Комплект Java разработчика Liberica JDK 11. Руководство пользователя.

Руководство по использованию основных компонентов Liberica JDK 11

ООО ”БЕЛЛСОФТ”

2018-11-20

Contents

+++

Введение

Liberica JDK - это бинарный дистрибутив соответствующий Java SE спецификациям и базирующийся на проекте с открытым исходным кодом. Проект создан компанией BellSoft на базе OpenJDK, в развитии которого BellSoft принимает активное участие. Liberica тщательно протестирована, проходит JCK и поставляется под лицензией OpenJDK. Версии Liberica для Windows x86 / 64, Mac x86 / 64, Linux x86 / 64 и ARMv7 также содержат JavaFX 11.0.1. Версии для Linux ARMv7 содержат Device IO API в качестве

дополнительного модуля и JavaFX с поддержкой аппаратного ускорения EGL.

Руководство по использованию основных компонентов Liberica JDK

Jar

Средство архивирования Java. Собирает множества файлов в единый архивный файл Jar.

Синтаксис

Параметры командной строки jar задаются в виде блока записанных слитно букв, которые передаются одним аргументом, а не через отдельные аргументы командной строки. Первая буква такого аргумента задаёт необходимое действие, которое должна выполнить программа jar.

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

u (update) Обновить имеющийся JAR-архив. В качестве последних аргументов командной строки jar необходимо указать список файлов и/или каталогов.

t (table of contents view) 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 (extract) 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 (file) Указывает на то, что имя JAR-файла, который необходимо создать, из которого нужно извлечь файлы или получить список содержащихся файлов, задаётся в командной строке. Если f используется вместе с c, u, t или x, имя JAR-файла должно задаваться в качестве второго аргумента командной строки вызова y (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 (manifest) 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 задаётся 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 (verbose) Описание обрабатываемых файлов. Если этот параметр задаётся вместе с c или u, то выводится имя каждого добавляемого или обновляемого в архиве 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.

Примеры

Создание JAR-файла

jar c[v0M]f jarfile [-C *dir*] inputfiles [-[J]*option*]

jar c[v0]mf manifest jarfile [-C *dir*] inputfiles [-[J]*option*] [-e *entrypoint*]

jar c[v0M] [-C *dir*] inputfiles [-[J]*option*]

jar c[v0]m manifest [-C *dir*] inputfiles [-[J]*option*]

Обновление JAR-файла

jar u[v0M]f jarfile [-C *dir*] inputfiles [-[J]*option*]

jar u[v0]mf manifest jarfile [-C *dir*] inputfiles [-[J]*option*] [-e *entrypoint*]

jar u[v0M] [-C *dir*] inputfiles [-[J]*option*]

jar u[v0]m manifest [-C *dir*] inputfiles [-[J]*option*]

Распаковка JAR-файла

jar x[v]f jarfile [inputfiles] [-[J]*option*]

jar x[v] [inputfiles] [-[J]*option*]

Вывод содержимого JAR-файла

jar t[v]f jarfile [inputfiles] [-[J]*option*]

jar t[v] [inputfiles] [-[J]*option*]

Добавление индексов в JAR-файлы

jar i jarfile [-[J]*option*]

Параметры

Ниже указанные опции должны идти в том же порядке, как и параметры каждой опции в дальнейшем.

inputfiles

> Требуемые файлы или каталоги в команде разделяются пробелами при сжатии (c) или обновлении существующего архива (u), либо при извлечении (x) или получении списка файлов (t). Все каталоги обрабатываются рекурсивно. Файлы будут сжиматься, если не указан параметр -0 (ноль)

manifest

> Предварительно созданный файл манифеста MANIFEST.MF содержащий пары "ключ - значение" для дополнительных атрибутов (например, вендора ПО) добавляется в JAR-файл.

entrypoint

> Наименование класса, который назначается в качестве загрузчика приложений, реализованных в виде исполняемого JAR-файла.

-C *dir*

> Осуществляет временный (на период выполнения команды, аналог `cd dir`) переход в указанный каталог при обработке аргумента `inputfiles`. Допускается использовать несколько аргументов -C в одной команде.

