

Программирование на языке Java. Инструментальные средства JDK

Картузов А.В.

1. `appletviewer` — программа просмотра апплетов Java

1.1. Доступность

JDK версии 1.0 и более поздних версий.

1.2. Синтаксис вызова

```
appletviewer [-debug] [-Jаргумент] [-encoding кодировка] url/файл...
```

1.3. Описание

Программа `appletviewer` загружает один или несколько HTML-документов по указанным в командной строке URL. Она читает или загружает все апплеты, на которые содержатся ссылки в документах, и отображает каждый из них в собственном окне. Если ни один из названных документов не содержит тег `<applet>`, `appletviewer` не делает ничего.

1.4. Параметры

`-debug` При указании данного параметра `appletviewer` запускается под управлением `jdb` (отладчик Java). Это позволяет отлаживать апплет, на который ссылается документ или документы.

`-J` аргумент Передает аргумент командной строки интерпретатору Java. Указанный аргумент не должен иметь пробелов. Если интерпретатору Java нужно передать

аргумент, включающий в себя несколько слов, следует использовать соответствующее количество параметров -J. Список допустимых параметров интерпретатора Java можно найти в параграфе, посвященном описанию Java. Доступен в JDK версии 1.1 или более поздних версий.

-encoding кодировка Определяет кодировку символов, которая должна использоваться программой `appletviewer` при чтении содержимого указанных файлов или URL. Он используется в процессе преобразования значений параметров апплета в Unicode. Доступен в JDK версии 1.1 или более поздних версий.

1.5. Команды

В окне программы `appletviewer` содержится единственное меню **Applet** со следующими командами:

Restart Останавливает и уничтожает текущий апплет, затем заново инициализирует его и запускает повторно.

Reload Останавливает, уничтожает и выгружает апплет, затем снова загружает, инициализирует и запускает его.

Stop Останавливает текущий апплет. Доступна в JDK версии 1.1 или более поздних версий.

Save Сериализует апплет и сохраняет его в файле `Applet.ser` в домашнем каталоге пользователя. Перед вызовом этой команды апплет необходимо остановить. Доступна в JDK версии 1.1 или более поздних версий.

Start Повторно запускает остановленный апплет. Доступна в JDK версии 1.1 или более поздних версий.

Clone Создает копию апплета в новом окне `appletviewer`.

Tag Открывает диалоговое окно, где выводится тег `<applet>` со всеми соответствующими тегами `<param>`, которые создают данный апплет.

Info Открывает диалоговое окно, содержащее информацию о данном апплете. Эту информацию позволяют получить методы `getAppletInfo ()` и `getParameterInfo ()`, которые реализуются апплетом.

Edit Эта команда не реализована, поэтому меню **Edit** не доступно.

Character Encoding Выводит текущую кодировку символов в строке состояния. Доступна в JDK версии 1.1 или более поздних версий.

Print Выводит апплет на печать. Доступна в JDK версии 1.1 или более поздних версий.

Properties Выводит диалоговое окно, которое позволяет пользователю устанавливать любимый набор параметров `appletviewer`, в том числе параметры брандмауэров и кэширующих Proxy-серверов.

Close Закрывает текущее окно `appletviewer`.

Quit Завершает работу `appletviewer`, закрывая все открытые окна.

1.6. Свойства

Программа `appletviewer` при запуске читает определения свойств из файла `~/.hotjava/properties` (в среде UNIX) или `\hotjava\properties` (в среде Windows), путь к которому определяется относительно переменной среды `home`. Эти свойства хранятся в списке системных свойств и используются для определения различных ошибок и выводимых апплетом сообщений о состоянии, а также для определения политики безопасности и особенностей применения Proxy-серверов. Свойства, которые влияют на безопасность и работу Proxy-серверов, приведены ниже.

1.7. Безопасность

Следующие свойства определяют те связанные с безопасностью ограничения, которые накладываются программой `appletviewer` на работу не пользующихся доверием апплетов.

acl.read Представляет собой список файлов и каталогов, которые разрешено читать не пользующимся доверием апплетам. Элементы списка должны разделяться двоеточиями в среде UNIX и точками с запятой в среде Windows. В среде UNIX символ—заменяется домашним каталогом текущего пользователя. Если в качестве элемента списка появляется символ `+`, он заменяется значением свойства `acl.read.default`. Проще всего разрешить доступ для чтения — задать `acl.read` равным `+`.

По умолчанию не пользующимся доверием апплетам запрещено читать какие-либо файлы или каталоги.

acl.read.default Представляет собой список файлов и каталогов, которые разрешено читать не пользующимся доверием апплетам, если свойство `acl.read` равно `+`.

acl.write Представляет собой список файлов и каталогов, в которые разрешено записывать не пользующимся доверием апплетам. Элементы списка должны разделяться двоеточиями в среде UNIX и точками с запятой в среде Windows. В среде UNIX символ `~` заменяется именем домашнего каталога текущего пользователя. Если в качестве элемента списка появляется символ `+`, то он заменяется значением свойства `acl.write, default`. Проще всего разрешить доступ для записи — задать `acl.write` равным `+`. По умолчанию не пользующимся доверием апплетам запрещено записывать в какие-либо файлы или каталоги.

acl.write.default Представляет собой список файлов и каталогов, в которые разрешено записывать не пользующимся доверием апплетам, если свойство `acl.write` равно `+`.

appletviewer.security.mode Указывает типы сетевого доступа, разрешенного не пользующимся доверием апплетам. Значение `none` показывает, что апплет вообще не может работать в сети, значение `host` (принято по умолчанию), — что апплет в состоянии связываться только с узлом, с которого он загружен, а значение `unrestricted`, — что апплет имеет возможность связаться с любым узлом без ограничений.

package.restrict.access. префикс_пакета Свойствам данного вида можно присвоить значение `true`, чтобы запретить не пользующимся доверием апплетам использовать классы любого пакета, имя которого начинается с указанного префикса. Например, чтобы помешать апплету использовать любой из классов `Sun` (такой как компилятор Java или сама программа просмотра апплетов), распространяемых в составе JDK, можно определить следующее свойство:

package.restrict.access.sun==true По умолчанию значение этого свойства равно `true` для пакетов `sun.*` и `netscape.*`.

package.restrict.definition.префикс_пакета Свойствам данного вида можно присвоить значение `true`, чтобы запретить не пользующимся доверием апплетам определять класс любого пакета, имя которого начинается с указанного префикса.

