Java Developer JAR File

Lesson1

JAR

- JAR Java ARchive файл
  - архив zip с файлами .class и файлом манифеста META-INF/MANIFEST.MF
  - используется для упаковки проекта в один файл
  - для поддержки криптозащиты кода

JAR CMD Line

- Создание JAR файла может быть в пакете или командной строке jar/jmxm
  - в пакете IDEA
  - в строке используется утилита jar
- JAR создание архива в командной строке
  - options при создании архива
  - c создать архив
  - v вывести verbose диагностику
  - f вывести в файл, а не на стандартный вывод
  - e задать новую точку входа
  - m добавить информацию в существующий MANIFEST.MF
  - jar-file имя выходного JAR файла
  - manifest-addition имя текстового файла, который будет добавлен в MANIFEST.MF
  - содержимое файла должно быть в UTF-8 формате
  - input-file(s) имя файлов *.class для добавления в JAR архив
- ВНИМАНИЕ. Содержимое текста для MANIFEST.MF файла ДОЛЖНО быть в UTF-8 формате
- ВНИМАНИЕ. Порядок ключей f и m должен быть такой же как и файлы в командной строке

JAR Manifest

- JAR Manifest параметры манифеста
  - entry point точка входа программы в пакет это класс который будет запускаться main()
  - `Main-Class: classname`
  - <\n>
- ВНИМАНИЕ. После строки ОБЯЗАТЕЛЬНО должна быть добавлена еще одна пустая строка
- Пример. реализация

```
jar cfm MyJar.jar Manifest.txt MyPackage/*.class

Manifest.txt:
Main-Class: MyPackage.MyClass
Main-Path: lib/file-name.jar lib/file2-name.jar
```

- В результате будет создан файл MANIFEST.MF

```
Manifest-Version: 1.0
Created-By: 1.7.0_06 (Oracle Corporation)
Main-Class: MyPackage.MyClass
Main-Path: lib/file-name.jar lib/file2-name.jar
```

- Запуск полученного архива
- Пример. реализация

```
java -jar MyJar.jar
```

JAR Project

- JAR Project создание JAR файла в консоли без внешних библиотеке **jar/jmxm**
- Файл манифеста
  - Manifest.txt завершается обязательно пустой строкой
- Пример. реализация manifest.txt

```
Main-Class: java02.jmxmodel.jmxm.MainStart
```

- Файл создания JAR
  - check.cmd файл который компилирует все файлы классов
  - размещает их в пакете и создает JAR файл
  - запускает файл JAR на исполнение
- Пример. реализация check.cmd

```
rem создание JAR файла для группы файлов
rem из проекта, расположенных в оригинале
rem в пакете  src/java02/jmxmodel/jmxm

javac *.java
mkdir java02\jmxmodel\jmxm
move *.class java02\jmxmodel\jmxm\
jar -cvfm  jmxm.jar manifest.txt java02\*

java -jar jmxm.jar
```

JAR Project with librarises

- JAR Project создание JAR файла в консоли с внешними библиотеками **jar/jmxh**
- Файл манифеста
  - Manifest.txt завершается обязательно пустой строкой
- Пример. реализация manifest.txt

```
Main-Class: java02.jmx.jmxh.SimpleAgent
Class-Path: _lib/jmxtools.jar _lib/jmxri.jar
```

- Файл создания JAR
  - check.cmd файл который компилирует все файлы классов
  - размещает их в пакете и создает JAR файл
  - запускает файл JAR на исполнение
- Пример. реализация check.cmd

```
rem Create JAR file for sources from project
rem placed in src/java02/jmx/jmxh package  java02/jmx/jmxh
rem with library _lib/jmxtool.jar
rem Attention  DO NOT FORGET ./ in -cp .;_lib/*
rem            IMPOSSIBLE to use other jar inside given withou Classloader in main jar

set CLASSPATH=
javac -cp .;_lib/* *.java
mkdir java02\jmx\jmxh
move /y *.class java02\jmx\jmxh\
jar -cvfm  jmxh.jar manifest.txt java02\*
java -cp .;_lib/* -jar jmxh.jar
```

