

## ЛАБОРАТОРНАЯ РАБОТА № 2. ПАКЕТИРОВАНИЕ И РАСПРОСТРАНЕНИЕ ПРИЛОЖЕНИЙ JAVA

### Цель работы:

1. Изучить способы компиляции и запуска приложений Java вне IDE
2. Получить практические навыки создания исполняемого файла JAR
3. Получить практические навыки создания exe-файла для приложения Java с помощью программы launch4j

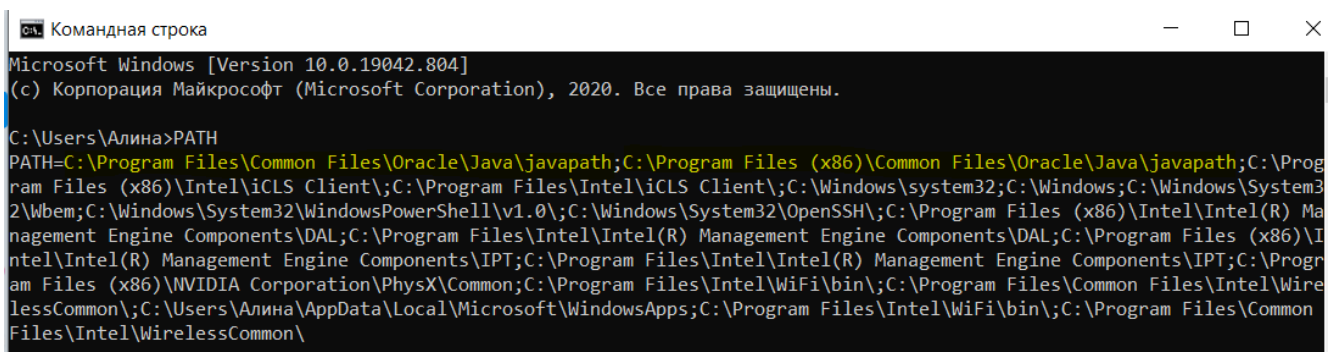
### Теоретические сведения

Существует несколько способов запуска программ на Java:

- Через IDE
- Вызов приложения из командной строки
- Вызов приложения из файла сценариев
- Запуск исполняемого JAR-файла
- Создание exe-файла приложения из JAR-файла

### 1. Компиляция и вызов приложения из командной строки

Независимо от того, какую операционную систему вы используете, если на вашем компьютере установлен JDK или JRE, то можно откомпилировать, а после и запустить приложение на Java из командной строки. Для этих целей используются программы `javac.exe` (компилятор) и `java.exe` (JVM). Эти файлы можно запускать по имени, без указания пути (но перед запуском убедитесь, что пути к ним прописаны в переменной среды `PATH` вашей операционной системы). В Windows это можно проверить, написав в командной строке слово `PATH`.



```
Командная строка
Microsoft Windows [Version 10.0.19042.804]
(c) Корпорация Майкрософт (Microsoft Corporation), 2020. Все права защищены.

C:\Users\Алина>PATH
PATH=C:\Program Files\Common Files\Oracle\Java\javapath;C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Program Files (x86)\Intel\iCLS Client\;C:\Program Files\Intel\iCLS Client\;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common;C:\Program Files\Intel\WiFi\bin\;C:\Program Files\Common Files\Intel\WirelessCommon\;C:\Users\Алина\AppData\Local\Microsoft\WindowsApps;C:\Program Files\Intel\WiFi\bin\;C:\Program Files\Common Files\Intel\WirelessCommon\
```

Путь к этим файлам также можно указать непосредственно в командной строке.

```
c:\>"c:\Program Files\Java\jdk-15.0.2\bin\java.exe"
```