Например, чтобы помешать апплету определить классы в любом стандартном пакете Java, можно задать следующее свойство:

package.restrict.definition.java=true По умолчанию значение этого свойства равно true для пакетов java.*, sun.* и netscape.*.

property.applet Когда свойству с таким именем в Java I.I присваивается значение true, это значит, что апплету разрешается читать свойство под именем property из списка системных свойств. По умолчанию апплетам можно читать лишь десять стандартных системных свойств (их список находится в [главе 12](#)). Например, чтобы разрешить апплету читать свойство user.home, необходимо указать это свойство в следующем виде:

user.home.applet=true

1.8. Прoxy-серверы

Свойства, перечисленные ниже, определяют работу программы appletviewer с брандмауэрами и кэширующими Прoxy-серверами.

firewallHost Определяет брандмауэр, с которым надо связываться, если свойство firewallSet имеет значение true.

firewallPort Определяет порт брандмауэра, с которым надо связываться, если свойство firewallSet имеет значение true.

firewallSet Сообщает, должна ли программа просмотра апплетов использовать брандмауэр. Может иметь значения true и false.

proxyHost Определяет кэширующий Прoxy-сервер, с которым надо связываться, если свойство proxySet установлено в true.

proxyPort Определяет порт кэширующего Прoxy-сервера, с которым нужно связаться, если свойство proxySet установлено в true.

proxySet Сообщает, должна ли программа просмотра апплетов использовать кэширующий Прoxy-сервер. Может иметь значения true и false.

1.9. Переменные среды

CLASSPATH Содержит упорядоченный список (с двоеточиями в качестве разделителей для UNIX и с точками с запятой — для Windows) каталогов и файлов с расширением `zip`, в которых `appletviewer` должен искать определения классов. Когда путь указан с помощью переменной среды, `appletviewer` всегда неявно добавляет место расположения системных классов к концу пути. Если эта переменная не указана, по умолчанию список содержит текущий каталог и путь к системным классам. Отметим, что `appletviewer` поддерживает аргумент командной строки `-classpath`, за исключением его косвенной поддержки через опцию `-J`.

1.10. Смотри также

`java`, `javac`, `jdb`

2. jar — программа создания архивов Java

2.1. Доступность

JDK версии 1.1 и более поздних версий.

2.2. Синтаксис вызова

```
jar c|t|x[f][m][v] [jar-файл] [файл описания] [файлы]
```

2.3. Описание

Программа `jar` используется для создания архивных файлов Java (JAR) и работы с ними. JAR-файл представляет собой сжатый ZIP-файл с дополнительным файлом описания. Синтаксис команды `jar` напоминает синтаксис команды `tar` (tape archive — архив на магнитной ленте) ОС UNIX.

Параметры командной строки `jar` задаются в виде блока записанных слитно букв, которые передаются в виде одного аргумента, а не через отдельные аргументы командной строки. Первая буква такого аргумента задает необходимое действие, которое должна выполнить программа `jar`. Остальные буквы в этом аргументе являются необязательными. Различные аргументы файлов зависят от того, какие буквы параметров заданы.

2.4. Параметры

Первым аргументом командной строки `jar` является набор символов, задающих операцию, которая должна быть выполнена. Первый символ определяет основную операцию и является обязательным. Возможны следующие варианты:

c Создать новый JAR-архив. В качестве последних аргументов командной строки `jar` необходимо указать список файлов и/или каталогов.

t Вывести список файлов, содержащихся в JAR-архиве. Если задано имя JAR-файла с помощью параметра `f`, то список файлов выводится для него. В противном случае имя JAR-файла читается со стандартного устройства ввода.

x Извлечь содержимое JAR-архива. Если задано имя JAR-файла с помощью параметра `f`, то извлекается содержимое этого файла. В противном случае имя JAR-файла читается со стандартного устройства ввода. Когда командная строка завершается списком файлов и/или каталогов, из JAR-архива извлекаются только файлы и каталоги, перечисленные в этом списке. В противном случае из архива извлекаются все файлы.

Вслед за идентификатором, определяющим выполняемое действие, могут следовать необязательные параметры:

f Указывает на то, что имя JAR-файла, который необходимо создать, из которого нужно извлечь файлы или получить список содержащихся файлов, задается в командной строке. Если `f` используется вместе с `c`, `t` или `x`, имя JAR-файла должно задаваться в качестве второго аргумента командной строки вызова `jar` (т.е. оно должно располагаться непосредственно за блоком параметров). Когда этот параметр не задан, `jar` записывает создаваемый JAR-файл в стандартное устройство вывода или читает его со стандартного устройства ввода.

m Используется только в сочетании с параметром `c` и указывает на то, что `jar` должна читать файл описания, указанный в командной строке и использовать его в качестве основы для создания описания, которое включается в JAR-файл. Когда этот параметр задается после параметра `f`, имя файла описания должно указываться после имени создаваемого архива. Если `m` стоит перед параметром `f`, то имя файла описания должно предшествовать имени файла создаваемого архива.

`v` Описание. Если этот параметр задается вместе с параметром `c`, то `jar` выводит имя каждого добавляемого в архив файла со статистикой его сжатия. Когда параметр используется в сочетании с `t`, `jar` выводит список файлов, в котором кроме имени файла содержится его объем и дата последнего изменения. Если `v` указывается одновременно с `x`, то `jar` выводит имя каждого извлекаемого из архива файла.

2.5. Примеры

Создание простого JAR-архива:

```
% jar cvf my.jar *.java images
```

Получение списка содержимого архива:

```
% jar tvf your.jar
```

Извлечение файла описания из JAR-файла:

```
% jar xf the.jar META-INF/MANIFEST.MF
```

Создание JAR-файла с заданным описанием:

```
% jar cfmv YesNoDialog.jar manifest.stub oreilly/beans/yesno
```

2.6. Смотри также

`javakey`

3. Java — интерпретатор Java

3.1. Доступность

JDK версии 1.0 и более поздних версий.

3.2. Синтаксис вызова

```
Java [опции интерпретатора] имя класса [аргументы программы]  
Javag [опции интерпретатора] имя класса [аргументы программы]
```

3.3. Описание

Программа `Java` представляет собой интерпретатор байт-кода `Java`, который

запускает Java-программы. Программа `java_g` — это версия интерпретатора с возможностью отладки. Она не оптимизирована и обладает дополнительными средствами для отслеживания процесса выполнения программы.

Программа, которую надо выполнить, — это класс, указанный в аргументе `имя_класса`. Имя должно быть полным и включать имя пакета, но не содержать расширение `class`. Отметим, что имена пакета и класса разделяются точками, а не символами косой или обратной косой черты, как при указании пути. Если в классе Java отсутствует оператор `package`, значит, этот класс не принадлежит ни одному пакету, и указывается только его имя. Например:

```
%java david.games.Checkers %java test
```

В описании параметра `-classpath` и переменной среды `classpath` показано, где интерпретатор Java должен искать классы. Класс, указанный с помощью аргумента `имя_класса`, должен содержать метод `main ()` с таким объявлением:

```
public static void main(String argv[])
```

Любые аргументы, следующие за именем класса в командной строке Java, помещаются в массив и передаются методу `main ()` при запуске Java. Если в методе `main ()` создаются какие-либо потоки, то Java выполняется до тех пор, пока не завершится последний поток. В противном случае интерпретатор выполняет тело `main ()` и завершает работу. Хотя при вызове Java указывается имя только одного класса, интерпретатор автоматически загружает все дополнительные классы, необходимые для выполнения программы. Эти файлы классов задаются относительно пути для классов Java, который определяется параметром `-classpath`, описанным ниже.