-
- ВНИМАНИЕ. Вложенные JAR библиотеки напрямую использовать нельзя, их надо загружать ClassLoader
-

JAR Project IDEA

- JAR Project IDEA создание JAR из модулей или проекта в IDEA **jar/jmx_idea**
- Создание JAR из модуля
    - Project Structure >> Artifacts >> Add module >> JAR >> From module and dependencies
    - Module >> jmxh
    - MainClass >> Project >> jmxh >> src >> SimpleAgent
    - Выбрать будут ли библиотеки JAR извлечены и встроены или пойдут отдельно
    - Extract >> Select >> extract to the target JA
    - все классы будут извлечены и добавлены напрямую в JAR
    - Non Extract >> Select >> copy to the output and link via manifest
    - в Манифест будет добавлен путь до файлов JAR которые используются в модуле
    - в каталог с JAR будут скопированы все файлы JAR libraries
    - OK
- Компиляция и создание JAR файла
    - Build >> Rebuild Module
    - Build >> Build Artifacts >> jmxh.jar >> Build
    - Все, забрать файл в папке out/artifacts/jmxh
- Запуск в работу
    - запустить check.cmd
    - java -jar jmxh.jar

JAR Manifest Path

- JAR Manifest изменение параметров Manifest.MF
- Смена пути для библиотек: **jar/jmx_idea/jmxh**
    - Можно все библиотеки поместить в отдельную папку _lib в каталоге с JAR файлом
    - создать папку _lib и скопировать туда все файлы
    - создать файл manifest.txt с содержанием
    - `Class-Path: _lib/jmxtools.jar _lib/jmxri.jar`
    - запустить команду обновления пути до библиотек
    - `jar ufm jmxh.jar manifest.txt`
- Смена пути может быть при помощи 7Zip
    - было:
    - `Manifest-Version: 1.0`
    - `Class-Path: jmxtools.jar jmxri.jar`
    - `Main-Class: SimpleAgent`
    - стало:
    - `Manifest-Version: 1.0`
    - `Class-Path: _lib/jmxtools.jar _lib/jmxri.jar`
    - `Main-Class: SimpleAgent`
- Запустить файл JAR
    - `java -jar jmxh.jar`
- Смена entry point: **jar/jmx_idea/jmxhe**
    - запустить команду обновления entry-point
    - `jar -uvfe jmxhe.jar MainStart`

JAR Manifest Version

- JAR Manifest изменение параметров Manifest.MF **jar/jmx_idea/jmxh**
- Файл Manifest.txt

```
Name: java/util/
Specification-Title: Java Utility Classes
Specification-Version: 1.2
Specification-Vendor: Example Tech, Inc.
Implementation-Title: java.util
Implementation-Version: build57
Implementation-Vendor: Example Tech, Inc.
```

- Команда на выполнение
    - `jar -cvfm  jmxh.jar manifest.txt java02\*`
- Пример. реализация Manifest.FM после выполнения строки

```
Manifest-Version: 1.0
Name: java/util/
Specification-Title: Java Utility Classes
Specification-Version: 1.2
Specification-Vendor: Example Tech, Inc.
Implementation-Title: java.util
Implementation-Version: build57
Implementation-Vendor: Example Tech, Inc.
Main-Class: java02.jmx.jmxh.SimpleAgent
Class-Path: _lib/jmxtools.jar _lib/jmxri.jar
Created-By: 10 (Oracle Corporation)
```

JAR Manifest Sealed

- JAR Manifest изменение параметров Manifest.MF **jar/jmx_idea/jmxh**
    - Sealed запечатывание JAR файла для сохранения всех файлов одной версии
- Файл Manifest.txt

```
Sealed: true
```

- Команда на выполнение
    - `jar -cvfm  jmxh.jar manifest.txt java02\*`
- Пример. реализация Manifest.FM после выполнения строки

```
Manifest-Version: 1.0
Name: java/util/
Specification-Title: Java Utility Classes
Specification-Version: 1.2
Specification-Vendor: Example Tech, Inc.
Implementation-Title: java.util
Implementation-Version: build57
Implementation-Vendor: Example Tech, Inc.
Main-Class: java02.jmx.jmxh.SimpleAgent
Class-Path: _lib/jmxtools.jar _lib/jmxri.jar
Sealed: true
Created-By: 10 (Oracle Corporation)
```