-[J]*option*

> Передаёт опции загрузчика Java-приложений виртуальной машине Java. Например, `-J-Xms48m` устанавливает память запуска равной 48 мегабайтам.

Помимо стандартных опций доступно использование символов подстановки * и @ для перечисления объектов и ссылки на файл с перечнем объекта. Например, в примере ниже создаётся файл со списком классов, которые впоследствии упаковываются в архив my.jar:

```
C:\Java> dir /b *.class > classes.list

C:\Java> jar cf my.jar @classes.list
```

Java

Компонент для запуска Java-приложений. Запуск приложений включает в себя старт JRE, загрузку указанного класса (см. `entrypoint`) и выполнения основного метода данного класса. Выполняемый при

запуске метод должен быть объявлен, как public и static, а также не должен возвращать ответ и должен иметь входной параметр массива строк. Пример:

```
1 public static void main(String[] args)
```

По умолчанию первый аргумент без параметров, который должен быть вызван, это имя класса. Должно использоваться имя класса с указанием полного пути к нему. Если указана опция `-jar`, то первым аргументом команды должно быть имя JAR-файла, содержащего класс и ресурсы приложения, с загрузчиком определяемым заголовком файла манифеста. JRE осуществляет поиск загрузочного и остальных классов в следующем порядке:

1. Класс, указанный в исходном загрузчике.
2. В установленных расширениях.
3. В пользовательских классах.

Команда `javaw` идентична команде `java` за исключением того, что `javaw` не связано с окном консоли. `javaw` используется в случаях, когда не нужно, чтобы появлялось окно командной строки.

Синтаксис

java [*options*] class [*arguments*]

java [*options*] -jar file.jar [*arguments*]

javaw [*options*] class [*arguments*]

javaw [*options*] -jar file.jar [*arguments*]

Опции

class

> Наименование вызываемого класса.

file.jar

> Наименование вызываемого JAR-файла. Может быть использовано только совместно с командой `-jar`.

arguments

> Аргументы основной функции.

Параметры

-client/-server

> Выбор клиентской или серверной модификаций JVM

-javaagent

> загрузка Java-агента. javaagent это один из параметров JVM, который позволяет указать агент который будет запущен перед стартом приложения. Сам агент - это отдельное приложение которое предоставляет доступ к механизму манипуляции байт-кодом (java.lang.instrument) во время выполнения.

-cp(-classpath)

> Указание пути, по которому содержатся классы, необходимые для запуска. Классы разделяются между собой точкой с запятой (;). Указанием `-classpath` перезаписывается переменная окружения CLASSPATH. Если не указаны переменная окружения CLASSPATH и параметр `-classpath`, то поиск класса осуществляется только в текущей директории.

-jar

> Выполняет программу из JAR-файла. Первый аргумент является именем JAR-файла и выполняет указанный в манифесте загрузочный класс.

-agentlib:libname[=*options*]

> Загрузка отладочного агента, например: `-agentlib:hprof -agentlib:jdwp=help -agentlib:hprof=help`. Команда `-agentlib:foo=opt1,opt2` означает, что Windows должнен найти и загрузить — foo.dll, а OS Solaris — libfoo.so.

-[D]*property*=”value”

> Установка системных свойств. Пример вызова:

```
1  java -Dmydir="some string" SomeClass
```

-enableassertions *:... |:*

–ea *:... |:*

> Включает диагностические утверждения, которые по умолчанию выключены. Диагностические утверждения осуществляют проверки данных в методах и, если некие данные не удовлетворяют условию проверки, осуществляется завершение приложения. Если не указаны аргументы, то диагностические утверждения будут включены во всём приложении, в противном случае только в указанных пакетах/классах. Если задать один пакет и значение ”...”, то диагностические утверждения будут включены в указанном пакете и всех подпакетах. Если указать значение ”...”, то

> диагностические утверждения будут включены во всех пакетах, найденных в текущей директории.
> Пример:

```
java -ea:com.wombat.fruitbat... <Main Class>
```

–disableassertions \:*<package name>...* \|\:*<class\;*

–da \:*<package name>...* \|\:*<class name>*

> Отключает диагностические утверждения

–enablesystemassertions

–esa

> Включение диагностических утверждений в системных классах

–disablesystemassertions

–dsa

> Отключение диагностических утверждений в системных классах

-javaagent:jarpath[=*options*]

> Загрузка агента Java. См. выше.