По умолчанию Java запускает верификатор байт-кода во всех загруженных по сети классах. Такой верификатор выполняет ряд проверок байт-кода загруженного класса, чтобы убедиться, например, в том, что он не разрушает внутренний стек операндов и осуществляет соответствующие проверки, к примеру, ссылок на массивы во время выполнения. Параметры `-verify`, `-nonverify` и `-verifyremote` управляют процессом верификации байт-кода.

3.4. Параметры

-classpath путь Определяет путь, который Java использует для поиска указанного имени класса и всех остальных загружаемых им классов. Указание этого параметра переопределяет путь, заданный по умолчанию, а также переменную среды classpath.

Путь — это упорядоченный список каталогов и ZIP-файлов, в которых Java ищет названные классы. В среде Windows каталоги и ZIP-файлы (в них могут присутствовать спецификаторы дисков с использованием двоеточия) отделены друг от друга точками с запятой, а в среде UNIX — двоеточиями. Например, запись -classpath для UNIX выглядит таким образом:

```
-classpath/usr/lib/java/classes:~/Java/classes
```

А в среде Windows она имеет следующий вид:

```
-classpath C:\tools\java\classes.zip;.;D:\users\david\classes
```

Точка в спецификации пути показывает, что поиск ведется в текущем рабочем каталоге. Каталоги и ZIP-файлы просматриваются в порядке их упоминания в списке. Размещайте стандартные классы Java первыми в строке пути, если вы не хотите, чтобы их случайно перекрыли одноименные классы из других каталогов.

Интерпретатор Java предполагает найти файл класса в иерархии каталогов (или в имени каталога внутри ZIP-файла), в соответствии с его полным именем. Таким образом, в среде UNIX Java загрузит класс `java.lang.String` из файла `java/lang/String.class`, расположенного в одном из каталогов, которые указаны в пути поиска класса. Аналогично в Windows 95 или Windows NT (которые поддерживают длинные имена) Java будет искать файл `java\lang\String.class` в одном из заданных каталогов или внутри указанного ZIP-файла.

Если не задан ни аргумент -classpath, ни переменная среды classpath, путь поиска класса по умолчанию следующий:

```
.:$JAVA/classes:$JAVA/lib/classes.zip в среде UNIX
```

```
.:$JAVA\classes:$JAVA\lib\classes.zip в среде Windows
```

где \$java — каталог, в котором установлен JDK.

-cs,-checksource Оба параметра указывают интерпретатору Java на необходимость проверки времени модификации указанного файла класса и соответствующего ему файла с исходным текстом. Если файл класса не найден или устарел, он автоматически перекомпилируется из исходного.

-Dимя_свойства=значение Присваивает свойству из списка системных свойств значение, равное указанному. Java-программа после этого может искать указанное свойство по его имени. Существует возможность задать любое количество параметров -D. Например:

```
%java -Dawt.button.color=gray -Dmy.class.pointsize=14 my.class
```

-debug Заставляет Java выводить при запуске пароль, который используется для того, чтобы разрешать отладчику jdb участвовать в данном сеансе работы интерпретатора. Заметим, что этот пароль не считается безопасным с точки зрения шифрования информации.

-help Выводит сообщение о формате вызова данной программы.

-1цифра Задаёт уровень ведения протокола трассировки. Применяется только для программы java_g.

-ms начальная_память[k | m] Указывает объём памяти, который выделяется под динамически распределяемую область памяти, или кучу (heap), при запуске интерпретатора. По умолчанию параметр начальная_память задается в байтах. Данное значение можно указать и в килобайтах, добавив опцию k, или в мегабайтах посредством опции m. По умолчанию выделяется 1 Мб. При запуске крупных или интенсивно использующих память приложений (например, компилятора Java) производительность такой программы можно увеличить, попросив интерпретатор выделить больше памяти при запуске. Начальный объём этой памяти должен быть по крайней мере 1000 байтов.

-mx максимальная_память [k | m] Указывает максимальный объём динамически распределяемой области памяти, которую может выделять интерпретатор для хранения объектов и массивов. По умолчанию объём указывается в байтах, однако это значение можно задать и в килобайтах, добавив опцию k, а также в мегабайтах, используя опцию t. По умолчанию используется значение 16 Мб. Нельзя указывать

объем менее 1000 байтов.

-noasyncgc Не производит асинхронный сбор мусора. Если этот параметр указан, то Java производит сбор мусора только при нехватке памяти или при явном вызове сборщика мусора. Когда параметр не задан, Java запускает сборщик мусора как отдельный поток с низким приоритетом.

-noclassgc Не производит сбор мусора для загруженных классов, которые больше не используются. Этот параметр можно задавать только в JDK версии 1.1 и более поздних версий.

-noverify Никогда не проводит проверку байт-кода.

-oss размеры_стека[k | m] Устанавливает размер стека для кода каждого потока выполнения. По умолчанию параметр `размеры_стека` указывается в байтах. Но его можно задать в килобайтах, добавив опцию `k`, или в мегабайтах с помощью опции `m`. По умолчанию используется значение 400 Кб. Размер стека не должен быть меньше 1000 байтов.

-prof[:файл] Выводит информацию протоколирования в указанный файл или в файл `java.prof` в текущем каталоге. Формат этой информации не достаточно полно документирован. До появления JDK 1.1 протокол всегда выводился в файл `/java.prof` и указать другой файл было невозможно.

-ss размер_стека[k | m] Устанавливает размер стека для каждого потока выполнения. По умолчанию указывается в байтах. Но можно задать и в килобайтах (опция `k`), и в мегабайтах (опция `m`). По умолчанию составляет 128 Кб. Размер стека не должен быть менее 1000 байтов.

-t Выводит трассировку для всех выполняемых байт-кодов. Применимо только для `java_g`,

-tm Выводит трассировку для всех выполняемых методов. Применимо только для `java_g`.

-v, -verbose Выводит на экран сообщение всякий раз, когда Java загружает класс.

-verbosegc Выводит сообщение каждый раз, когда сборщик мусора освобождает память.

-verify Запускает верификатор байт-кода для всех загруженных классов.

-verify remote Запускает верификатор байт-кода для всех классов, загруженных через загрузчик классов. (Обычно это классы, динамически загруженные из не пользующегося доверием источника.) Для Java данный параметр установлен по умолчанию.

-version Выводит версию интерпретатора Java и выходит из программы.