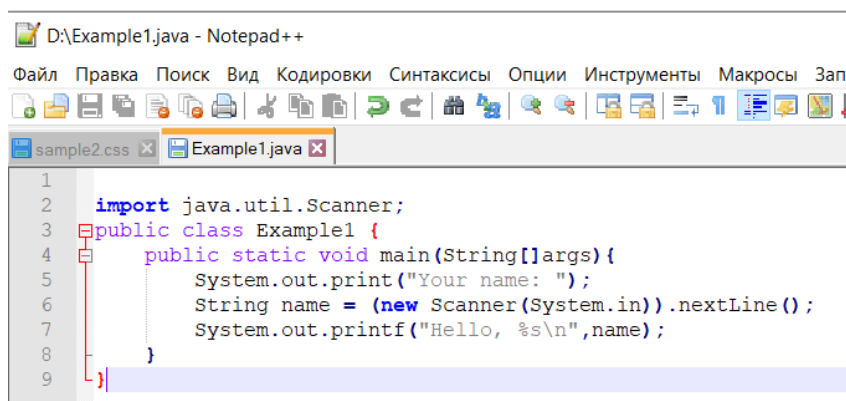
**Важно**, чтобы установленные версии JDK (JRE), совпадали с версией JDK вашего приложения!

Следует также отметить, что при компиляции – необходимо в команде **javac.exe** в качестве параметра передавать *имя соответствующего файла* (с расширением `.java`)! А для

запуска программы, используется программа **java.exe**, которой в качестве параметра указывается имя главного класса (не файла!!!) приложения.

### Пример 1. *Компиляция и выполнения одиночного класса Java*

Создадим (например, в блокноте) на диске D: файл Example1.java, содержащий следующий код:



```
D:\Example1.java - Notepad++
Файл  Правка  Поиск  Вид  Кодировки  Синтаксисы  Опции  Инструменты  Макросы  Запу
sample2.css  Example1.java
1
2  import java.util.Scanner;
3  public class Example1 {
4      public static void main(String[] args){
5          System.out.print("Your name: ");
6          String name = (new Scanner(System.in)).nextLine();
7          System.out.printf("Hello, %s\n",name);
8      }
9  }
```

Для его компиляции в командной строке вызовем следующую команду (при этом указывается путь к файлу (расширение .java обязательно!)):

```
c:\>javac.exe d:\Example1.java
```

По умолчанию, откомпилированные классы (.class) сохраняются в той же папке, что и исходные классы на java.

Таким образом, откомпилированный файл Example1.class будет создан в корневом каталоге диска D:.

Для запуска этого файла на выполнения вызовем программу java.exe (указываем только имя класса, без расширения!!!).

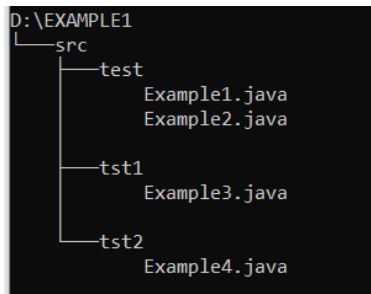
```
c:\>java.exe -cp d:\ Example1
Your name: Jack
Hello, Jack
```

Если текущий каталог не совпадает с папкой размещения запускаемого файла, следует указать дополнительный параметр **-cp** (или **-classpath**) и **путь к папке, содержащей указанный класс**. На рисунке – это диск d:\, в противном случае, можно запустить файл непосредственно по имени, без дополнительных параметров.

```
D:\>java.exe Example1
Your name: Sasha
Hello, Sasha
```

## Пример 2. Компиляция и выполнения приложения на Java, состоящего из нескольких пакетов и классов

Пусть приложение Java сохранено в папке D:\Example1, которая содержит подпапку *src*, состоящую из нескольких пакетов. В каждом из пакетов имеются исходные файлы на java. Структура проекта представлена на рисунке.



