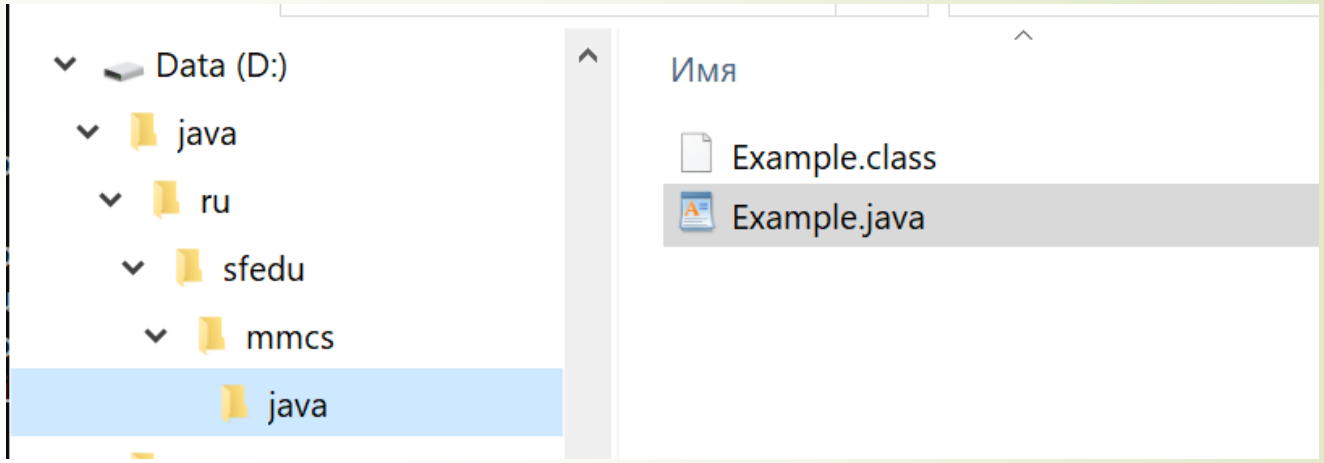
Пакеты

Пакеты

- Основная причина использования пакетов – гарантия однозначности имен классов
- Имя пути к файлу класса в файловой системе должно совпадать с именем пакета

Пример пакета

```
package ru.sfedu.mmcs.java;

public class Example{
. . .

}
```




```
D:\java>javac ru/sfedu/mmcs/java/Example.java

D:\java>java ru/sfedu/mmcs/java/Example
```

Рекомендуется размещать файлы .java и файлы .class в разных каталогах, Например, src и classes

Jar файлы

- Применяются для формирования пакета библиотеки в виде одного файла
- А также для создания файла запускаемого приложения (имеющего класс для запуска с методом main)

Команда

java –jar aplication.jar

Найдет в архиве главный запускающий класс и начнет выполнять его метод main

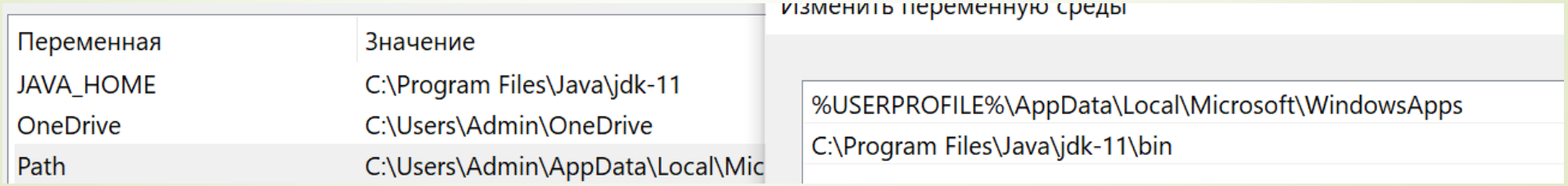
Путь к классу

- Если нужно просто запустить класс на выполнение

```
D:\java>javac ru/sfedu/mmcs/java/Example.java

D:\java>java ru/sfedu/mmcs/java/Example
```

JRE

- JRE представляет собой комбинацию виртуальной машины Java (JVM) и библиотек, необходимых для выполнения кода во время выполнения
- JDK - это пакет инструментов для разработки приложений Java, тогда как JRE - это пакет инструментов для запуска приложений Java.
- JRE входит в JDK.
- Путь к JDK необходимо знать программам, которые будут использовать приложения из набора – например, среде разработки. Задается переменной среды JAVA_HOME
- Переменная PATH задает путь к каталогу /bin, в котором находятся инструменты разработки



Добавление библиотеки к проекту

- Переменная CLASSPATH может быть добавлена в переменные среды
- Или значение ее можно установить перед выполнением компиляции и запуска

Точка!!!