3.5. Переменные среды

CLASSPATH Задаёт упорядоченный список (с двоеточиями в качестве разделителей для UNIX и точками с запятой — для Windows) каталогов и файлов с расширением `zip`, в которых интерпретатор Java должен искать определения классов. Когда путь указан с помощью этой переменной среды, Java всегда неявно добавляет в его конец места расположения системных классов. Если данная переменная не задана, по умолчанию список содержит текущий каталог и путь к системным классам. Эта переменная переопределяется параметром `-classpath`. Более подробную информацию об указании пути можно найти в приведенном выше описании параметра `-classpath`.

3.6. Смотри также

`javac, jab`

4. javac — компилятор Java

4.1. Доступность

JDK версии 1.0 и более поздних версий.

4.2. Синтаксис вызова

```
javac [параметры] файлы
```

4.3. Описание

Программа `javac` компилирует исходные тексты Java (из файлов с расширением `java`)

в байт-код Java (в файлы с расширением `class`). Компилятор Java сам написан на языке Java.

Компилятор `javac` может передать в командной строке любое количество файлов исходных текстов Java, чьи имена должны иметь расширение `Java`. Компилятор `javac` генерирует отдельный файл для каждого класса, определенного в файле исходного текста, независимо от количества файлов исходного текста. Другими словами, однозначного соответствия между файлами исходных текстов и файлами классов может и не быть. Отметим также, что компилятор требует, чтобы в одном файле исходного текста был определен только один `public`-класс и чтобы имя файла (без расширения `Java`) совпадало с именем класса (без имени пакета).

По умолчанию `javac` располагает созданные файлы классов в том же каталоге, где находились соответствующие файлы с исходными текстами. Это можно переопределить с помощью параметра `-d`.

Когда файл исходного текста ссылается на класс, не определенный в командной строке какого-либо другого файла исходного текста, `javac` ищет этот класс, используя параметр `-classpath`. По умолчанию данный параметр содержит только текущий каталог и системные классы. Можно указать дополнительные классы и пакеты, где следует искать класс, используя параметр `-classpath` или переменную среды `classpath`.

4.4. Параметры

`-classpath` путь Определяет путь, используемый `javac` для поиска классов, на которые имеются ссылки в исходных текстах. Данный параметр переопределяет путь по умолчанию и любой путь, заданный переменной среды `classpath`. Аргумент `путь` представляет собой упорядоченный список каталогов и ZIP-файлов, разделителями в котором служат двоеточия в среде UNIX и точки с запятой в среде Windows.

Для определения дополнительных каталогов и ZIP-файлов, без переопределения пути по умолчанию, используют переменную среды `classpath`. Более подробно об этом рассказано в описании Java.

`-d` каталог Определяет каталог, в котором должны храниться файлы классов. По умолчанию `javac` помещает созданные им файлы с расширением `class` в те же каталоги, что и файлы с расширением `Java`, из которых они откомпилированы.

Однако если определен флаг `-d`, то указанный каталог рассматривается в качестве корневого в иерархии классов, и файлы с расширением `class` помещаются в этот каталог или в его соответствующий подкаталог в зависимости от имени пакета класса. Поэтому следующая команда:

```
%javac -d Java/classes java/src/Checkers.Java
```

помещает файл `Checkers.class` в каталог `Java/classes`, если у файла `Checkers.java` нет оператора `package`. С другой стороны, когда файл исходного текста указывает, что он находится в пакете:

```
package david.games;
```

файл с расширением `class` хранится в `java/classes/david/games`.

Если параметр `-d` указан, то `javac` автоматически создает нужный для хранения класса каталог в соответствующем месте.

-depend Сообщает `javac` о необходимости перекомпилировать любой встретившийся ему устаревший файл класса и не только из числа тех, на которые имеются ссылки в заданных файлах с исходными текстами.

-deprecation Сообщает `javac` о необходимости выдавать предупреждение при каждом использовании устаревшего API. По умолчанию `javac` генерирует такое предупреждение только один раз, в программе используются устаревшие API. Доступен в JDK версии 1.1 и более поздних версий.

-g Заставляет `javac` добавлять в файл класса информацию о номерах строк и локальных переменных для использования ее отладчиками. По умолчанию `javac` добавляет только номера строк, а при указании параметра `-o` не делает даже этого.

-Jаргумент Передает аргумент непосредственно интерпретатору Java (пробелы в аргументе должны отсутствовать). Если интерпретатору необходимо передать несколько аргументов, следует использовать несколько параметров `-J`. Доступен в JDK версии 1.1 и более поздних версий.

-nowarn Отключает вывод предупреждений. Сообщения об ошибках выводятся, как обычно.

-nowrite Сообщает `javac` о том, что не следует создавать файл класса. Файлы с исходными текстами обрабатываются обычным путем, но результаты такой обработки не записываются. Этот параметр полезен в том случае, когда требуется, не выполняя фактической компиляции, проверить, может ли компилироваться некоторый файл.

-O Разрешает оптимизировать файлы классов. Может заставить `javac` компилировать методы, объявленные как `static`, `final` и `private`, в машинные коды, обеспечивая более быстрое их выполнение. Это достигается ценой увеличения объемов файлов классов. Данный параметр также предотвращает добавление отладочной информации о номерах строк в файлы классов.

-verbose Дает компилятору указание выдавать сообщения о том, что он делает.

4.5. Переменные среды

`CLASSPATH` Содержит список (с двоеточиями в качестве разделителей для UNIX и с точками с запятой — для Windows) каталогов и файлов с расширением `zip`, в которых `javac` должна искать определения классов. Когда путь указан с помощью переменной среды, `javac` всегда неявно добавляет каталог расположения системных классов к концу пути. Если эта переменная среды не указана, то по умолчанию список будет содержать текущий каталог и путь к системным классам. Данная переменная переопределяется параметром `-classpath`.

4.6. Смотри также

java, jdb

5. javadoc — генератор документации Java

5.1. Доступность

JDK версии 1.0 и более поздних версий.

5.2. Синтаксис вызова

```
javadoc [параметры] имя пакета
```

```
javadoc [параметры] имена файлов
```

5.3. Описание

Программа `javadoc` создает документацию на API в формате HTML для указанных пакетов или отдельных файлов исходных текстов Java, заданных в командной строке.

Если в командной строке указано имя пакета, `javadoc` ищет соответствующий каталог пакета относительно параметра `-classpath`. Затем анализирует все файлы с исходными текстами в этом каталоге и формирует HTML-файл документации для каждого класса и HTML-индекс для классов пакета. По умолчанию HTML-файлы создаются в текущем каталоге. Изменить это положение позволяет параметр `-d`.

Отметим, что аргумент `имя_пакета` определяет имя пакета (компоненты, разделенные точками), а не имя каталога. Иногда возникает необходимость указать параметр `-sourcepath`, чтобы `javadoc` мог найти файлы с исходными текстами пакета, если они хранятся не там же, где файлы классов пакета.