Откомпилированные файлы с байт-кодом будем сохранять в папке D:\Example1\out. При компиляции следует указать:

- 1) Путь к папке с исходными кодами (параметр **-sourcepath**)
- 2) пути к классам во всех пакетах (несколько путей разделяются пробелами)
- 3) путь для сохранения байт-кодов классов (параметр **-d**)

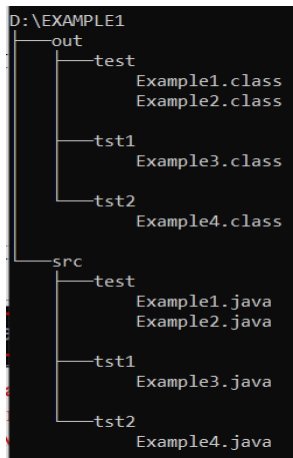
Для удобства работы перейдем в папку приложения D:\Example1, выполнив команду:

```
cd D:\Example1
```

Далее выполним компиляцию проекта следующей командой:

D:\Example1>javac -sourcepath src src/test/*.java src/tst1/*.java src/tst2/*.java -d out			
Путь к папке с исходными кодами	Пути к исходным кодам классов приложения <i>Если в приложение нужно также включить байт-коды классов или jar-файлы, то пути к ним следует указать в параметре -cp (-classpath)</i>		Путь к папке для откомпилированных классов

В результате выполнения предыдущей команды, в папке проекта появится подпапка **out**. Новая структура приведена на рисунке:



Как видим из рисунка, в папке out созданы папки для всех пакетов приложения. Для запуска приложения воспользуемся программой java.exe, указав 2 аргумента:

- 1) путь к папке с байт-кодами классов (параметр **-cp** или **-classpath**, в примере: out)
- 2) название главного класса, начиная с имени пакета (в примере: test.Example1).

```
D:\Example1>java -cp out test.Example1
```

#### ПРИМЕЧАНИЕ:

Часто в ОС Windows возникают проблемы с отображением русских букв. Для их устранения следует воспользоваться следующей командой:

CHCP 65001	- установить кодировку UTF-8
CLS	- очистить экран

Команда CHCP меняет текущую кодовую страницу командной консоли на время работы текущего окна.

## 2. Вызов приложения из файла сценариев

Чтобы не писать каждый раз команду запуска для консольных приложений на Java удобно сохранить ее в пакетном файле. В Windows пакетный файл имеет расширение .bat. Такой файл можно создать в любом текстовом редакторе. На рисунке ниже приведен пример файла run2.bat, созданный в программе Notepad++. Путь в файле – относительный, т.е. зависит от расположения самого пакетного файла run2.bat.

Для запуска приложения достаточно запустить файл run2.bat (например, двойным щелчком).

```
run2.bat
1 @echo off
2 rem отключаем вывод комментариев среды
3
4 rem указываем путь к папке с пакетами приложения
5 rem это путь относительно расположения создаваемого пакетного файла
6 rem в этом примере, .bat файл размещен в корневой папке приложения
7 set programpath="out/production/LocalDate_test"
8
9 rem указываем путь к главному классу приложения
10 set mainclass="lab1.MainClass"
11
12 rem запускаем программу
13 java -cp %programpath% %mainclass%
14
15 rem останавливаем выполнение до нажатия любой клавиши
16 pause
```

## 3. Запуск исполняемого JAR-файла приложения

Для распространения приложения в среде IDE и его последующего запуска вне среды IDE, следует упаковать его в исполняемый файл **JAR**.

Файл JAR – это файл архива, в который могут быть вложены файлы и папки. Файлы JAR подобны файлам ZIP, но файлы JAR могут иметь дополнительные атрибуты, полезные при распространении приложений Java. В число этих атрибутов входят цифровая подпись файлов JAR, дополнительное сжатие, совместимость с различными платформами и т. д.

**Отметим, главное отличие jar-архива от zip-архива – наличие файла манифеста. В манифесте указывается главный класс, который будет запускаться при выполнении jar-файла, а также параметр classpath.**

JAR-файл можно создать средствами JDK или средствами среды программирования. Рассмотрим оба варианта.