set classpath= . ; C:\Drivers\firebirdsql-full.jar

- При каждом запуске команды javac или java задать ключ –classpath

java -classpath .; C:\Drivers\ firebirdsql-full.jar <имя класса>

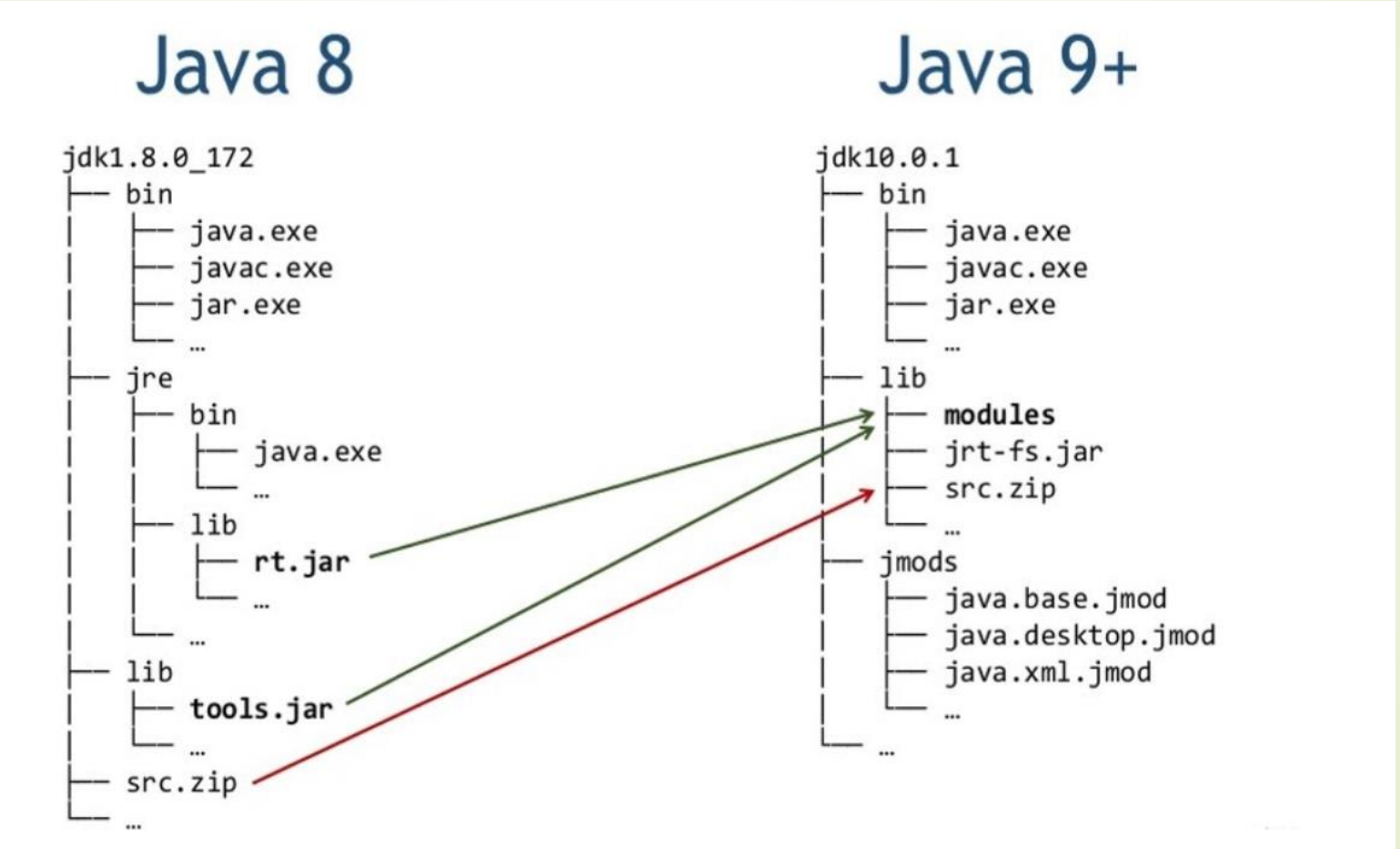
- Если подключается несколько библиотек, можно разместить их в одном каталоге и определи переменную classpath так:

set classpath= . ; C:\java\lib*

Модули

- Java Platform Module System (начиная с Java 9)
- Модуль — это новый уровень агрегации пакетов и ресурсов
- Утилита JLINK позволяет создавать наборы JRE, которые будут включают только «нужные» модули, которые реально необходимы вашему приложению
- Модули повышают инкапсуляцию и безопасность

Новая структура JDK



Дескриптор модуля

- module-info.java
- Включается в jar архив модуля
- Содержит
  - имя,
  - зависимости,
  - экспортируемые пакеты,
  - потребляемые и предоставляемые сервисы,
  - разрешения для reflection доступа