-

JAR Manifest Security

- JAR Manifest изменение параметров Manifest.MF **jar/jmx_idea/jmxh**
- Permissions атрибут доступа, может иметь значения sandbox, all-permissions
    - sandbox RIA запускается в защищенной sandbox и не запрашивает ресурсы
    - all-permissions RIA запрашивает дополнительные ресурсы
    - Пример. реализация

```
Permissions: sandbox
Permissions: all-permissions
```

- Codebase атрибут ограничивает работу приложения в рамках конкретного URL домена
    - 127.0.0.1 в рамках http://127.0.0.1, http://127.0.0.1:8080
    - www.example.com в рамках https://www.example.com, http://www.example.com
    - Пример. реализация

```
Codebase: 127.0.0.1:8080
```

- Application-Name атрибут задает имя приложения
    - Пример. реализация

```
Application-Name: Hello World
```

- Application-Library-Allowable-Codebase атрибут допустимого расположения приложения
    - https://host.example.com *.samplehost.com домены откуда можно запускать приложение
- Пример. реализация

```
Application-Library-Allowable-Codebase: https://host.example.com *.samplehost.com
```

- Caller-Allowable-Codebase атрибут допустимого домена откуда могут быть вызовы приложению
    - host.example.com 127.0.0.1 домены откуда JavaScript может делать запросы приложению
- Пример. реализация
    - `Caller-Allowable-Codebase: host.example.com 127.0.0.1`

- Entry-Point атрибут допустимых классов которые могут быть entry_point
- Пример. реализация
    - `Entry-Point: apps.test.TestUI apps.test.TestCLI`

- Trusted-Only атрибут разрешает загружать только trusted компоненты
- Пример. реализация
    - `Trusted-Only: true`

- Trusted-Library атрибут не дает использовать привилегированный код untrusted компонентами
- Пример. реализация
    - `Trusted-Library: true`

JAR Manifest Signing Code

- JAR Manifest изменение параметров Manifest.MF **jar/jmx_idea/jmxh**
- JAR Signing Code процедура подписи JAR файла при помощи ключа
  - –keystore URL указывает хранилище ключей, если не используется .keystore database
  - –sigfile file задает имя .SF или .DSF файлов, если alias не используется
  - –signedjar file указывает имя подписанного JAR файла на выходе если надо сохранить src
  - –tsa URL задает timestamp для подписи используя TSA которое задается URL
  - –tsacert alias задает timestamp для подписи используя TSA которое задается alias
  - –altsigner class показывает альтернативный механизм подписи при помощи класса
  - –altsignerpath CPath задает путь для подписи altsigner где располагаются классы
- Пример. реализация подпись JAR файла app.jar
  - параметры alias "johndoe"
  - keystore "mykeys" в текущем каталоге
  - TSA URL http://tda.url.example.com
  - выполнение будет запрошен пароль доступа к keystore и alias
  - т.к. нет опции `-sigfile` будут созданы файлы `JOHNDOE.SF, JOHNDOE.DSF`
  - т.к. нет опции `-signedjar` файл `app.jar` будет перезаписан signed версией

```
jarsigner -keystore mykeys -tsa http://tsa.url.example.com app.jar johndoe
```

JAR Manifest Signing Tools

- JAR Manifest Signing Tools **jar/jmx_idea/jmxh/signed**
  - keytool утилита для создания подписанного JAR файла
  - jarsigner утилита для подписи JAR файла
- Процедура подписания JAR файла
  - создать <keypair> в стандартном keystore для <alias> = jmxh и заполнить все данные
  - заполнить все параметры для данной пары <keypair>
  - `keytool -genkeypair jmxh`
  - посмотреть entries в keystore
  - `keytool –list`
  - <необязательно>
  - создать сертификат для данного alias и распечатать данный сертификат
  - `keytool –exportcert -alias jmxh –file certjmxh.txt`
  - `keytool –printcert –file certjmxh.txt`
  - подписать JAR файл без использования TSA
  - `jarsigner jmxh.jar jmxh -signedjar jmxhs.jar`
  - подписать JAR файл с использованием TSA
  - `jarsigner jmxh.jar jmxh -signedjar jmxhs2.jar`
  - `-tsa http://timestamp.digicert.com`
  - `jarsigner jmxh.jar jmxh -signedjar jmxhs3.jar`
  - `-tsa http://sha256timestamp.ws.symantec.com/sha256/timestamp`
  - проверка
  - `jarsigner –verify jmxhs3.jar`