### 3.1. Создание исполняемого файла JAR без IDE через командную строку.

Для создания JAR-файла приложения воспользуемся утилитой `jar.exe` из комплекта разработчика. По умолчанию путь к этой утилите не прописывается в переменной среды `PATH`. Можно этот путь добавить в `PATH` и вызывать утилиту просто по имени. Или придется указывать полный путь к ней (располагается в папке `bin`<sup>1</sup> комплекта разработчика).

Перед началом работы следует создать файл манифеста `manifest.mf`. Его содержимое включает строки вида:

**параметр: значение**

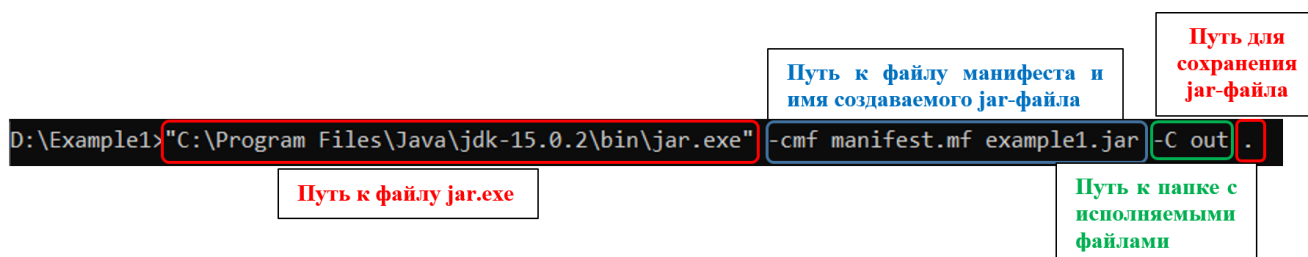
Некоторые параметры файла манифеста:

- **main-class** – указывает класс, содержащий метод `main`, который будет выполнен при запуске приложения
- **class-path** – путь к скомпилированным классам или дополнительным библиотекам

Для создания jar-файла приложения следует в команде `jar.exe` указать следующие параметры:

- 1) Путь к файлу манифесту и название выходного jar-файла (параметр **-cmf**)
- 2) Путь к откомпилированным классам (параметр **-C**)
- 3) Путь, куда будет сохранен jar-файл

Применительно к приложению из примера 2, команда создания jar-файла будет выглядеть следующим образом:



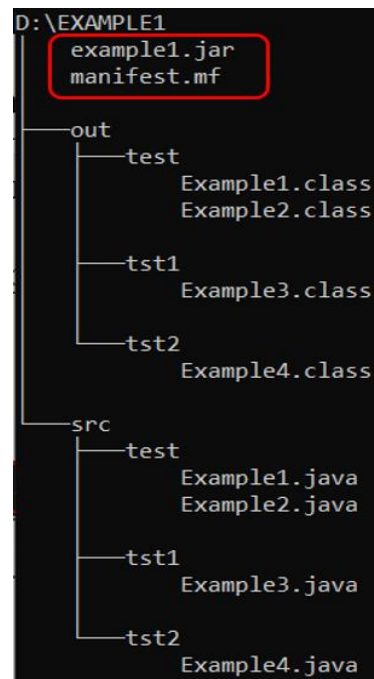
<sup>1</sup> По умолчанию, jdk устанавливается в папку `"C:\Program Files\Java\jdk-версия"` или `"C:\Program Files (x86)\Java\jdk-версия"` в зависимости от разрядности операционной системы Windows.

В команде полагается, что файл манифеста располагается в текущей папке (т.е. D:\Example1). JAR-файл должен быть создан также в этой папке (.).