Примеры дескрипторов модулей

```
module java.sql {
  requires transitive java.logging;
  requires transitive java.transaction.xa;
  requires transitive java.xml;
  exports java.sql;
  exports javax.sql;
  uses java.sql.Driver;
}
```

```
module jdk.javadoc {
  requires java.xml;
  requires transitive java.compiler;
  requires transitive jdk.compiler;
  exports jdk.javadoc.doclet;
  provides java.util.spi.ToolProvider
      with  jdk.javadoc.internal.tool.JavadocToolProvider;
  provides javax.tools.DocumentationTool
      with jdk.javadoc.internal.api.JavadocTool;
  provides javax.tools.Tool
      with jdk.javadoc.internal.api.JavadocTool;
}
```

Типы модулей

- System Modules — Java SE и JDK модули
- Application Modules — модули приложения, а также те зависимости (от сторонних библиотек), которые использует наше приложение
- Automatic Modules — это модули с открытым доступом, создаваемые Java автоматически из JAR-файлов
- Unnamed Module — безымянный модуль, автоматически создаваемый из всех JAR-файлов, которые прописаны в classpath. Это универсальный модуль для обеспечения обратной совместимости с ранее написанным Java кодом

Путь к библиотеке модулей

- Аналог classpath для модулей

Параметр --module-path указывает путь к модулю

Параметр --module определяет имя главного запускаемого файла в модуле

Сборки проектов Apache NetBeans

- Apache Ant (Another neat tool)
  - Похож на утилиту сборки make
  - Использует файл формата XML
  - На больших проектах разрастаются
  - Плохо структурированы
  - Нет ограничения на использование jar-файлов

```
<?xml version="1.0"?>
<project name="HelloWorld" default="run">
    <target name="compile">
        <mkdir dir="build/classes"/>
        <javac destdir="build/classes" includeantruntime="false">
            <src path="src"/>
        </javac>
    </target>
    <target name="run" depends="compile">
        <java classname="HelloWorld" classpath="build/classes"/>
    </target>
    <target name="clean">
        <delete dir="build"/>
    </target>
</project>
```

Сборки проектов Apache NetBeans

```
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
/project>
```

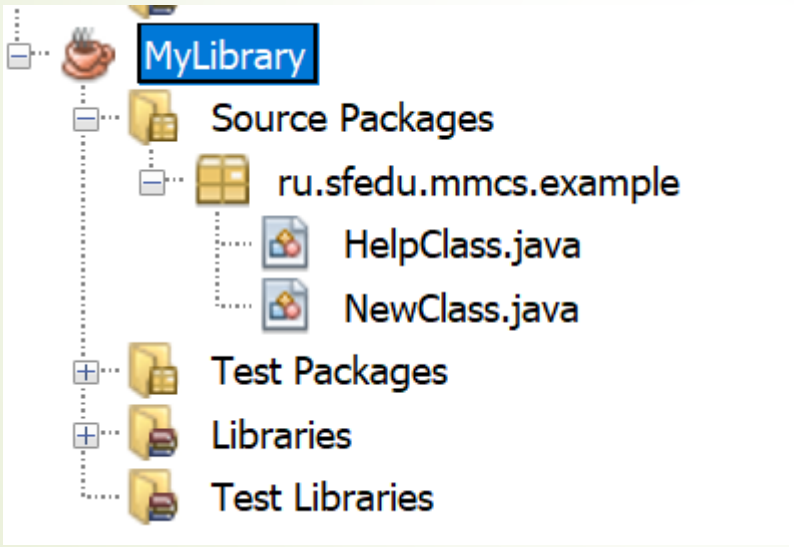
- Maven
  - Использует XML для построения конфигурации (файл называется *POM.xml*)
  - *POM.xml* содержит не отдельные команды, а описание структуры проекта
  - Требует четко придерживаться определенной структуры каталогов
  - Допускает использование только одного jar-файла
  - Выполняет автоматическое разрешение зависимостей (скачивание с учетом версии, распаковку, подключение)
  - Все задачи по обработке файлов maven выполняет посредством их обработки последовательностью встроенных и внешних плагинов