-jre-restrict-search

> Флаг включения пользовательских сред выполнения Java в поиск версии командой `-version` или `-showversion`.

-no-jre-restrict-search

> Флаг исключения пользовательских сред выполнения Java.

-showversion

> Отображение информации о версии и продолжение работы.

-splash:)imagepath_

> Отображение экране изображения, расположенного в указанном в параметре пути.

-version

> Отображение информации о версии и завершение работы.

-version:*release*

> Вывод информации о версии указанных релизов.

-verbose:class

> Отображает информацию о каждом загруженном классе.

-verbose:gc

> Отображает отчёт по сборщику мусора (garbage collection event).

-verbose:jni

> Информация об используемых нативных методах или других Java Native Interface.

Расширенные параметры

-X Ключ используемых расширенных параметров.

-Xdebug

> Запуск приложения в режиме отладки

-Xint

> Действовать только в режиме интерпретатора. Компиляции в родной код производиться не будет и байткод будет только выполняться интерпретатором. В этом режиме преимущества адаптивного компилятора Java HotSpot Client VM не будет использоваться.

-Xbootclasspath:bootclasspath

> Указание списка каталогов, JAR-файлов или ZIP-архивов для поиска загрузочных классов. Список указывается через точку с запятой.

-Xbootclasspath/a:path

> Указание списка путей к каталогам, JAR-файлам или ZIP-архивам для поиска загрузочных классов после (append) класса по умолчанию. Список указывается через точку с запятой.

-Xbootclasspath/p:path

> Указание списка путей к каталогам, JAR-файлам или ZIP-архивам для поиска загрузочных классов перед (prepend) классом по умолчанию. Список указывается через точку с запятой.

–Xcheck:jni

> Дополнительная проверка для функций Java Native Interface (JNI). В частности, виртуальная машина Java проверяет параметры, переданные функции JNI, а также данные среды выполнения перед обработкой запроса JNI. Любые обнаруженные неверные данные указывают на проблему в нативном коде. В таких случаях виртуальная машина Java завершается с ошибкой. При использовании данной опции будет снижение производительности.

–Xfuture

> Выполнение строгих проверок файлов классов. Для обратной совместимости с прошлыми версия JVM поддерживается нестрогий режим проверки.

-Xnoclassgc

> Отключает сбор мусора (garbage collection). Использование этой опции предотвратит очистку памяти загруженных классов, но повысит общее использование памяти, тем самым обуславливая возможную ошибку OutOfMemoryError.

-Xincgc

> Включает инкрементальный сбор мусора (garbage collector). Инкрементальный сбор мусора (garbage collector) по умолчанию отключён для снижения длительности пауз во время выполнения программы из-за конкурентной работы сборщика мусора. Инкрементальная сборка мусора требует значительной вычислительной мощности процессора.

-Xloggc:file

> Отображает каждое событий при сборке мусора (garbage collection) и выводит результаты в текстовый файл. Дополнительный ключ `-verbose:gc` добавляет к записям событий время (в секундах), которое прошло с момента первого события сборки мусора. Рекомендуется использовать локальное хранилище для того, чтобы избежать задержек при передаче информации по сети. Запись файл может быть прекращена при истечении свободного места, при этом запись будет продолжена в том же файле путём перезаписи прежних событий.

-Xmnsize or -XX:NewSize

> Задаёт размер для экстренного сборщика мусора (garbage collection).

-Xms_n_

> Указывается начальный размер пула выделяемой памяти. Данная величина должна быть кратна 1024 и выше, чем 1 Мб. Указание символа k или K определяет килобайты. Указание символа m или M - мегабайты.

-Xrs

> Снижает отслеживание сигналов ОС от Java VM (нажатие сочетаний клавиш выхода из консоли, закрытие окна консоли, завершение сеанса операционной системы или выключение устройства) для своевременного сохранения дампа памяти на долговременное хранилище с целью последующего восстановления.

-Xssn

> Задаёт размер стека потока.