Генератор `javadoc` может вызываться с любым числом файлов исходного текста на Java в командной строке. Заметим, что это имена файлов, а не классов, и они указываются с необходимыми компонентами каталогов и с расширением `Java`. Когда `javadoc` вызывается таким образом, он читает указанные файлы исходных текстов и создает HTML-файлы (по умолчанию в текущем каталоге), которые описывают каждый `public`-класс, определенный в указанных файлах.

Файлы документации классов, создаваемые `javadoc`, описывают класс (или интерфейс) и его иерархию наследования, индекс и каждый член класса, объявленный как `public` или `protected`. Созданный файл также содержит комментарии для документации, которые связаны с классами и их методами, конструкторами и переменными. Комментарий для документации — это комментарий Java, начинающийся символами `/**` и оканчивающийся символами `*/`. Он может включать любой HTML-тег (хотя не должен содержать структурных тегов типа `<H1>` или `<hr>`), а также значения тегов, которые обрабатываются `javadoc` специальным образом.

5.4. Параметры

-author путь Указывает на необходимость вывода информации об авторе, заданной с

помощью тега `@author`. Эта информация по умолчанию не выводится.

-classpath путь Определяет путь, который `javadoc` использует для поиска, как файлов классов, так и файлов с исходными текстами для указанного пакета. Чтобы указать `javadoc`, где искать файлы с исходными текстами, при определении этого параметра следует не забыть включить в него стандартный системный путь поиска каталога с классами, иначе `javadoc` не сможет найти нужные классы. Этот параметр переопределяет принимаемый по умолчанию или заданный переменной среды `classpath` путь, который представляет собой список каталогов и ZIP-файлов, где в качестве разделителей применяются двоеточия (в среде UNIX) или точки с запятой (в среде Windows).

Для указания дополнительных каталогов и ZIP-файлов без переопределения пути по умолчанию используется переменная среды `classpath`. Подробнее об определении пути рассказано в разделе, посвященном Java .

-d каталог Каталог, в который `javadoc` должен записывать создаваемые им HTML-файлы. По умолчанию это текущий каталог.

-decoding название_кодировки Задаёт кодировку символов, которая будет использоваться в документации, создаваемой с помощью `javadoc`. Используется в Java версии 1.1 и более поздних версий.

-encoding название_кодировки Задаёт кодировку символов, которая будет использоваться при чтении файлов с исходными текстами и комментариями, содержащимися в них. Доступен в Java версии 1.1 и более поздних версий.

-Jаргумент Передаёт аргумент непосредственно интерпретатору Java (аргумент не должен содержать пробелы). Если в интерпретатор требуется передать несколько аргументов, следует использовать несколько параметров `-J`. Доступен в Java версии 1.1 и более поздних версий.

-nodeprecated Сообщает `javadoc` о том, что теги `@deprecated` в создаваемые документы включать не нужно (установка по умолчанию). Доступен в Java версии 1.1 и более поздних версий.

-noindex Сообщает `javadoc` о том, что формировать индексный файл `All-Names.html`, который она создает по умолчанию, не требуется.

-notree Сообщает javadoc о том, что формировать файл иерархии классов `tree.html`, который она создает по умолчанию, не нужно.

-sourcepath каталог Синоним `-classpath`. Отметим, что любой задаваемый путь должен включать в себя системный путь поиска каталога классов.

-verbose Заставляет javadoc выдавать сообщения о том, что он делает в данный момент.

-version каталог Указывает на необходимость выводить информацию о версии, заданную с помощью тега `@ version`. Эта информация не выводится по умолчанию. Отметим, что данный параметр не заставляет javadoc выводить номер собственной версии.

5.5. Переменные среды

CLASSPATH Задаёт упорядоченный список (с двоеточиями в качестве разделителей для UNIX и с точками с запятой — для Windows) каталогов и файлов с расширением `zip`, в которых javadoc должна искать определения классов. Когда путь указан с помощью переменной среды, javadoc всегда неявно добавляет место расположения системных классов к концу пути. Если эта переменная не указана, то по умолчанию список содержит текущий каталог и путь к системным классам. Данная переменная переопределяется параметром `-classpath`.

5.6. Ошибки

Когда javadoc не может найти указанный пакет, она создает файл-заглушку HTML и не предупреждает, что пакет не найден.

5.7. Смотри также

java, javac

6. javah — генератор C-файлов

6.1. Доступность

JDK версии 1.0 и более поздних версий.

6.2. Синтаксис вызова

```
javah [параметры] имена_классов
```

6.3. Описание

Программа `javah` создает файлы заголовков и исходных текстов на C (файлы с расширением `h` и `c`), которые описывают указанные классы. Отметим, что классы задаются с помощью имен классов, а не имен файлов. Генерируемые файлы несут информацию, необходимую для реализации методов указанных классов на C способом, зависящим от платформы. По умолчанию `javah` создает файлы, которые могут использоваться платформ-нозависимым интерфейсом JDK 1.0. Если задан параметр `-jni`, программа генерирует файлы, предназначенные для использования платформно-зависимым интерфейсом Java Native Interface (JNI) в Java 1.1.

По умолчанию `javah` создает файл заголовков для указанного класса или классов. В этом файле объявляется C-структура `struct`, которая содержит переменные, соответствующие переменным экземпляра класса Java. В нем также объявляется процедура, которую необходимо реализовать для каждого платформно-зависимого метода, содержащегося в Java-классе. (Полное описание способов реализации методов Java на C выходит за рамки данной главы.)

Если `javah` выполняется с параметром `-stubs`, создается файл с расширением `c`, содержащий дополнительные процедуры-заглушки, необходимые для связи платформно-зависимого метода со средой Java. Отметим, что в этот файл не следует помещать текст реализации метода.

При заданном параметре `-jni` программа `javah` создает файлы заголовков на C, в которых объявляются прототипы для платформно-зависимой реализации каждого метода указанных классов. При использовании этого нового интерфейса определять какую-либо структуру не требуется. Для JNI не нужны также файлы-заглушки, поэтому параметр `-stub` не может использоваться в сочетании с параметром `-jni`.

По умолчанию `javah` создает C-файлы в текущем каталоге, и их имена включают имя класса. Если имя класса включает имя пакета, то C-файлы включают все компоненты

полного имени класса с заменой точек символом подчеркивания. Этот заданный по умолчанию режим может быть переопределен с помощью параметров `-d` и `-o`.

6.4. Параметры

-classpath путь

Определяет путь, который `javah` использует для поиска классов, указанных в командной строке. Этот параметр переопределяет путь, задаваемый по умолчанию или определяемый переменной среды `classpath`. Аргумент путь содержит упорядоченный список каталогов и ZIP-файлов, разделенных двоеточиями в среде UNIX или точками с запятой в среде Windows.