Сборки проектов Apache NetBeans

- Gradle
  - Gradle - предметно-ориентированный язык на основе Groovy

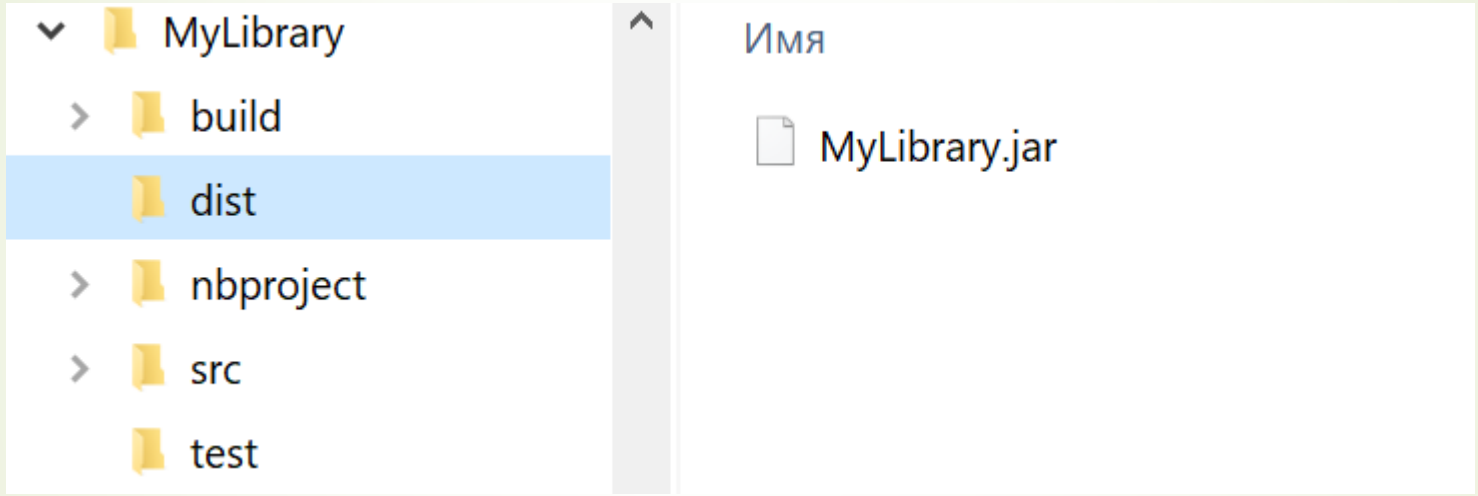
Среда разработки NetBeans

- Создадим библиотеку классов



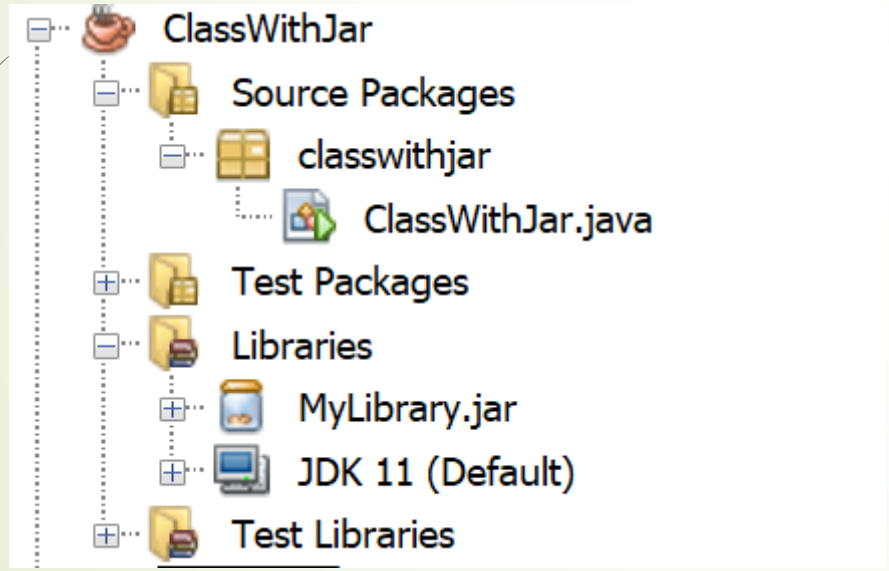

```java
package ru.sfedu.mmcs.example;

/**
 *
 * @author Admin
 */
public class HelpClass {
    public String toString(){
        return "Help Class";
    }

}
```

Среда разработки NetBeans



Среда разработки NetBeans

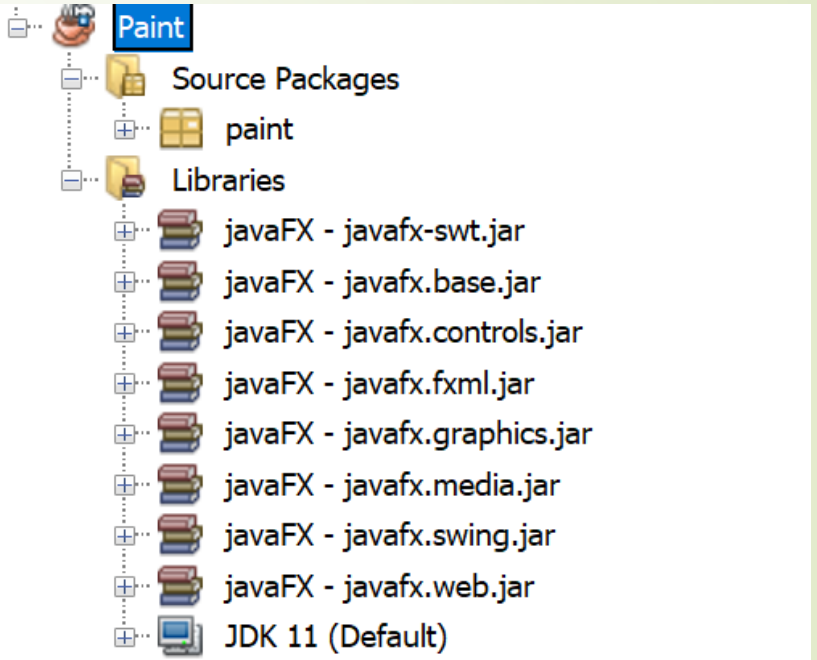
- Подключение в проект в сборке ANT




```java
package classwithjar;

import ru.sfedu.mmcs.example.HelpClass;

/**
 *
 * @author Admin
 */
public class ClassWithJar {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        HelpClass h = new HelpClass();
        System.out.println(h);
    }

}
```