Файл манифеста должен быть создан заранее. Содержимое файла **manifest.mf** данного проекта:

```
main-class: test.Example1
class-path: out/
```

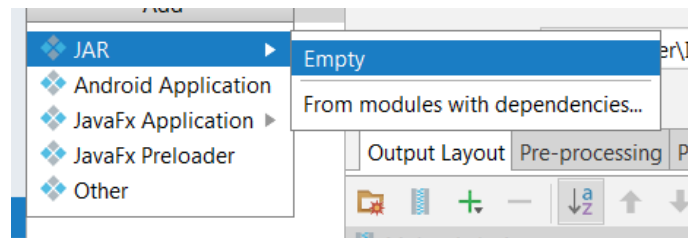
В результате выполнения приведенной команды получим следующую структуру приложения:



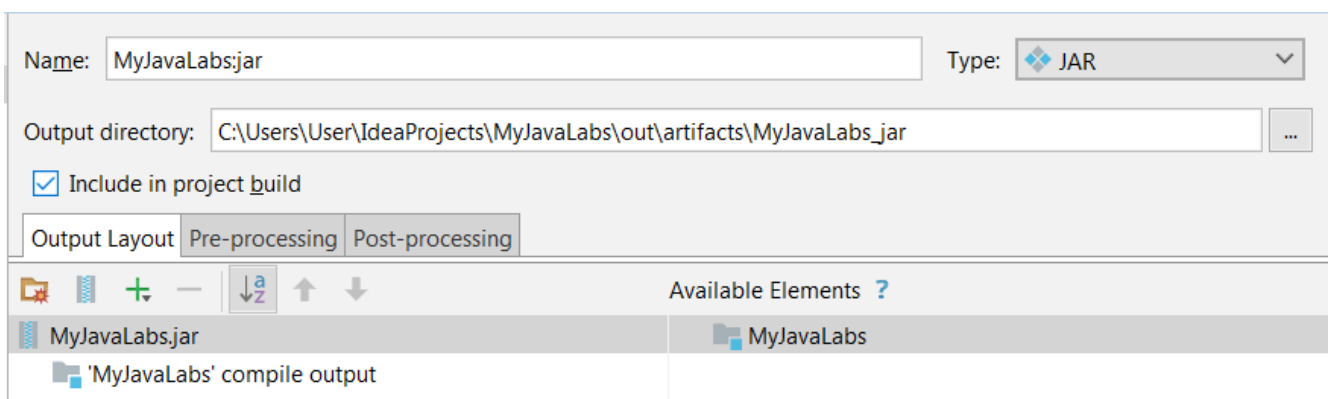
### 3.2.Создание исполняемого файла JAR в IDE IntelliJ IDEA

Для создания JAR-файла приложения в среде IntelliJ IDEA требуется:

1. Открыть меню **File / Project Structure**
2. Выбрать вкладку **Artifacts**
3. Нажать на кнопку +.
4. Далее выбрать пункт **JAR \ From modules with dependencies**



5. В появившемся окне следует указать папку, в которую будет помещена папка artifacts (например, в папке **out** приложения). Установите флаги **Include in project build** и **Show content of elements**. И сохраните сделанные установки.



6. В среде IntelliJ IDEA требуется скомпилировать проект. Для этого в меню **Build** выполните пункт **Build project**.

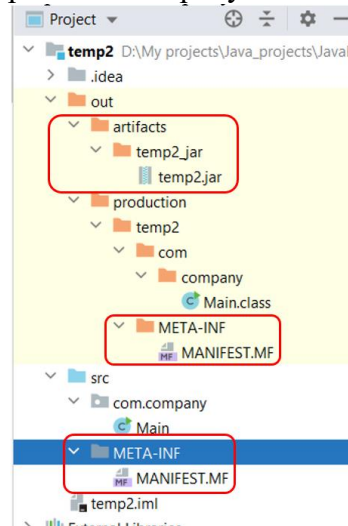
При сборке проекта среда IDE создает файл JAR и включает в него манифест.

Файл манифеста MANIFEST.MF является стандартной частью архива JAR, содержащей информацию о файле JAR, которая будет полезна для средства запуска java при запуске приложения.

Если все сделано корректно, то JAR-файл будет отображаться в структуре проекта. Также в проект будет добавлена META-информация (а именно, файл манифеста MANIFEST.MF) и в папку src, и в папку out.