-Xverify:mode

> Задаёт режим валидации байткода. Валидация байткода проверяет, что файлы классов корректно сформированы и не нарушают заданных JVM ограничений. Доступны режимы:

- remote - валидация всех не загруженных классов (по умолчанию).
- all - валидация всех классов.
- none - выключение механизма валидации байткода.

-XX:+AggressiveOpts

> Включает режим экстремальной оптимизации.

Для получения информации по всем доступным опциям интерпретатора Java требуется использовать ключ.

-help

> Перечень разрешённых опций интерпретатора Java.

javac

Компилятор языка программирования Java. Преобразует исходный код в промежуточный байткод, который хотя и не может быть выполнен непосредственно (как .exe файл, но может быть исполнен посредством интерпретатора java).

Существует два способа обработки файлов с исходными кодами:

1. При небольшом числе файлов они просто перечисляются в командной строке.
2. При значительном числе файлов их список, разделённый пробелами и новыми строками, ведётся во внешнем файле, который впоследствии указывается в команде после символа "@"

Файлы с исходными кодами должны иметь расширение .java. Файлы с классами должны иметь расширение .class, а также наименование, совпадающее с названием класса. Внутренние объявления классов при компиляции создают дополнительные файлы, наименование которых автоматически генерируется по имени исходного класса и внутреннего в формате: [*имя исходного класса*]**$**[*имя внутреннего класса*].class.

Следует размещать файлы с исходными кодами в дереве каталогов, отражающем их вложенность. Например, вы можете хранить все свои исходные коды в каталоге workspace, при этом исходный код класса com.example.myClass должен быть размещён в файле workspace > com > example > myClass.java.

По умолчанию компилятор размещает каждый файл класса в ту же директорию, в которой обнаружен файл с исходным кодом. Существует возможность указания каталога размещения файла при помощи команды `-d`.

Синтаксис

javac [options] [sourcefiles] [classes] [*@argfiles*]

Аргументы могут быть использованы в произвольном порядке.

options

> Опции командной строки.

sourcefiles

> Один и более файлов с исходными кодами для компиляции.

classes

> Один и более классов для обработки аннотаций.

@argfiles

> Один и более файлов, содержащих списки опций и исходных файлов. Параметры JVM `-J` не допускаются в данном компоненте.

Опции

Параметры компиляции можно изменять при помощи ключей компилятора javac:

-version

> Вывести версию компилятора

-J

> Свойство, передаваемое в JVM. Виртуальная машина может изменять своё поведение в зависимости от переданных параметров.

-target

> Указать версию JVM, для которой создаётся класс-файл.

-source

> Указать версию исходного кода.

-bootclasspath

> Указать путь, по которому можно найти классы, необходимые для запуска JVM.

-cp –classpath

> Указание пути, по которому содержатся классы, необходимые для запуска. Классы разделяются между собой точкой с запятой (;). Указанием `-classpath` перезаписывается переменная окружения CLASSPATH. Если не указаны переменная окружения CLASSPATH и параметр `-classpath`, то поиск класса осуществляется только в текущей директории.

-Akey[=*value*]

> Опции, передаваемые обработчикам аннотаций. Они не интерпретируются `javac` напрямую, а становятся доступными для частных обработчиков. Ключи должны быть отделены одной или более точкой ".".

-Djava.ext.dirs=*directories*

> Переопределение каталога размещения установленных расширений.

-d *directory*

> Указание корневого каталога для файлов классов. Каталог должен существовать, т.к. `javac` не сможет создать его. Если класс входит в состав пакета, то `javac` сможет разместить файл класса в подкаталог в соответствии с именем пакета.

-encoding *encoding*

> Выбор файла кодировки, например, EUC-JP или UTF-8. Если ключ `-encoding` не определён, то будет использован конвертер по умолчанию.

-extdirs *directories*

> Компиляция с использованием сторонних расширений другой версии платформы Java. Указываются каталоги размещения расширений через точку с запятой.

-g

> Режим отладки, включая отображения локальных переменных. По умолчанию отображается только номер строки и исходный файл. Возможны следующие режимы:

- g:none - не выводить отладочную информацию
- g:{*keyword list*} - выводить указанные поля отладочной информации:
  - source - название файла с исходными кодами.
  - lines - номера строк с ошибками.
  - vars - значения локальных переменных.