Среда разработки NetBeans

- Подключение библиотеки модулей в проект ANT

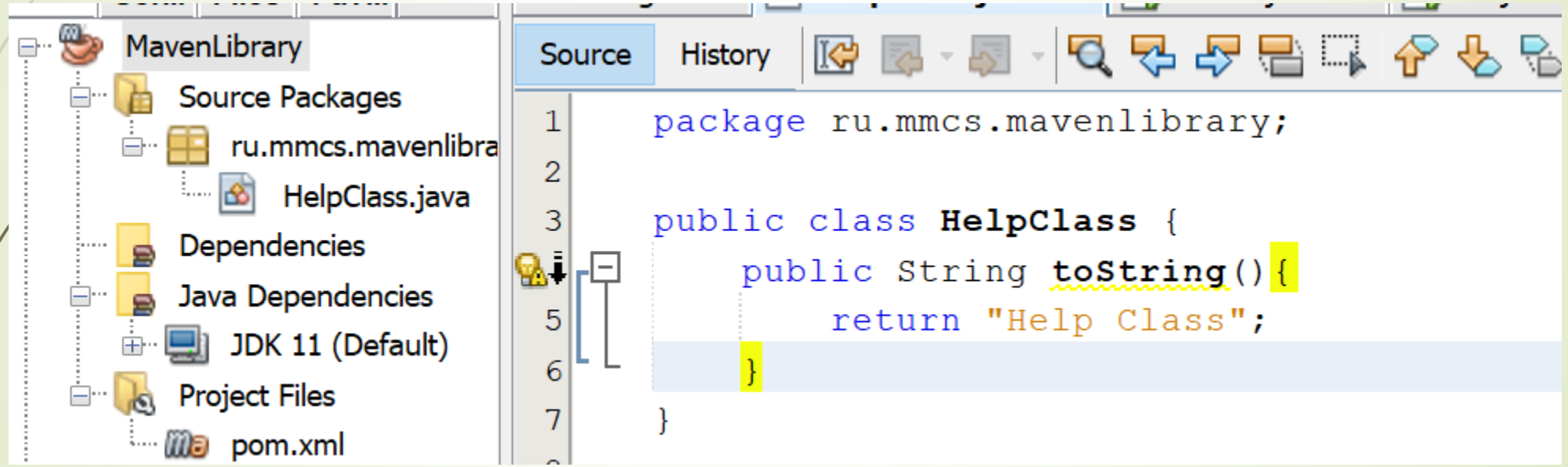


VM Options:

```
--module-path "D:\openjfx-19.0.2.1_windows-x64_bin-sdk\javafx-sdk-19.0.2.1\lib"
--add-modules javafx.controls,javafx.fxml
```

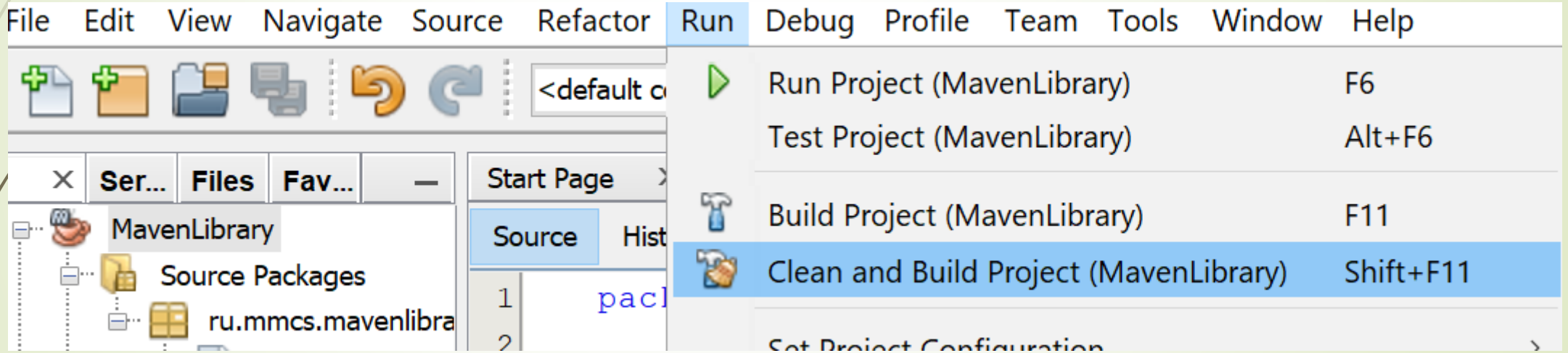
Среда разработки NetBeans

- Создадим библиотеку классов для сборки Maven (артефакт)



