

9. OOP. JAVA

java.io.File, java.io.FileFilter. JNI

Класс File

- Предназначен для работы с элементами файловой системы (ЭФС).
- Представление пути к ЭФС зависит от операционной системы.
- Объект класса **File** представляет собой абстрактное представление пути к ЭФС (файлу или каталогу), при этом **сам ЭФС может отсутствовать**.

Метод getPath

Метод **getPath** класса **File** возвращает строковое представление абстрактного пути к ЭФС, который был задан при создании объекта **File**, при этом используется разделитель файловой системы ОС, в которой выполняется JVM.

Метод `getAbsolutePath`

Метод `getAbsolutePath` класса `File` возвращает т.н. **абсолютный** путь к ЭФС вид которого зависит от ОС.

Метод `listFiles`

Метод `listFiles` класса `File` возвращает массив объектов `File`, содержащихся в каталоге, на который указывает объект `this`.

```
public File[] listFiles()
```

Возможны три случая:

1) **this** ссылается на непустой каталог: возвращаемый массив будет включать объекты **File**, соответствующие элементам файловой системы, которые содержатся в этом каталоге;

2) **this** ссылается на пустой каталог: возвращаемый массив будет иметь **нулевую длину**;

3) **this** ссылается на файл (не каталог), в этом случае метод `listFiles` вернет значение **null**.

Интерфейс FileFilter

Метод **listFiles** может принимать в качестве параметра объект класса, который реализует интерфейс **FileFilter**. Этот интерфейс содержит один метод **accept**.

```
boolean accept(File pathname)
```

В результирующий список, который возвращает **listFiles** будут включены те и только те объекты **File**, для которых метод **accept** возвращает **true**.

Метод getParent

Метод **getParent** класса **File** возвращает часть абстрактного пути ЭФС.

Метод `getCanonicalPath`

- Метод `getCanonicalPath` класса `File` возвращает т.н. канонический путь к ЭФС.
- Данный метод пытается определить реальный путь к ЭФС в файловой системе ОС, на которой выполняется JVM.
- Процесс определения зависит от ОС.

Java Native Interface (JNI)

это стандартный программный интерфейс для использования в Java программах кода на других языках (например, С или С++) и наоборот – встраивания виртуальной машины Java в программы на других языках (Invocation API).

<http://docs.oracle.com/javase/8/docs/technotes/guides/jni/>

<http://www.ibm.com/developerworks/java/tutorials/j-jni/j-jni.html>

JNI

Инструменты и компоненты

- Java compiler:
`javac.exe`
- Java virtual machine (JVM):
`java.exe`
- C Header and Stub File Generator:
`javah.exe`
- Библиотеки и заголовочные файлы JNI:
`jni.h`, `jni_md.h`, `jvm.lib`, `jvm.dll` или `jvm.so`
- C/C++ компилятор для создания динамической библиотеки:
`Visual C++` или `GCC`

JNI

Вызов C/C++ кода из Java программы

- 1) Написать Java код.
- 2) Откомпилировать Java код.
- 3) Создать C/C++ заголовочный файл.
- 4) Написать C/C++ код.
- 5) Создать файл динамической библиотеки.
- 6) Выполнить Java программу.

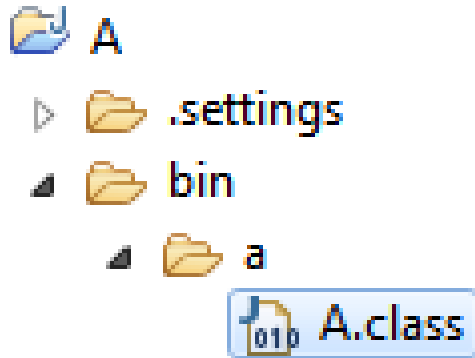
JNI

Написать Java код

```
package a;
public class A {
    static {
        // Имя библиотеки без расширения.
        System.loadLibrary("cls");
    }
    public static native void cls();
    public static void main(String[] args) {
        cls();
        System.out.println("AAA");
    }
}
```

JNI

Откомпилировать Java код



```
D:\pro\eclipse\java\A\bin>java a.A
```

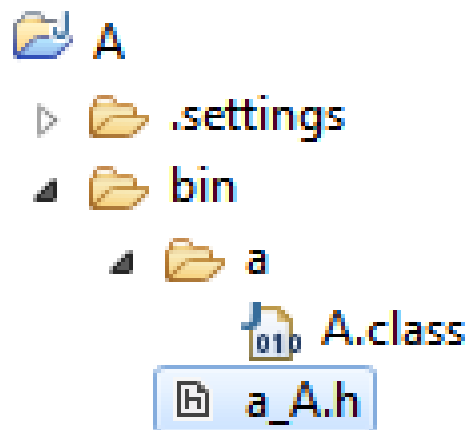
```
Exception in thread "main" java.lang.UnsatisfiedLinkError: no cls in java.library.path
    at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1864)
    at java.lang.Runtime.loadLibrary0(Runtime.java:870)
    at java.lang.System.loadLibrary(System.java:1122)
    at a.A.<clinit>(A.java:6)
```

```
D:\pro\eclipse\java\A\bin>
```