-nowarn

> Отключение отображения предупреждений. Аналогичен ключу `-Xlint:none`.

-s *dir*

> Определяет корневой каталог, в котором будут расположены файлы после компиляции. Файлы будут распределяться по подкаталогам на основании пространства имён пакета.

-verbose

> Включает текстовый вывод загруженных классов и наименований скомпилированных файлов.

–deprecation

> Отображает описание каждого использованного или осуществляется переопределение устаревшего участника класса. Баз данного ключа `javac` отображает имена исходных файлов, которые должны быть использованы для переопределения.

-Werror

> Завершение компиляции, при обнаружении предупреждений.

-X

> Расширенные параметры компиляции

-Xcheck:jni

> Дополнительные проверки для JNI кода.

-Xstdout

> Перенаправление вывода программы.

-Xlint

> Выводить предупреждения о некорректном коде программы. Доступны следующие варианты вывода предупреждений:

- -Xlint:all

  > Включает вывод всех типов рекомендуемых предупреждений. К типам предупреждений относятся внешне корректные синтаксические конструкции Java, которые могут обуславливать ошибки в коде. Например, ненужное приведение типов (тип cast), некорректное содержимое файла класса (тип classfile), использование устаревших методов (тип deprecation) и т.д.

- -Xlint:none

  > Отключает вывод предупреждений.

- -Xlint:name

  > Включает отображение указанного типа предупреждений.

- -Xlint:-name

> Отключает отображение указанного типа предупреждений.

-Xmaxerrs/-Xmaxwarns

> Установить максимальное число выводимых ошибок/предупреждений.

-Xbootclasspath/a (/p)

> Заменить классы, необходимые для запуска компилятора. /a - поставить в начало последовательности классов. /p - поставить в конец последовательности классов.

-Xprefer:{newer,source}

> Определяет в каком файле осуществлять поиск типа: в файле исходного кода или в файле класса.

-Xpkginfo:{always,legacy,nonempty}

> Определяет обработку информационных файлов пакетов

-Xprint

> Текстовое представление указанных типов (для отладки), для которых не будет осуществляться обработка аннотаций и компиляция.

-XprintProcessorInfo

> Вывод информации о запросах на обработку аннотаций.

-XprintRounds

> Вывод информации о начальном и последующих циклах обработки аннотаций.

Помимо стандартных опций доступно использование символов подстановки * и @ для перечисления объектов и ссылки на файл с перечнем объекта.

Для получения информации по всем доступным опциям компилятора требуется использовать ключ.

-help

> Перечень разрешённых опций компилятора.

Javadoc

Генератор документации к Java API. Позволяет создавать HTML-страницы из файлов с исходными кодами.

Синтаксис

javadoc [options] [packagenames] [sourcefilenames] [-subpackages *pkg1:pkg2:...*] [*@argfiles*]

Аргументы могут быть использованы в произвольном порядке.

options

> Аргументы модуля javadoc.

packagenames

> Дополнительные пакеты, разделённые пробелами, которые требуется включить в документацию (например, java.lang java.lang.reflect java.awt). Символы автозамены не поддерживаются, для рекурсивного поиска пакетов требуется использовать ключ `-subpackages`. Компонента Javadoc использует ключ `-sourcepath` для поиска пакетов.

sourcefilenames

> Список файлов с исходными кодами. Допускается использование символов автозамены, например '*'. Для исключения отдельных файлов следует воспользоваться символом "-". Для поиска файлов с исходными кодами ключ `-sourcepath` не используется.

subpackages *pkg1:pkg2:...*

> Создаёт документацию из файлов с исходными кодами указанных пакетов и рекурсивно в их подпакетах. Используется в качестве альтернативы задания пакетам (*packagenames*) или файлам (*sourcefilenames*).

@argfiles

> Один и более файлов, содержащих списки опций и исходных файлов. Параметры JVM -J не допускаются в данном компоненте.

Опции

-overview *path/filename*

> Указывает javadoc-файл, который должен идти в качестве обзорного раздела к документации. Данная опция доступна только при наличии двух и более пакетов. Заголовок документа указывается при помощи ключа `-doctitle`.

-public

> Выводит в документацию только public-классы.

