)
{
c = 0;
}
else
{
c = 1;
}
	4: iload_1
	5: iload_2
	6: **if_icmple** 14
	9: iconst_0
	10: istore_3
	11: goto 16
	14: iconst_1
	15: istore_3
	16: return
		0

if(a < b)
{
c = 1;
}
else
{
c = 0;
}
	4: iload_1
	5: iload_2
	6: **if_icmpge** 14
	9: iconst_1
	10: istore_3
	11: goto 16
	14: iconst_0
	15: istore_3
	16: return
		1

Замена ветвлений кода в условиях

- Не меняют размер исполняемого файла.
- Объем вложения ограничен.
- Обнаружение зависит от вложенного объёма информации.

Перестановка операций

- Перестановка элементов, участвующих в операциях.
- Изменение порядка присваивания переменных.

Перестановка элементов, участвующих в операциях

- a+b ⇔ b+a
- a+b+c ⇔ a+c+b
- a*b ⇔ b*a
- a*b*c ⇔ a*c*b

Математические операции: сложение, умножение.

Логические операции: AND, OR, XOR

$$x_1 + x_2 + \cdots + x_n$$

Количество, вкладываемых бит равно

$$[log_2\,(\mathrm{n!})]$$

При условии что все элементы разные.

Перестановка элементов, участвующих в операциях

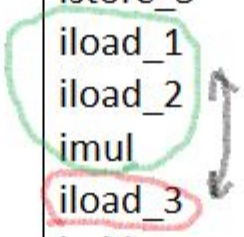
Исходный код	Байт код	Вкладываемый бит

int a = 2;
int b = 4;
int c = **a + b;**
	iconst_2
	istore_1
	iconst_4
	istore_2
	iload_1
	iload_2
	iadd
	istore_3
		0

int a = 2;
int b = 4;
int c = **b + a;**
	iconst_2
	istore_1
	iconst_4
	istore_2
	iload_2
	iload_1
	iadd
	istore_3
		1

Исходный код	Байт код

int a = 2;
int b = 4;
int c = 5;
int d = a * b + c;
	iconst_2
	istore_1
	iconst_4
	istore_2
	iconst_5
	istore_3
	iload_1
	iload_2
	imul
	iload_3
	iadd
	istore 4



Исходный код	Байт код

int a = 2;
int b = 4;
int c = 5;
int d = a * b + c;
	iconst_2
	istore_1
	iconst_4
	istore_2
	iconst_5
	istore_3
	iload_1
	iload_2
	imul
	iload_3
	iadd
	istore 4

Перестановка элементов, участвующих в операциях.

- Не меняют размер исполняемого файла.
- Объем вложения ограничен.
- Обнаружение зависит от вложенного объёма информации.

Изменение порядка присваивания переменных.

Исходный код	Байт код	Вкладываемый бит
int b; int a; a = 2; **b = 3;**	iconst_2 istore_2 iconst_3 istore_1	0
int b; int a; **b = 3;** a = 2;	iconst_3 istore_1 iconst_2 istore_2	1

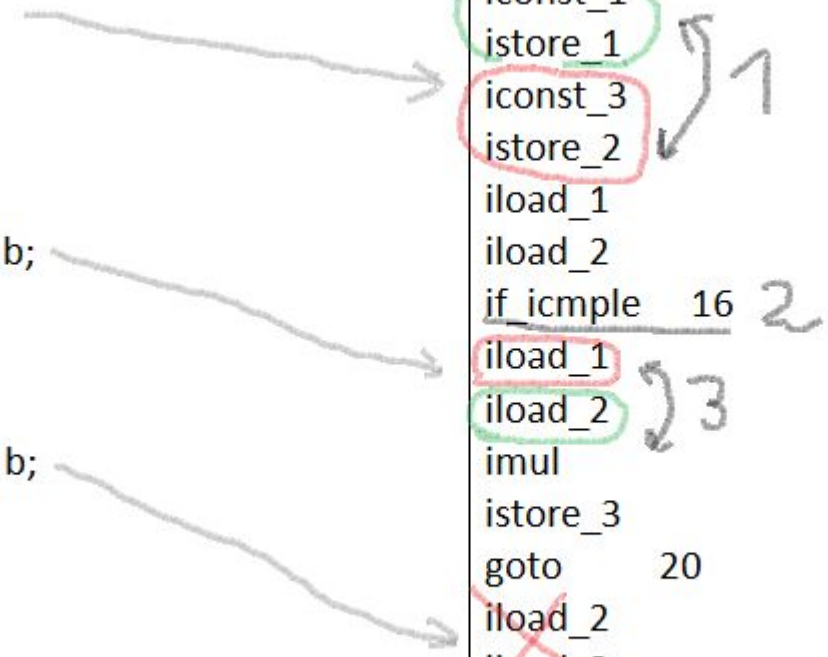
Перестановка не должна менять результат вычисления

Изменение порядка присваивания переменных.

- Не меняют размер исполняемого файла.
- Объем вложения ограничен.
- Обнаружение зависит от вложенного объёма информации.

Исходный код	Байт код

int a = 1;
int b = 3;
int c;
if(a > b)
{
  c = a * b;
}
else
{
  c = b * b;
}
	iconst_1
	istore_1
	iconst_3
	istore_2
	iload_1
	iload_2
	if_icmple 16
	iload_1
	iload_2
	imul
	istore_3
	goto 20
	iload_2
	iload_2
	imul
	istore_3



Можем вложить три бита

Сравнение методов вложения

Метод вложения	**Размер файла**	**Объем вложения**	**Обнаружимость**
Перестановка элементов массивов	Не меняется	Ограничен	Не обнаруживается
Вставка операций	Увеличивается	Неограничен	Обнаруживается
Замена конструкций	Может, увеличивается (но не значительно)	Ограничен	Не обнаруживается (при определенном объеме вложения)
Перестановка операций	Не меняется	Ограничен	