Процедура подписания JAR в custom keystore

- Процедура подписания JAR в custom keystore **jar/jmx_idea/jmxh/_signed**
    - процедура check.cmd
- Пример. реализация подписания JAR в custom .\keystore\keystore
    - `keytool -genkeypair -keystore .\keystore\keystore -alias jmxhc`
    - `jarsigner -keystore keystore/keystore jmxh.jar -signedjar jmxhsc.jar jmxhc`
    - проверка
    - `jarsigner –verify –verbose –certs jmxhs3.jar`
- Процедура подписания JAR в custom keystore и внешние библиотеки **jar/jmx_idea/jmxh/_signed**
    - процедура check_signed.cmd
- Пример. реализация подписания JAR в custom .\keystore\keystore
    - `mkdir signed`
    - `mkdir signed\_lib`
    -
    - `rem custom keystore`
    - `keytool -genkeypair -keystore .\keystore\keystore -alias jmxhc`
    - `jarsigner -keystore keystore/keystore jmxh.jar -signedjar signed\jmxh.jar jmxhc -tsa http://sha256timestamp.ws.symantec.com/sha256/time`
    - `jarsigner -keystore keystore/keystore _lib\jmxri.jar -signedjar signed\_lib\jmxri.jar jmxhc -tsa http://sha256timestamp.ws.symantec.com/sha256/time`
    - `jarsigner -keystore keystore/keystore _lib\jmxtools.jar -signedjar signed\_lib\jmxtools.jar jmxhc -tsa http://sha256timestamp.ws.symantec.com/sha256/time`
    - `cd signed`
    - `java -jar jmxh.jar`
    - `pause`
    - проверка
    - `jarsigner -verify signed\jmxh.jar`
    - `jarsigner -verify signed\_lib\jmxtools.jar`
    - `jarsigner -verify signed\_lib\jmxri.jar`
    - `pause`
    - проверка полная
    - `jarsigner -verify -verbose -certs signed\jmxh.jar`
    - `pause`
    - `jarsigner -verify -verbose -certs signed\_lib\jmxtools.jar`
    - `pause`
    - `jarsigner -verify -verbose -certs signed\_lib\jmxri.jar`
    - `pause`
    - `cmd-verify –verbose –certs signed\_lib\jmxri.jar`
    - `cmd`
- ВНИМАНИЕ. При проверке предупреждения, это НЕВАЖНО, так как используются разные версии Java
- Чтобы устранить предупрежлдения надо удалить JRE7 и установить JRE6 uypdate 30