-protected

> Выводит в документацию только protected- и public-классы. Режим по умолчанию.

-package

> Выводит в документацию пакеты, protected- и public-классы.

-private

> Выводит все классы.

-doclet *class*

> Указывает класс для запуска доклета - приложения, работающие со средством Javadoc. Требуется указывать полное имя класса. Доклеты определяют содержание и формат вывода. Если опция не задана, то будет использоваться стандартный доклет, который создаёт HTML-страницу.

-docletpath *classpathlist*

> Указывает путь к файлу класса для запуска доклета (указывается в опции `-doclet`).

-sourcepath *sourcepathlist*

> Определяет пути поиска файлов исходного кода с расширением *.java при поиске имён пакетов или указанных в команде `-subpackages`. Для ввода нескольких путей следует разделять их точкой с запятой. Javadoc осуществляет поиск в каталогах и всех подкаталогах.

-classpath *classpathlist*

> Указывает пути, по которым javadoc будет искать файлы классов с расширением *.class files.

-subpackages *package1:package2:...*

> Создаёт документацию из файлов с исходными кодами, в указанных пакетах и рекурсивно найденных подпакетов. Данная опция полезна при добавлении новых исходников в подпакеты. Например:

```
1 javadoc -d docs -sourcepath C:\user\src -subpackages java:javax.swing
```

Команда на примере выше создаёт документы для пакетов ”java” и ”javax.swing”, а также для всех их подпакетов. Опцию `-subpackages` можно использовать для исключения из списка пакетов для генерации документов (совместно с ключом `-exclude`).

-exclude *packagename1:packagename2:...*

> Исключает пакеты и их подпакеты из списка на генерацию документации. Список формируется при помощи команды `-subpackages`.

-bootclasspath *classpathlist*

> Указывает пути размещения загрузочных классов. Каждый путь отделяется точкой с запятой.

-extdirs *dirlist*

> Указывает каталоги размещения расширений. Каждый путь отделяется точкой с запятой.

-verbose

> Расширенный текстовый вывод при запуске команды javadoc.

-quiet

> Отключение информационных сообщений.

-locale *language_country_variant*

> Указывает региональные настройки и язык документации. В качестве аргумента используется определённые в java.util.Locale константы. Например: en_US (English, United States) или en_US_WIN (Windows variant). Примечание: если используется опция `-locale`, то она должна быть размещена самой первой (слева), в противном случае будут ошибки.

-Jflag

> Флаги управления JVM.

-d *directory* Указывает директорию назначения, куда будут размещаться созданные файлы.

-version

> Включает тег *@version* в документы.

-author

> Включает тег *@author* в документы.

-charset *name*

> Указывает кодировку для генерируемого документа.

Для получения информации по всем доступным опциям генератора документации требуется использовать ключ.

-help

> Перечень разрешённых опций генератора документации.

Javap

Дизассемблер для файлов классов Java. Разбирает класс-файл. Выводимая информация варьируется в зависимости от используемых опций. По умолчанию javap выводит название пакета, а также protected и public поля и методы анализируемого класса.

Синтаксис

javap [options] *classes*

Опции

options

> Опции командной строки.

classes

> Список из одного или более классов, разделённых пробелами. -l Вывод списка локальных переменных.

-public

> Выводит в документацию только public-классы.

-protected

> Выводит в документацию только protected- и public-классы.

-package

> Выводит в документацию пакеты, protected- и public-классы.

-private

> Выводит все классы. Режим по умолчанию.

-[J]*flag*

> Указание параметров JVM.

-s

> Показывает пакет, в котором расположен класс, а также его -protected и -public поля и методы.

-c

> Вывод дизассмеблированного кода, т.е. инструкций, включённых в байткод.

-verbose

> Выводит размер стека, число локальных переменных и аргументов для метода.

Для получения информации по всем доступным опциям генератора заголовочных файлов и файлов-заглушек требуется использовать ключ.

-help

> Перечень разрешённых опций генератора заголовочных файлов и файлов-заглушек.

jlink

jlink инструмент сборки и оптимизации набора модулей с их зависимостями в кастомный образ рантайма.