Чтобы указать дополнительные каталоги и ZIP-файлы, не переопределяя заданный по умолчанию путь поиска системных классов, используется переменная среды `classpath`. Подробнее об определении пути рассказано в описании Java .

-d каталог Указывает каталоги, в которые `javah` должен записывать создаваемые им файлы. По умолчанию они создаются в текущем каталоге. Этот параметр не работает вместе с параметром `-o`, поэтому внутри имени файла в параметре `-o` следует также указывать необходимый каталог.

-help Заставляет `javah` вывести простое сообщение о формате вызова и завершить работу.

-jni Указывает `javah` на необходимость создания файла заголовков, предназначенного для работы с новым интерфейсом JNI (Java Native Interface), а не со старым платформно-зависимым интерфейсом Java 1.0. Доступен в JDK версии 1.1 или более поздних версий.

-o выходной_файл Объединяет все файлы с расширением `s` и `h` в один выходной файл. Это удобно при реализации зависящих от платформы методов для нескольких классов одного пакета, а также позволяет избежать наличия большого количества маленьких файлов с расширениями `h` и `s`, с которыми приходится работать по отдельности.

-stubs Создает файлы с расширением `s` для класса или классов, но не файлы заголовков. Без этого параметра `javah` создает файлы заголовков.

-td каталог Каталог, в котором javah должна хранить временные файлы. По умолчанию временные файлы хранятся в каталоге /tmp.

-trace Указывает на необходимость для javah включать команды вывода информации о трассировке в создаваемые ею файлы-заглушки.

-v Побуждает javah выводить сообщения о выполняемых операциях.

-version Заставляет javah выводить информацию о номере своей версии.

6.5. Переменные среды

CLASSPATH Содержит упорядоченный список (с двоеточиями в качестве разделителей для UNIX и с точками с запятой — для Windows) каталогов и файлов с расширением zip, в которых javah должна искать определения классов. Когда путь указан с помощью переменной среды, javah всегда неявно добавляет место расположения системных классов к концу пути. Если эта переменная не указана, то по умолчанию список содержит текущий каталог и путь к системным классам. Данная переменная переопределяется параметром -classpath.

6.6. Смотри также

java, javac

7. javakey — программа управления ключами и цифровыми подписями

7.1. Доступность

JDK версии 1.1 и более поздних версий.

7.2. Синтаксис вызова

javakey параметры

7.3. Описание

Программа `javakey` обеспечивает интерфейс командной строки для построения и работы с рядом сложных ключей и сертификатов, включая создание цифровых подписей. Существует всего несколько параметров, обеспечивающих выполнение ряда строго определенных операций. Программа `javakey` работает с системной базой данных, для каждой записи которой могут быть заданы открытые и секретные ключи и/или сертификаты, связанные с ней. Кроме того, каждая из этих записей может быть объявлена вызывающей доверие, или наоборот, а также может быть либо идентификатором (`identity`), либо подписчиком (`signer`). С идентификаторами связывают только открытые ключи, тогда как для подписчиков существуют как открытые, так и секретные ключи, поэтому с помощью подписчиков можно создавать подписи для файлов.

Операции `javakey` задаются посредством различных параметров, приведенных ниже.

7.4. Параметры

-с название_идентификатора [true | false] Создает и добавляет идентификатор с заданным именем в базу данных. Если после имени идентификатора следует `true`, он объявляется вызывающим доверие. В противном случае идентификатор считается не вызывающим доверия.

-с имя_подписчика [true | false] Создает и добавляет подписчик с заданным именем в базу данных. Если после имени подписчика следует `true`, подписчик объявляется вызывающим доверие. В противном случае он не считается таковым.

-t название_записи [true | false] Указывает, является ли заданная запись вызывающей доверие (`true`) или нет (`false`).

-l Выводит список имен всех записей в базе данных системы безопасности.

-ld Выводит подробный список имен и прочей информации для записей в базе данных системы безопасности.

-li название_записи Выводит подробную информацию о записи с указанным именем из базы данных системы безопасности.

-r название_записи Удаляет запись с указанным именем из базы данных системы безопасности.

-ik название идентификатора файл_ключа Читает открытый ключ из указанного файла и связывает его с заданным идентификатором. Ключ должен задаваться в формате X.509.

-ikr имя_подписчика файл_открытого_ключа файл_секретного_ключа Читает заданные открытый и секретный ключи из соответствующих файлов и связывает их с записью для именованного подписчика. Ключи должны задаваться в формате x.509.

-is название_записи файл_сертификата Читает сертификат из именованного файла и связывает его с указанной записью. Если для этой записи открытый ключ уже существует, то он сравнивается с ключом, указанным в сертификате, и в случае, когда они не совпадают, выводится соответствующее предупреждение. Когда открытый ключ для записи не задан, используется соответствующий ключ из сертификата.

-ii название_записи Эта команда позволяет ввести произвольную текстовую информацию об указанной записи в базу данных.

-gk подписчик алгоритм размер [файл_открытого_ключа [файл_секретного_ключа]] Создает открытый и секретный ключи и связывает их с указанным подписчиком, используя заданный алгоритм. В настоящее время поддерживается только алгоритм DSA. Создает ключи с заданным количеством битов (значение количества битов должно находиться в пределах от 512 до 1024). Если указан файл открытого ключа, в него записывается открытый ключ. Когда указан файл секретного ключа, туда записывается секретный ключ.

-g подписчик алгоритм размер [файл_открытого_ключа [файл_секретного_ключа]] Синоним команды -gk.

-gs командный_файл Создает сертификат в соответствии с параметрами, заданными в командном файле. Командный файл — это файл с расширением `properties`, где должны содержаться значения следующих свойств:

- `issuer . name` — имя записи, для которой создается сертификат.
- `issuer.cert` — номер сертификата создателя, который следует использовать в качестве подписи для нового сертификата (если сертификат не создает эту подпись сам).
- `subject. name` — имя записи в базе данных, для которой создается сертификат.
- `subject. real. name` — настоящее имя записи, для которой создается сертификат.
- `subject. country` — страна, к которой относится данная запись.
- `subject.org` — организация, с которой связана данная запись.
- `subject.org.unit` — подразделение организации, с которой связана данная запись.
- `start. date` — дата (и время) вступления в силу данного сертификата.
- `end. date` — дата (и время) окончания действия данного сертификата.
- `serial, number` — серийный номер сертификата (он должен быть уникальным для каждого сертификата, создаваемого данной организацией).
- `out .file` — необязательное имя файла, в который должен записываться данный сертификат.

-dc файл_сертификата Выводит содержимое сертификата, записанного в файле сертификата.

-ec запись номер _сертификата файл Выводит заданный с помощью номера сертификат для определенной записи в указанный файл. С помощью команды `-li` можно проверить, какие номера сертификатов относятся к заданной записи.