При назначении главного класса проекта мы убеждаемся, что главный класс указан в манифесте.

Примерная структура проекта приведена на рисунке.



Информация из файла манифеста показана на следующем рисунке



### 3.3. Запуск исполняемого файла JAR

JAR-файл также можно запустить из командной строки программой java.exe или из файла сценариев. Для этого в программе java.exe указывается дополнительный параметр -jar.

*Синтаксис:*

**java.exe -jar <путь к JAR-файлу>**

```
C:\Users\Алина>java.exe -jar "D:\My projects\Java_projects\LocalDate_test\out\artifacts\LocalDate_test_jar\LocalDate_test.jar"
Введите дату (и/или время):
12.02.2002
this is date
День месяца: 12
День месяца: 12
День недели: TUESDAY
День года: 43
Месяц: FEBRUARY
Год: 2002
Неделя года: 7
```

Примерный вид файла сценариев для запуска JAR-файла представлен на рисунке ниже.

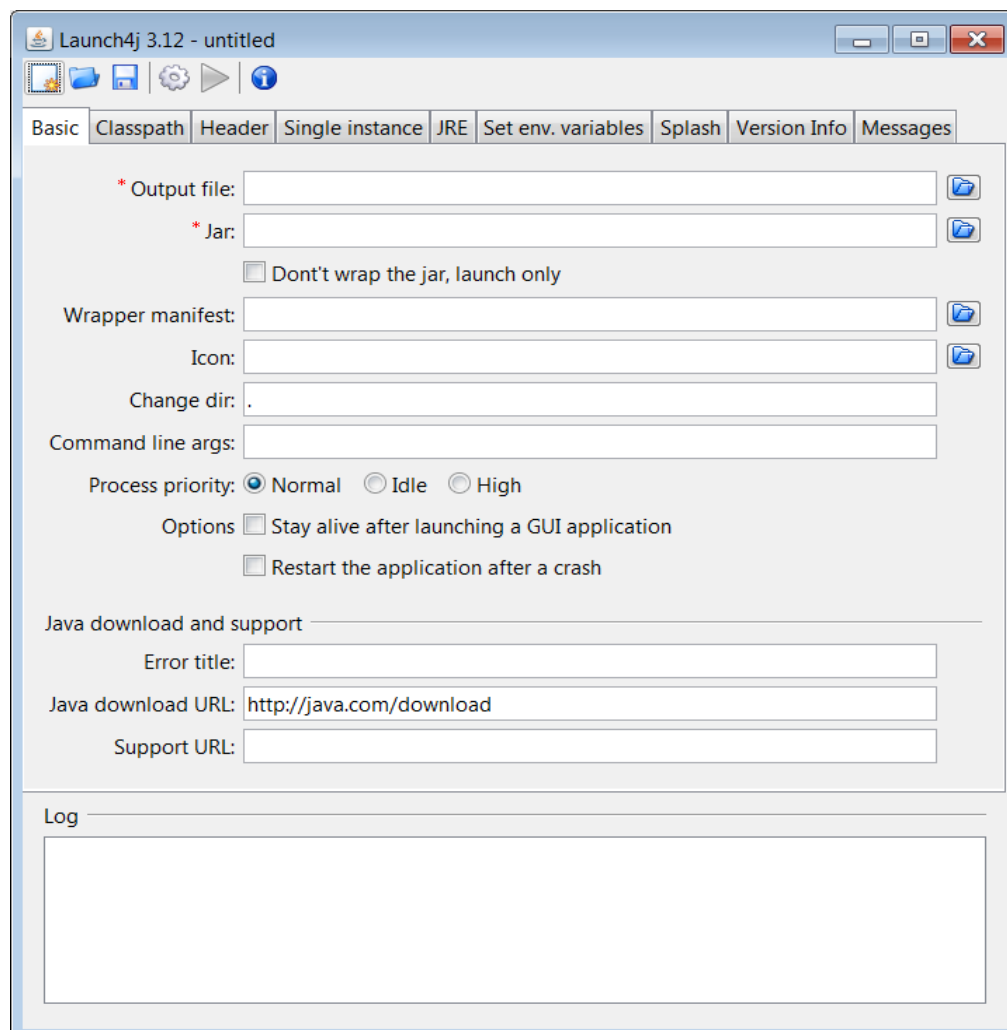
```
run_jar.bat x
1 @echo off
2 set jar_path="out/artifacts/LocalDate_test_jar/LocalDate_test.jar"
3 java -jar %jar_path%
4 pause
```