Синтаксис

jlink [options] --module-path modulepath --add-modules module [,module...] options

options

> Опции командной строки. Разделяются пробелами.

modulepath

> Путь, по которому jlink ищет доступные модули. Эти модули могут быть модульными JAR-файлами, JMOD-файлами, или развернутыми модулями.

module

> Имя модуля добавляемого в имидж рантайма. Jlink добавит этот модуль со всеми зависимостями.

Примечание: Ответственность за обновление кастомных образов рантайма собранных jlink лежит на разработчике

Опции

--add-modules mod [,mod...]

> Добавляет именованные модули mod в корневой набор модулей по-умолчанию. По-умолчанию корневой набор модулей является пустым.

--bind-services

> линкует модули сервис провайдера и их зависимости.

-c ={0|1|2} or --compress={0|1|2}

> Включает компрессию:

1. Без компрессии
2. Разделяемые константы
3. ZIP

--disable-plugin pluginname

> Отключает заданные плагины.

--endian {little|big}

> Определяет байтовый порядок в сгенерированном имидже. Порядком по-умолчанию является порядок в системе пользователя.

-h or --help

> Печать сообщения помощи

--ignore-signing-information

> Не выдавать критические ошибки проверки подписи модульных JAR-файлов слинкованных с рантаймовым имиджем. Относыщиеся к ошибкам подписи файлы не будут скопированы в рантаймовый имидж.

--launcher command=module or --launcher command=module/main

> Specifies the launcher command name for the module or the command name for the module and main class (the module and the main class names are separated by a slash (/)).

--limit-modules mod [,mod...]

> Limits the universe of observable modules to those in the transitive closure of the named modules, mod, plus the main module, if any, plus any further modules specified in the --add-modules option.

--list-plugins

> Lists available plug-ins, which you can access through command-line options; see jlink Plug-ins.

-p or --module-path modulepath

> Specifies the module path. If this option is not specified, then the default module path is $JAVA_HOME/jmods. This directory contains the java.base module and the other standard and JDK modules. If this option is specified but the java.base module cannot be resolved from it, then the jlink command appends $JAVA_HOME/jmods to the module path.

--no-header-files

> Excludes header files.

--no-man-pages

> Excludes man pages.

--output path

> Specifies the location of the generated runtime image.

--save-opts filename

> Saves jlink options in the specified file.

--suggest-providers [name, ...]

> Suggest providers that implement the given service types from the module path.

--version

> Prints version information.

@filename

> Reads options from the specified file. An options file is a text file that contains the options and values that you would typically enter in a command prompt. Options may appear on one line or on several lines. You may not specify environment variables for path names. You may comment out lines by prefixing a hash symbol (#) to the beginning of the line.

Jdb

Отладчик Java. jdb требуется для обнаружения и исправления ошибок в программах Java.

Синтаксис

jdb [options] [class] [arguments]

options

> Опции командной строки.

class

> Имя класса для отладки.

arguments

> Аргументы передаваемые в метод main() класса.

Опции

Большинство опций интерпретатора java доступно и в jdb, например, `-D`, `-classpath` или `-X<_option_>`.

-sourcepath <*dir1:dir2:...*>

> Использует указанные пути для поиска исходных файлов. Если опция не задана, поиск осуществляется в текущем каталоге.

-attach

> Связывает отладчик с предварительно запущенной VM при помощи механизма по умолчанию.

-listen

> Ожидает момента подключения запущенной VM к указанному адресу.

-listenany

> Ожидает момента подключения запущенной VM к любому (any) адресу.

-launch

> Запускает отладку при старте jdb. Данная опция позволяет избежать необходимости ручного выполнения команды `run`.

-listconnectors

> Вывод списка подключений данной VM.

-connect *<connector-name>:<name1>=<value1>,...*

> Подключается к целевой VM по её имени с перечисленными значениями аргументов.

-dbgtrace [*flags*]

> Вывод отладочной информации jdb.

-tclient

> Запуск приложения в Java HotSpot(tm) VM (Client).

-tserver

> Запуск приложения в Java HotSpot(tm) VM (Server).

-[J]*option*

> Передаёт опции загрузчика Java-приложений виртуальной машине Java.

Для получения информации по всем доступным опциям отладчика Java требуется использовать ключ.

-help

> Перечень разрешённых опций отладчика Java.