JAR загрузка классов

- JAR загрузка классов **java_questions/jarload**
  - внешняя JAR libraries размещаются в одной из папок и в атрибуте Class:Path Manifest.MF
  - прописывается путь до каждой JAR library
  - runtime JAR libraries размещаются в одной из папок, никаких путей нет в Manifest.MF
  - создается ClassLoader JAR application,в него добавляется путь до каждого JAR library
  - ClassLoader загружает все классы JAR libraries, которые дают ClassDefNotFound
  - ClassLoader загружает основной JAR app класс и запускает его main() метод
- JAR процедура загрузки
  - создать JarClassLoader на базе URL("app/jmxh.jar") файла
  - получить подключение к файлу,и загрузить атрибуты jar и получить имя Main-Class
  - загрузить класс
  - запустить main() метод Main-Class
- JAR Внешние библиотеки
  - получить JarClassLoader запускаемого JAR пакета
  - добавить addURL() к загрузчику URL("lib/jmxtools.jar") и по сути это тоже самое –cp lib/*
  - загрузить класс JAR library
  - продолжить с основным классом, загрузить класс, запустить его main()
- JAR Внешняя загрузка **java_questions/jarload/JarRunner2**
  - classpath -cp задается с командной строки java –cp lib/* путь до JAR libraries
  - запуск JarRunnerMainStart.main() используется Runtime.getRunTime().exec("cmd /c")
  - запуск по сути идет с командной строки
  - сами классы загружаются в приложении
- Пример. реализация загрузки
- JAR Runtime загрузка **java_questions/jarload/JarRunner3**
  - classpath задается добавлением URL(lib/*) до библиотек в основной ClassLoader
  - запуск JarRunner3 стандартный вызов main() и загрузка JAR файлов и классов
  - сами классы загружаются в приложении
- Пример. реализация загрузки JarClassLoader
- ```
  public class JarClassLoader extends URLClassLoader {
      private URL url;
      public JarClassLoader(URL url) {    // replace with custom constructor
          super(new URL[]{url});
          this.url = url;
      }
      public void addURL(String s) {
          URL url = null;
          try{
              url = new URL(s);
              addURL(url);

          }catch (MalformedURLException e) {
              e.printStackTrace();
          }
      }
      public String getMainClassName() throws IOException {
          URL u = new URL("jar", "", url + "!/");                    // url for jar
          JarURLConnection uc = (JarURLConnection) u.openConnection();
          Attributes attr = uc.getMainAttributes();                  // attributes from jar
          if (attr != null) return attr.getValue(Attributes.Name.MAIN_CLASS);
          return null;
      }
  ```
  ```

- Пример. реализация загрузки JarClassLoader
- ```
  public void invokeClass(String name, String[] args) throws
              ClassNotFoundException, NoSuchMethodException,
              InvocationTargetException {
  // reflection
          Class c = loadClass(name);
          Method m = c.getMethod("main", new Class[]{args.getClass()});  // String.class
          m.setAccessible(true);
          int mods = m.getModifiers();
          if(m.getReturnType() != void.class || !Modifier.isStatic(mods) ||
          !Modifier.isPublic(mods)) {
              throw new NoSuchMethodException("main"); // main not void, not static, not public
          }
          try {
              m.invoke(null,new Object[] {args});
          }catch (IllegalAccessException e) {
              e.printStackTrace();
          }
      }
  }
  ```
- Пример. реализация загрузчика JAR
- ```
  public class JarRunner3 {
      private static final String[] JAR_FILES = new String[]{
              "app/jmxh.jar", "lib/jmxtools.jar", "lib/jmxri.jar"
      };
      private static void fatal(String s, int code) {
          System.err.println(s);
          System.exit(code);
      }
      private static void fatal(String s) {
          fatal(s, 0);
      }
      private static void usage() {
          fatal("Usage: java JarRunner [args...]");
      }
      private static JarClassLoader getJarLoader(String s) {
          URL url = null;
          try {
              url = new URL(s);
          } catch (IOException e) {
              fatal("Invalid URL: " + s);
          }

          return new JarClassLoader(url);
      }

   // независимая загрузка всех классов
      private static void loadJarClasses(String pathToJar)
                                              throws IOException, ClassNotFoundException {
          pathToJar = pathToJar.substring(9, pathToJar.length());
          JarFile jarFile = new JarFile(pathToJar);
          Enumeration<JarEntry> e = jarFile.entries();

          URL[] urls = {new URL("jar:file:" + pathToJar + "!/")};
          URLClassLoader cl = URLClassLoader.newInstance(urls);

          while (e.hasMoreElements()) {
              JarEntry je = e.nextElement();
              if (je.isDirectory() || !je.getName().endsWith(".class")) {
                  continue;
              }
              // -6 because of .class
              String className = je.getName().substring(0, je.getName().length() - 6);
              className = className.replace('/', '.');
              Class c = cl.loadClass(className);
          }
      }
  ```

- Пример. реализация загрузчика JAR (продолжение)

```
// независимая загрузка одного класса
    private static void loadJarClass(String pathToJar, String name) throws IOException {
        pathToJar = pathToJar.substring(9, pathToJar.length());
        try {
            File jarFile = new File(pathToJar);
            URLClassLoader loader = new URLClassLoader(new URL[]{jarFile.toURI().toURL()});