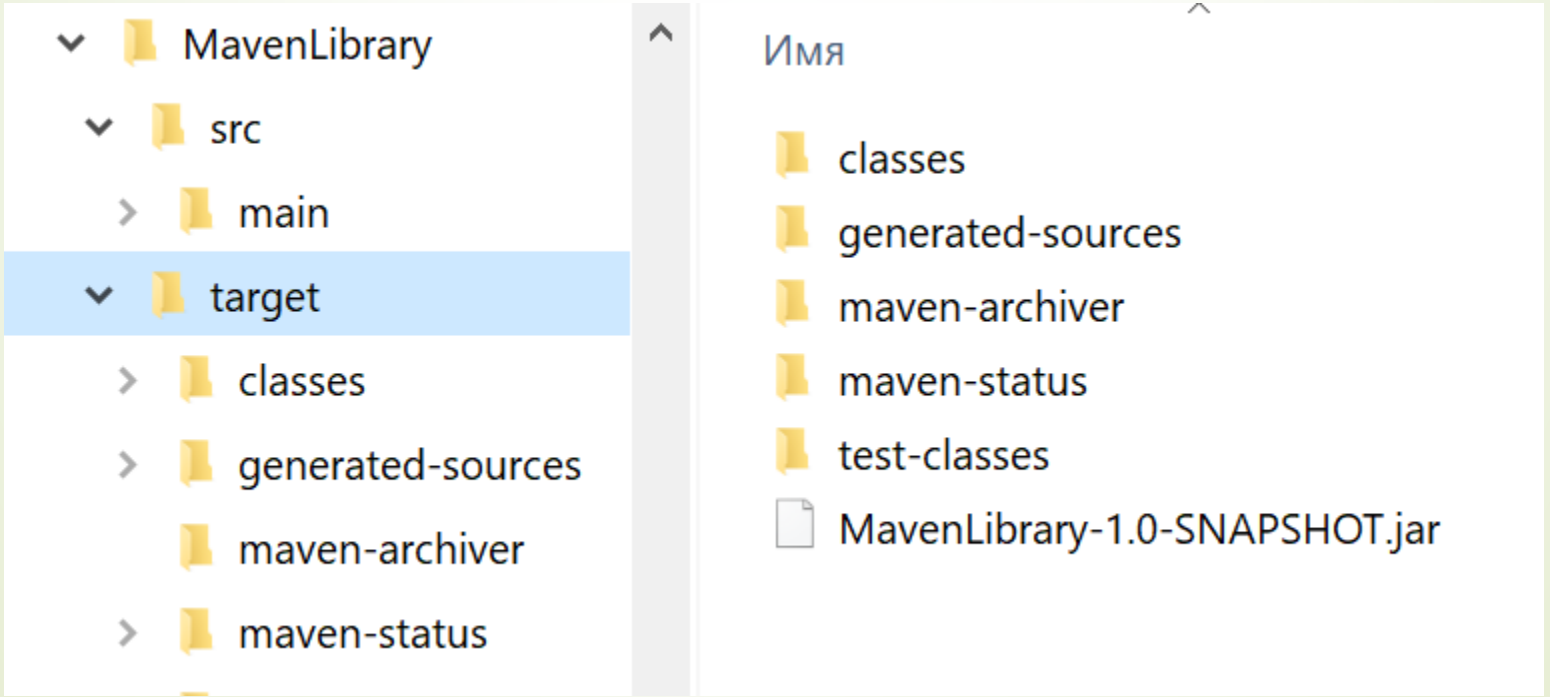
Среда разработки NetBeans

- Создадим библиотеку классов для сборки Maven



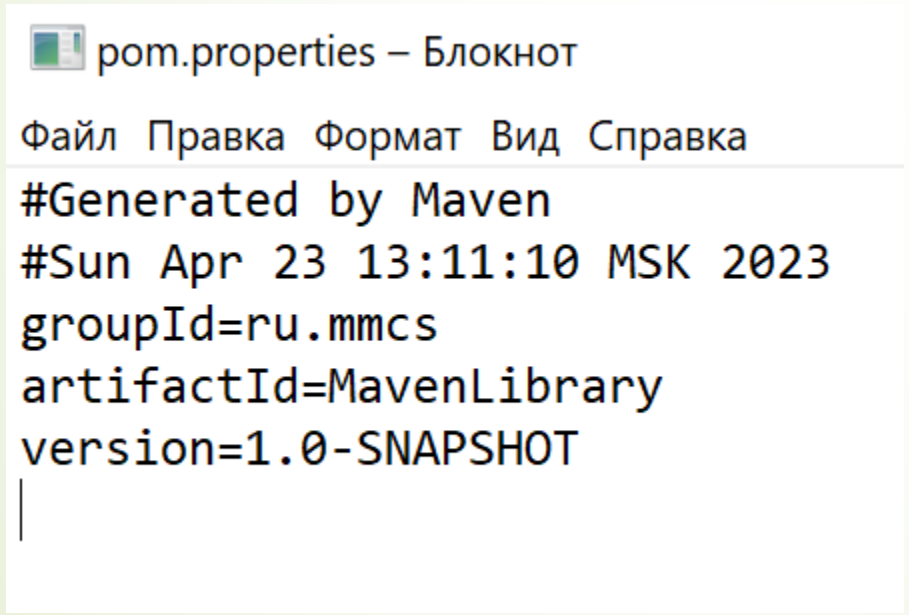
Среда разработки NetBeans