JNI

Создать C/C++ заголовочный файл

javah a.A

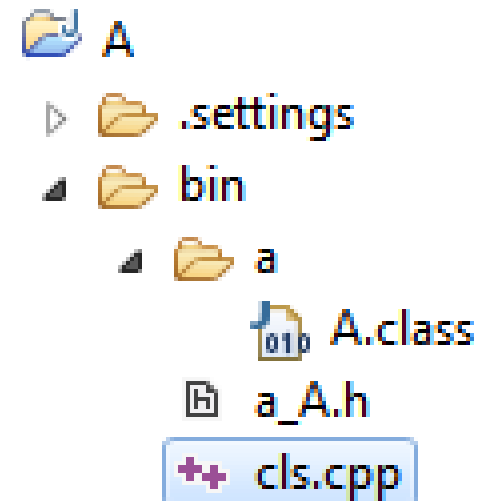


```
1  /* DO NOT EDIT THIS FILE - it is machine generated */
2  #include <jni.h>
3  /* Header for class a_A */
4
5  #ifndef _Included_a_A
6  #define _Included_a_A
7  #ifdef __cplusplus
8  extern "C" {
9  #endif
10 /*
11  * Class:      a_A
12  * Method:     cls
13  * Signature:  ()V
14  */
15 JNIEXPORT void JNICALL Java_a_A_cls
16     (JNIEnv *, jclass);
17
18 #ifdef __cplusplus
19 }
20 #endif
21 #endif
```

JNI

Написать C/C++ код

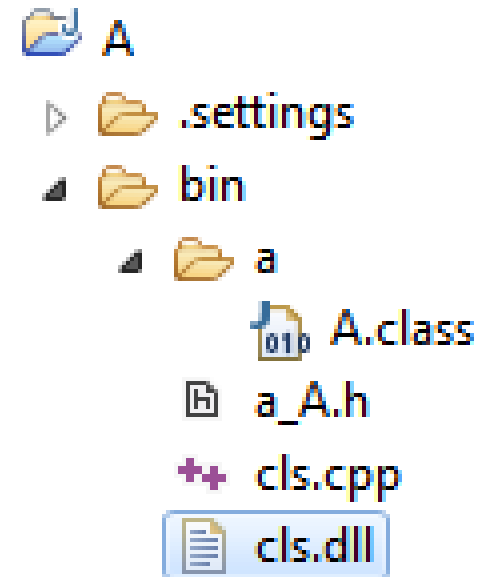
```
1 #include "a_A.h"
2 #include <stdlib.h>
3
4 JNIEXPORT void JNICALL Java_a_A_cls
5     (JNIEnv *, jclass) {
6     system("cls");
7 }
```