Для запуска своего приложения измените переменную `jar_path`, указав путь к своему JAR-файлу.

## 4. Создание EXE-файла приложения из JAR-файла

Пользователям Windows привычнее использовать исполняемое приложение в виде exe-файла, нежели архивного jar-файла. Для этого предназначена программа `launch4j` (доступна на сайте: <https://sourceforge.net/projects/launch4j/>). Разработчики настольных java-приложений могут плагином **launch4j** не только обернуть исполняемый архивный jar-файл в оболочку exe-файла, но и включить в него иконку, автора, версию. Также данный плагин позволяет определить минимальную версию используемой JRE.

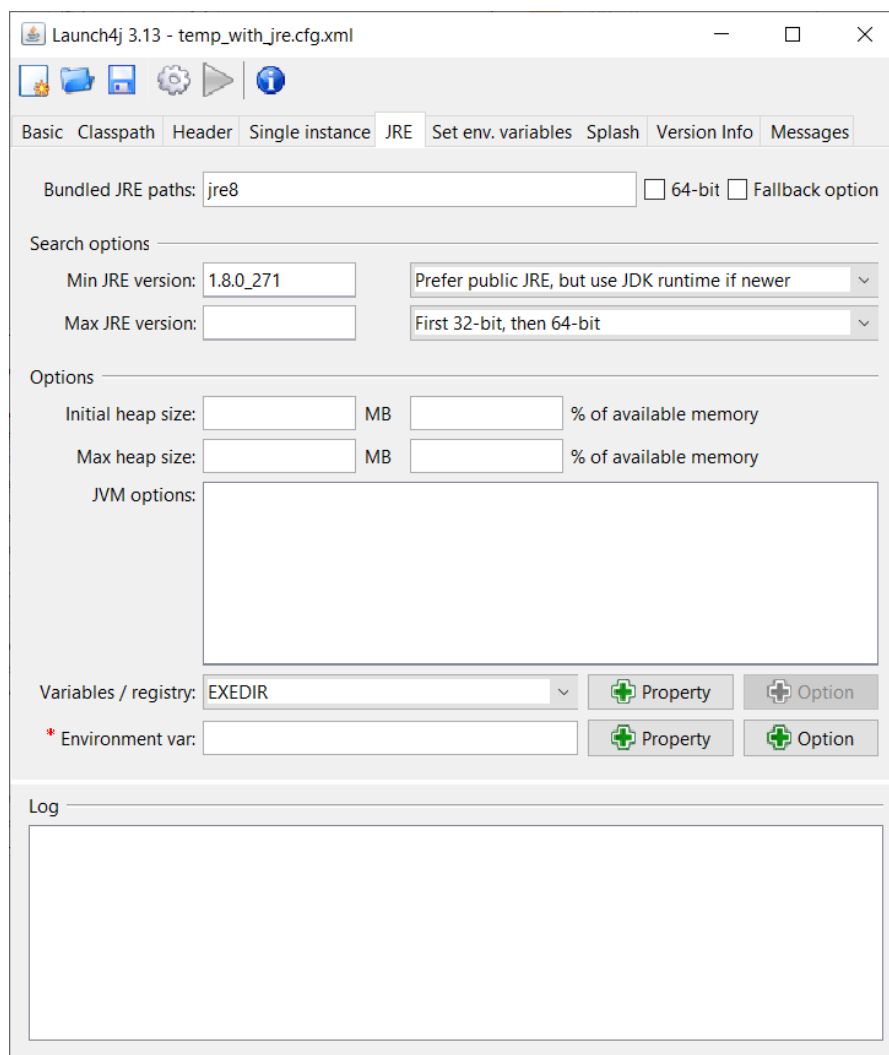
На следующем рисунке представлен скриншот программы `launch4j`.