- Создадим библиотеку классов для сборки Maven



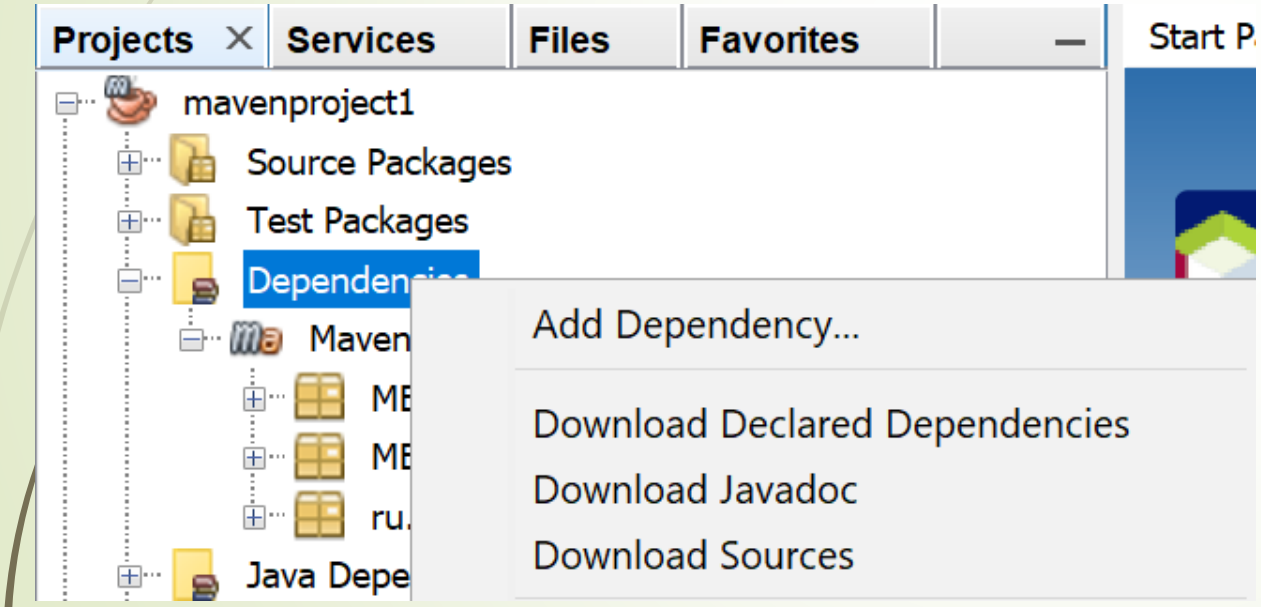
Среда разработки NetBeans

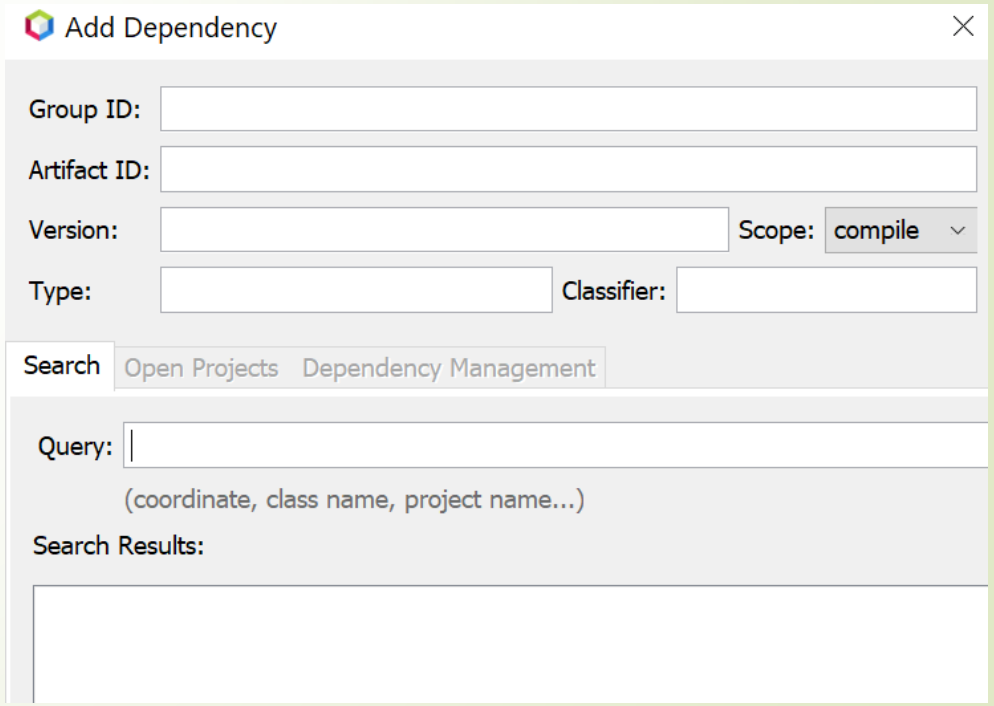
- \MavenLibrary-1.0-SNAPSHOT.jar\META-INF\maven\ru.mmcs\MavenLibrary\