JNI

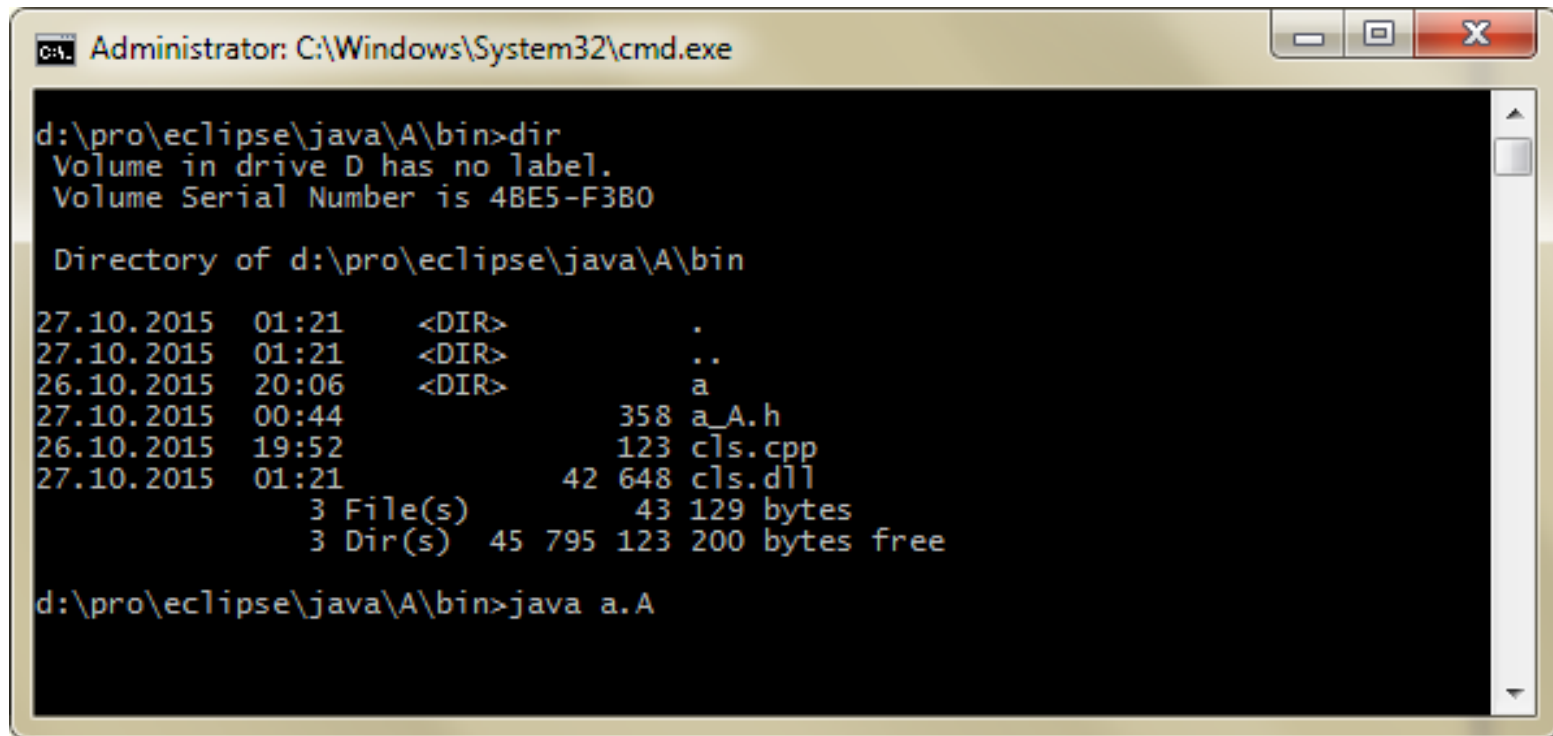
Создать файл динамической библиотеки

```
x86_64-w64-mingw32-c++  
-IC:\java\jdk\Include  
-IC:\java\jdk\include\win32  
-shared  
-o cls.dll  
cls.cpp
```



JNI

Выполнить Java программу



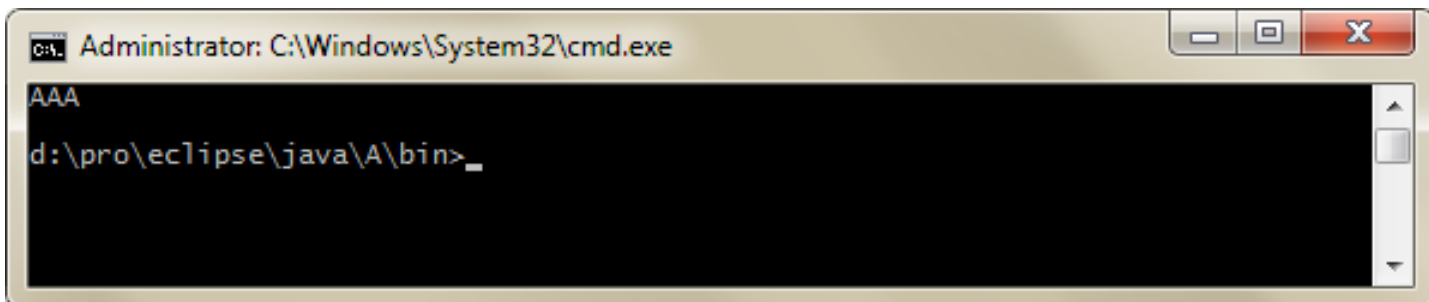
Administrator: C:\Windows\System32\cmd.exe

```
d:\pro\eclipse\java\A\bin>dir
Volume in drive D has no label.
Volume Serial Number is 4BE5-F3B0

Directory of d:\pro\eclipse\java\A\bin

27.10.2015  01:21    <DIR>          .
27.10.2015  01:21    <DIR>          ..
26.10.2015  20:06    <DIR>          a
27.10.2015  00:44             358 a_A.h
26.10.2015  19:52             123 cls.cpp
27.10.2015  01:21        42 648 cls.dll
               3 File(s)              43 129 bytes
               3 Dir(s)  45 795 123 200 bytes free

d:\pro\eclipse\java\A\bin>java a.A
```



Administrator: C:\Windows\System32\cmd.exe

```
AAA
d:\pro\eclipse\java\A\bin>_
```