-ek запись файл_открытого_ключа [файл_секретного_ключа] Выводит открытый ключ для заданной записи в указанный файл. Если эта запись соответствует подписчику и указан файл секретного ключа, то для заданной записи в этот файл дополнительно экспортируется секретный ключ.

-gs командный_файл jar-файл Создает цифровую подпись для заданного JAR-файла с использованием директив, приведенных в указанном командном файле. Командный файл представляет собой файл с расширением `properties`, в котором должны содержаться значения следующих свойств:

- `signer`—имя записи для подписчика.
- `cert`—номер сертификата, который должен использоваться в подписи.
- `chain`—длина цепочки сертификатов, которые необходимо включить. В настоящее время этот параметр не поддерживается; его следует задавать равным 0.

- `signature.file`—базовое имя файла подписи, вставляемой в JAR-файл. Его длина не должна превышать 8 символов и имя не должно конфликтовать ни с одной другой цифровой подписью, которая может вводиться в JAR-файл.
- `out.file`—задает имя, которое должно использоваться для создаваемого JAR-файла с цифровой подписью. Свойство является необязательным.

7.5. Смотри также

`jar`

8. `javap` — дизассемблер классов Java

8.1. Доступность

JDK версии 1.0 и более поздних версий.

8.2. Синтаксис вызова

```
javap [параметры] имена_классов
```

8.3. Описание

Программа `javap` дизассемблирует файлы классов, имена которых указаны в командной строке, и выводит их тексты в доступном для чтения виде.

По умолчанию `javap` выводит объявления членов (не объявленных как `private`) классов, указанных в командной строке. Параметры `-l`, `-r` и `-c` задают дополнительную информацию, которую следует выводить, включая результаты полного дизассемблирования байт-кода каждого из указанных классов. Программу `javap` можно также использовать для запуска верификатора классов Java.

8.4. Параметры

-c Выводит инструкции виртуальной машины Java для всех методов указанного класса. Дизассемблирует все методы, включая `private`.

-classpath путь Путь, который `javap` использует для поиска классов, указанных в

командной строке. Этот параметр переопределяет путь, заданный по умолчанию или указанный в переменной среды `classpath`. Аргумент `путь` — это упорядоченный список каталогов и ZIP-файлов, разделяемых двоеточиями в среде UNIX и точками с запятой в среде Windows.

Чтобы указать дополнительные каталоги и ZIP-файлы, не переопределяя пути, заданного по умолчанию, используется переменная среды `classpath`. Подробнее о ней рассказано в описании Java .

-h Выводит класс в форме, пригодной для включения его в файл заголовков C.

-l Выводит номера строк и таблицы локальных переменных в дополнение к переменным `public`-класса. Отметим, что информация о номерах строк и локальных переменных используется отладчиками. Информация о локальных переменных присутствует только в том случае, если класс откомпилирован `javac` с параметром `-d`; информация о номерах строк присутствует, если класс откомпилирован без параметра `-o`.

-p Помимо методов и переменных, объявленных как `public`, выводит методы и переменные указанных классов, объявленные как `private`. Отметим, что некоторые компиляторы (но не `javac`) могут исказить информацию о `private`-переменных, так что такие переменные и аргументы методов перестают иметь значащие имена. Это затрудняет дизассемблирование классов Java и восстановление исходного текста.

-s Выводит объявления членов класса во внутреннем формате виртуальной машины Java.

-v Выводит дополнительную информацию (в виде компонентов Java) обо всех членах каждого указанного класса.

-verify Заставляет `javap` запускать верификатор для заданных классов и выводить результаты их проверки.

-version Заставляет `javap` выводить информацию о номере своей версии.

8.5. Переменные среды

CLASSPATH Содержит упорядоченный список (с двоеточиями в качестве

разделителей для UNIX и с точками с запятой—для Windows) каталогов и файлов с расширением `zip`, в которых `javap` должна искать определения классов. Когда путь задан с помощью переменной среды, `javap` всегда неявно добавляет место расположения системных классов к концу пути. Если эта переменная не указана, то по умолчанию список содержит текущий каталог и путь к системным классам. Данная переменная переопределяется параметром `-classpath`.

8.6. Смотри также

`java`, `javac`

9. jdb — отладчик Java

Доступность JDK версии 1.0 и более поздних версий.

9.1. Синтаксис вызова

```
jdb [ параметры Java ] класс  
jdb [ -host имя сервера ] -password пароль
```

9.2. Описание

Программа `jdb` — отладчик классов Java. Она работает в текстовом режиме, ориентирована на командную строку и синтаксис ее вызова аналогичен синтаксису отладчиков UNIX `dbx` и `gdb`.

Когда `jdb` вызывается с именем класса Java, она запускает другую копию интерпретатора Java, передавая ему указанные параметры. Отладчик `jdb` сам является Java-программой, которая выполняется собственной копией интерпретатора. Эта новая копия загружает указанный файл класса и прерывает его выполнение, ожидая ввода команд отладки.

Отладчик `jdb` может запускаться с аргументами `-password` и `-host`. При таком вызове `jdb` подключается к уже выполняющейся копии интерпретатора, запущенного с параметром `-debug`. Когда интерпретатор запущен таким способом, он выводит пароль, который следует указать в параметре `-password` отладчика `jdb`.

После запуска сеанса отладки, можно выполнять любую из директив, описанных ниже.

9.3. Параметры

При вызове `jdb` с заданным именем файла класса можно указать любой параметр интерпретатора Java. Назначение этих параметров объяснялось в описании Java.

При подключении `jdb` к уже работающему интерпретатору Java можно задавать следующие параметры:

-host имя_сервера Указывает имя узла, на котором запущен нужный сеанс интерпретатора.

-password пароль Этот параметр необходим для присоединения к работающему интерпретатору. Интерпретатор должен быть запущен с параметром `-debug`, и параметр `-password` указывает пароль, который сообщает интерпретатор. К интерпретатору может присоединиться только отладчик, знающий этот пароль. Отметим, что созданный Java пароль нельзя считать безопасным с точки зрения шифрования.