Для создания ехе-файла из jar в среде launch4j, выполните следующие шаги:

1. На вкладке *Basic* указываются два обязательных параметра:
  - a. Полное имя выходного ехе-файла (который будет создан)
  - b. Путь к JAR-файлу приложения
  - c. Также здесь можно выбрать, например, иконку приложения
2. На вкладке Header укажите тип приложения: GUI или Console
3. На вкладке JRE (см. рисунок) нужно указать версии JRE, под которыми возможен запуск приложения и разрядность ОС.



В данном окне можно указывать следующую информацию:

- **Bundled JRE paths:** Путь к папке с JRE, относительно EXE-файла. Данный пункт используется, если планируется поставлять ехе-файл вместе с нужной версией JRE. В этом случае папка с JRE копируется в папку с ехе-файлом (эту папку можно переименовать). В приведенном рисунке эта папка называется jre8
- **Min JRE version, Max JRE version:** Минимальная и максимальная версии JRE, под которыми программа будет выполняться.
- **Prefer public JRE, but use JDK runtime if newer:** если на компьютере установлены и JRE и JDK разных версий, то предпочтительнее запускать под JDK, если он новее
- **Prefer JDK runtime, but use public JRE if newer:** если на компьютере установлены и JRE и JDK разных версий, то предпочтительнее запускать под JRE, если она новее

- и т.д.



После установки нужных параметров, нажмите на кнопку **Build wrapper**, на панели инструментов. Внизу окна в поле **Log** отобразится информация о ходе процесса. Если процесс прошел без ошибок, то ехе-файл будет создан по указанному ранее пути.

## 5. Системы сборки

На первый взгляд, умение компиляции и сборки кода без IDE кажется бесполезным. Действительно, зачем мучиться с командными строками и гуглить все команды, когда есть IDE с плагинами, автопроверкой всего что можно и интеграцией с популярными системами. Однако практика показывает, что умение собирать код без среды разработки и понимание каждого шага этого процесса — суровая необходимость. Этот навык сэкономит вам немало нервных клеток и времени вам и вашей компании.

Также следует отметить, что на сегодняшний день имеются системы сборки, которые облегчают процесс работы. Несколькими командами эти системы могут собрать проекты любой сложности. Самые известные системы сборки на Java — это Ant, Maven и Gradle. Каждая из этих систем создана для решения определенных задач [Дополнительную информацию об указанных системах сборки можно посмотреть по адресу <https://javarush.ru/groups/posts/2318-kompiljacija-v-java>].

### Задание на лабораторную работу

1. Создайте java-файл в текстовом редакторе. Откомпилируйте и запустите его в командной строке Windows.
2. Скомпилируйте файлы проекта любой из предыдущих лабораторных работ посредством командной строки
3. На примере проекта из предыдущей лабораторной работы, продемонстрируйте различные варианты запуска приложения вне IDE:
  - a. из командной строки по имени класса
  - b. посредством файла сценариев
4. Создайте JAR-файл для вашего приложения:
  - a. Через утилиту jar.exe
  - b. Средствами IntelliJ IDEA
5. Выполните запуск JAR-файла из командной строки и посредством файла сценариев
6. Создайте ехе-файл вашего приложения в программе launch4j
  - a. Без встроенной JRE
  - b. С встроенной JRE

### Контрольные вопросы

1. Компиляция файлов приложения из командной строки
2. Перечислите варианты запуска приложений на Java без использования IDE
3. Запуск приложения через командную строку
4. Создание JAR-файла приложения
5. Запуск JAR-файла
6. Создание ехе-файла приложения