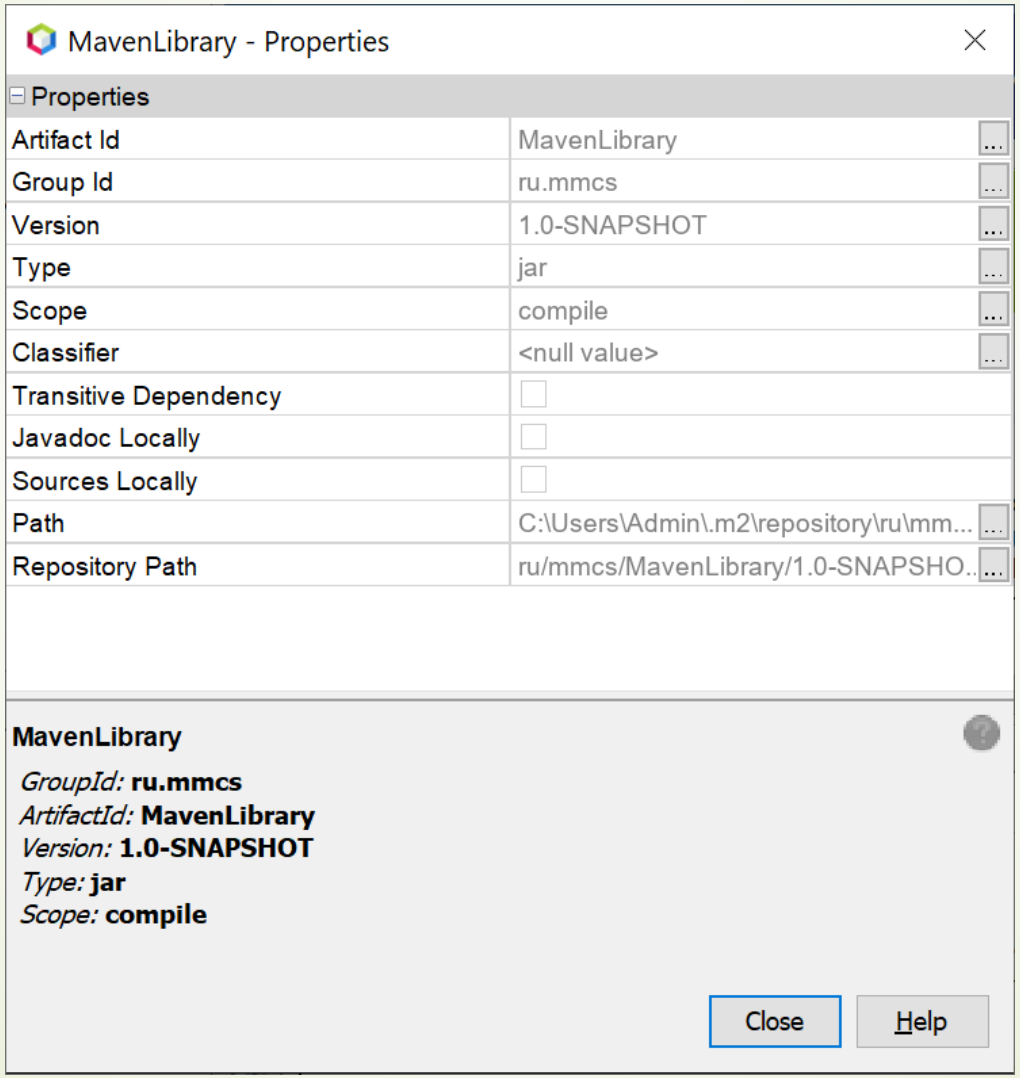
Среда разработки NetBeans

- Подключение библиотеки в jar-файле к Maven проекту





Среда разработки NetBeans



Среда разработки Intellij IDEA

Создать jar

- *Project Structure -> Artifacts -> Jar -> From modules with dependencies*
- *В диалоговом окне выбрать главный файл проекта +OK*
- *Build -> Build Artifact*

Intellij IDEA

Добавить jar в проект

- *File->Project Structure...->Modules->*

  Dependencies > '+' > JARs or directories…