9.4. Команды

Программа `jdb` понимает следующие команды:

!! Сокращенная команда, которая заменяется текстом последней введенной команды. За ней может следовать дополнительный текст, добавляемый к предшествовавшей команде.

catch [класс_исключения] Прерывает выполнение программы, если сгенерировано указанное исключение. Когда исключение не указано, выводится текущий список перехваченных исключений. Для отмены прерывания используйте команду `ignore`.

classes Выводит список всех загруженных классов.

clear [класс : строка] Удаляет точку останова, установленную в указанной строке данного класса. Команды `clear` и `stop` без аргументов выводят список текущих точек останова вместе с номерами строк, в которых они заданы.

cont Возобновляет выполнение программы и используется при остановке текущего потока выполнения в точке останова.

down [n] Перемещается на n кадров вниз в стеке вызовов текущего потока. Если n не указано, перемещается вниз на один кадр.

dump id (s) Выводит значения всех переменных указанного объекта (объектов). Если задано имя класса, команда dump отображает все (статические) методы и переменные класса, а также имя суперкласса и список реализуемых интерфейсов. Объекты и классы можно задать по имени или по их восьмизначному шестнадцатеричному ID-номеру, а потоки — по сокращенному имени t@номер_потока.

exit (или quit) Выход из jdb.

gc Запускает сборщик мусора для удаления неиспользуемых объектов.

help (или ?) Выводит список всех директив jdb.

ignore класс_исключения Генерирование указанного исключения не приводит к прерыванию выполнения программы. Эта команда отключает команду catch.

list [номер_строки] Выводит указанную строку исходного текста и несколько строк, стоящих перед ней и позади нее. Если номер строки не указан, использует номер строки текущего кадра стека для текущего потока. При этом выводятся строки из файла с исходным текстом для текущего кадра стека текущего потока. Команда use сообщает jdb, где искать файл с исходным текстом.

load имя_класса Загружает указанный класс в jdb.

locals Выводит список локальных переменных для текущего кадра стека. Код на Java должен компилироваться с параметром -d, чтобы включать информацию о локальных переменных.

memory Выводит суммарный объем памяти, которую занимает отлаживаемая программа.

methods класс Выводит список всех методов указанного класса. Для получения списка переменных экземпляра, объекта или класса (статического) используется команда dump.

print id(s) Выводит значение указанного аргумента или аргументов. Каждый аргумент может представлять собой класс, объект, переменную или локальную переменную и может быть задан посредством их имени или шестнадцатеричного ГО-номера. Кроме того, существует возможность ссылаться на потоки при помощи специального синтаксиса `t@номер_потока`. Команда `print` отображает значение объекта, вызывая его метод `toString()`.

resume [поток (и)] Возобновляет выполнение указанного потока (потоков). Если поток не указан, возобновляется выполнение всех прерванных потоков (см. также `suspend`).

run [класс] [аргументы] Выполняет метод `main()` данного класса, передавая ему указанные аргументы. Если класс или аргументы не заданы, используются класс и аргументы из командной строки `jdkb`.

step Выполняет текущую строку текущего потока и снова останавливает выполнение.

stop [at класс : строка] stop [in класс : метод] Устанавливают точку останова в указанной строке класса или в начале указанного метода класса. Выполнение программы останавливается при достижении этой строки или при вызове данного метода. Если команда `stop` выполняется без аргументов, то выводится текущий список точек останова.

suspend [поток (и)] Приостанавливает указанный поток или потоки. Если поток не указан, останавливает все выполняющиеся потоки. Для их повторного запуска используется команда `resume`.

thread поток Устанавливает указанный поток выполнения в качестве текущего. Этот поток неявно используется несколькими другими командами `./^/`. Поток может быть задан по имени или номеру.

threadgroup имя Устанавливает указанную группу потоков в качестве текущей.

threadgroups Выводит список всех групп потоков в отлаживаемой сессии работы интерпретатора Java.

threads [группа _потоков] Выводит список всех потоков указанной группы. Если группа не задана, выводятся потоки текущей группы (заданной с помощью

параметра группа_потоков) .

up [n] Перемещается на n кадров вверх по стеку вызовов текущего потока. Если n не указано, перемещается вверх только на один кадр.

use [путь_к_исходному_файлу] Задаёт путь, используемый jdb для поиска файлов с исходными текстами отлаживаемых классов. Если имя не указано, отображает текущее значение.

where[поток][all] Отображает стек вызовов указанного потока. Если поток не указан, отображает стек вызовов текущего потока. Когда указана опция all, отображаются стеки вызовов всех потоков.

9.5. Переменные среды

CLASSPATH Указывает упорядоченный список (с двоеточиями в качестве разделительных знаков в UNIX, с точками с запятой — в Windows) каталогов и ZIP-файлов, в которых jdb должна искать определения классов. Если путь задан с помощью этой переменной, то jdb всегда неявно присоединяет местоположение системных классов к его концу. Когда данная переменная не задана, путь по умолчанию указывает на текущий каталог и на каталог системных классов. Эта переменная переопределяется параметром -classpath.

9.6. Смотри также

java

10. native2ascii — преобразование исходных текстов Java в ASCII

10.1. Описание

Программа преобразования исходных текстов Java в ASCII-текст

10.2. Доступность

JDK версии 1.1 и более поздних версий.

10.3. Синтаксис вызова

```
native2ascii [ параметры ] [ входной файл [ выходной файл ] ]
```

10.4. Описание

Программа `javac` может обрабатывать только файлы, в которых используется кодировка символов ASCII и кодировка Unicode `\uxxxx`. Программа `native2ascii` осуществляет чтение файла с исходным текстом на Java, закодированным с использованием местной кодировки символов, и преобразование этого текста в кодировку ASCII плюс кодировку Unicode, необходимую для `javac`.

Параметры `входной_файл` и `выходной_файл` являются необязательными. Если они не заданы, используются стандартные устройства ввода-вывода, что позволяет применять `native2ascii` для работы с каналами (pipes).

10.5. Параметры

-encoding имя_кодировки Задаёт кодировку, которая используется в исходном файле. Если параметр не задан, имя соответствующей кодировки извлекается из системного свойства `file.encoding`.

-reverse Указывает на то, что должно осуществляться обратное преобразование — символов, закодированных в виде `\uxxxx`, в символы в местной кодировке.

10.6. Смотри также

`java.io.InputStreamReader`, `java.io.OutputStreamWriter`

11. serialver — генератор номера версии класса

11.1. Доступность

JDK версии 1.1 и более поздних версий.

11.2. Синтаксис вызова

```
serialver [-show] имя класса...
```

11.3. Описание

Программа `serialver` выводит номер версии или уникальный идентификатор для сериализации указанного класса или классов. Если в классе объявлена константа `serialVersionUID` типа `long`, то выводится ее значение, в противном случае уникальный номер версии для API, определяемого в классе, вычисляется с помощью безопасного алгоритма шифрования SHA (Secure Hash Algorithm). Эта программа применяется, в первую очередь, для вычисления исходного уникального номера версии класса, который затем присваивается объявляемой в данном классе константе. Результатом работы программы `serialver` является строка правильного текста на Java, которую можно затем вставить в определение класса.

11.4. Параметры

-show Если параметр задан, `serialver` выводит простой графический интерфейс, который позволяет пользователю вводить каждый раз по одному имени класса и получать для него уникальный идентификатор для сериализации. При задании параметра `-show` имя класса в командной строке можно не указывать.

11.5. Переменные среды

CLASSPATH Программа `serialver` написана на Java, и поэтому она чувствительна к значению переменной среды `class path` точно так же, как и интерпретатор Java. Поиск указанных классов проводится относительно заданного пути для классов.

11.6. Смотри также

`java.io.ObjectStreamClass`