            Class.forName(name, true, loader);
            System.out.println("Success");
            loader.close();
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        System.out.println("Runtime load JAR app and JAR
                                libraries with common loader and internal classpath:");
        System.out.println("Build module jmxh. Build Artifact Module jmxh: jmxh.jar");
        System.out.println("Copy from module jmxh: jmxh.jar to app/jmxh.jar");
        System.out.println("                       jmxtools.jar to lib/jmxtools.jar");
        System.out.println("                       jmxri.jar to lib/jmxri.jar");

        if (args.length < 1) {
//            usage();
            String path = JarRunner3.class.getResource(".").getPath();

            args = new String[JAR_FILES.length];
            for (int i = 0; i < JAR_FILES.length; i++) {
                args[i] = "file:///" + path+ JAR_FILES[i];
            }
        }
        JarClassLoader cl = getJarLoader(args[0]);   // loader JAR app
        cl.addURL(args[1]);                          // analog -cp  add JAR to classpath
        cl.addURL(args[2]);
        String name = null;
        try {
            name = cl.getMainClassName();
        } catch (IOException e) {
            fatal("I/O error while loading JAR file", 1);
        }

        String[] newArgs = new String[args.length - 3];
        System.arraycopy(args, 3, newArgs, 0, newArgs.length);

        try {
//            loadJarClasses(args[1]);
//            loadJarClass(args[1],"com.sun.jdmk.comm.HtmlAdaptorServer");
            cl.loadClass("com.sun.jdmk.comm.HtmlAdaptorServer");
            cl.invokeClass(name, newArgs);
        } catch (ClassNotFoundException e) {
            fatal("Class not found: " + e, 1);
        } catch (NoSuchMethodException e) {
            fatal("No such method: " + e, 1);
        } catch (InvocationTargetException e) {              // to load class which issue
            fatal(e.getTargetException().toString(), 1);     // NoClassDefFoundError
        }

    }
}
```

-

TRICKS

- JAR in JAR чтобы использовать другие jar внутри данного есть два способа
  - первый загружать дочерний jar при помощи Classloader
  - второй использовать сторонние библиотеки Uber Jar или Shade
- Example приложение показывающее погоду

  repo/app
  - выводит данные о погоде взятые из интернет, для указанного места
  -
  -
  -
- Пример. реализация
  -
  -
  - f

Programming Assignment

- Programming Assignment 2 задания **m55/src**
  - реализовать Example Implementation
  - реализовать Example Implementation

Example Implementation

- Example Implementation
  - реализовать класс Example
- Описание алгоритма
  - описание алгоритма
- Пример. реализация Example Implementation
  - описание деталей
- ВНИМАНИЕ. Example Implementation ОПИСАНИЕ пункта
-

Lesson2

Example

- Example приложение показывающее погоду **repo/app**
  - выводит данные о погоде взятые из интернет, для указанного места
  -
  -
- Пример. реализация
  -
  -

Example

- Example приложение показывающее погоду **repo/app**
  - выводит данные о погоде взятые из интернет, для указанного места
  -
  -
- Пример. реализация
  -
  -

Programming Assignment

- Programming Assignment 2 задания **m55/src**
  - реализовать Example Implementation
  - реализовать Example Implementation

Example Implementation

- Example Implementation
  - реализовать класс Example
- Описание алгоритма
  - описание алгоритма
- Пример. реализация Example Implementation
  - описание деталей
- ВНИМАНИЕ. Example Implementation ОПИСАНИЕ пункта
-
-

Appendix C

TRICKS

Example

- Example
    - problem description **repo/app**
    -
    - solution description **repo/app**
    -

GIT

- GIT
  - клонировать с github — git clone https://github.com/v777779/jup
  - удалить папку — git rm –r stage_one_temp
  - git commit –m "removed stage_one_temp folder"
  - git push
  - удалить физически — rmdir /s /q stage_one_temp
  - добавить файл — git add stage_one/_exam_code/*
  - git commit -m "added stage_one final code"
  - git push
  - восстановление всего — git checkout *
  - git checkout stage_one/v3
- Если не добавляет файл *.jar
  - значит работает .gitignore — git add * -f
  - форсировать добавление
  -