Шаг 0. Проверка версии

```
%>java -version


%>javac -version
```

Шаг 1. Компиляция и запуск в корне проекта

1. Создать каталог для работы lab1.
2. Внутри каталога создать каталог step1 для шага 1. (в проводнике или, например, с помощью команды **md <имя каталога>**)
3. Создать в каталоге step1 файл **Hello.java** со следующим кодом:

```
public class Hello {

    public static void main(String args[]) {

        System.out.println("Hello! Finished Step 1");

    }

}
```

4. В итоге получим следующую структуру каталогов:

```
lab1/

    step1/

        Hello.java
```

5. В консоли перейти в корень проекта для шага 1 (lab1/step1) (команда **cd <путь к каталогу>**)
6. С помощью утилиты **javac** скомпилировать модуль с исходным кодом Hello.java

```
%lab1\step1>javac Hello.java
```

7. Затем с помощью утилиты **java** запустить полученный после компиляции модуль с байт-кодом Hello.class.

```
%lab1\step1>java Hello

Hello! Finished step1
```

Шаг 2. Компиляция и запуск в папках src и bin

1. Создать следующую структуру каталогов и файлов:

```
lab1/
      step2/
            src/ – для модулей с исходным кодом
                  Hello.java
            bin/ – для модулей с байт-кодом
```

Hello.java

```
public class Hello {
      public static void main(String args[]) {
            System.out.println("Hello! Finished Step 2");
      }
}
```

2. В консоли перейти в корень проекта для шага 2 (lab1/step2)
3. Скомпилировать класс Hello, направив вывод компилятора в каталог bin.
4. Запустить класс Hello.

Шаг 3. Компиляция и запуск двух связанных классов

1. Создать следующую структуру каталогов и файлов:

```
lab1/
    step3/
        src/ – для модулей с исходным кодом
            Hello.java
            Dog.java
        bin/ – для модулей с байт-кодом
```

Hello.java

```
public class Hello {
    public static void main(String args[]) {
        System.out.println("Hello!");
        Dog mydog = new Dog();
        System.out.println("Dog: " + mydog.voice());
        System.out.println("Finished Step 3");
    }
}
```

Dog.java

```
public class Dog {
    public String voice() {
        return "Gav-gau!";
    }
}
```

2. В консоли перейти в корень проекта для шага 3 (lab1/step3)
3. Скомпилировать класс Hello вместе со связанным с ним классом Dog, направив вывод компилятора в каталог bin.
4. Запустить класс Hello.

Шаг 4. Компиляция и запуск с подключением класса из внешнего каталога

1. Создать следующую структуру каталогов и файлов:

```
lab1/
    external/ - для модулей с байт-кодом из внешнего проекта
        ExternalSummator.class
    step4/
        src/ – для модулей с исходным кодом
            Hello.java
            Dog.java
        bin/ – для модулей с байт-кодом
```

Hello.java

```
public class Hello {
    public static void main(String args[]) {
        System.out.println("Hello!");
        Dog mydog = new Dog();
        System.out.println("Dog: " + mydog.voice());
        System.out.println("2 + 2 = " + ExternalSummator.sum(2,2));
        System.out.println("Finished Step 4");
    }
}
```

Dog.java (Без изменений)

ExternalSummator.java

```
public class ExternalSummator {
    public static int sum(int a, int b) {
        return a+b;
    }
}
```

2. Скомпилировать класс ExternalSummator и поместить полученный модуль с байт-кодом в каталог lab1/external

3. В консоли перейти в корень проекта для шага 4 (lab1/step4)

4. Скомпилировать класс Hello вместе со связанным с ним классом Dog и с подключением внешнего класса ExternalSummator, направив вывод компилятора в каталог bin.

5. Запустить класс Hello.

дополнительно: найти различные варианты установки classpath

Шаг 5. Компиляция и запуск с подключением jar-файла из каталога lib проекта

1. Создать следующую структуру каталогов и файлов:

```
lab1/
    external/ - для модулей с байт-кодом из внешнего проекта
        ExternalSummator.class
    step5/
        src/ – для модулей с исходным кодом
            Hello.java
            Dog.java
        lib/ - для jar-библиотек
            secret.jar
        bin/ – для модулей с байт-кодом
```

Hello.java

```
public class Hello {
    public static void main(String args[]) {
        System.out.println("Hello!");
        Dog mydog = new Dog();
        System.out.println("Dog: " + mydog.voice());
        System.out.println("2 + 2 = " + ExternalSummator.sum(2,2));
        System.out.println("Secret code: " + SecretSafe.SECRET_CODE);
        System.out.println("Finished Step 5");
    }
}
```

Dog.java (Без изменений)

ExternalSummator.java (Без изменений)

secret.jar (Взять с сайта)

2. В консоли перейти в корень проекта для шага 5 (lab1/step5)
3. Скомпилировать класс Hello вместе со связанным с ним классом Dog и с подключением внешнего класса ExternalSummator и с подключением jar-файла secret.jar, направив вывод компилятора в каталог bin.
4. Запустить класс Hello.

Шаг 6. Создание командного файла для запуска приложения

1. Создать следующую структуру каталогов и файлов:

```
lab1/
    external/ - для модулей с байт-кодом из внешнего проекта
        ExternalSummator.class
    step6/
        start.bat – командный файл
        src/ – для модулей с исходным кодом
            Hello.java
            Dog.java
        lib/ - для jar-библиотек
            secret.jar
        bin/ – для модулей с байт-кодом
```

Hello.java

```
public class Hello {
    public static void main(String args[]) {
        System.out.println("Hello!");
        Dog mydog = new Dog();
        System.out.println("Dog: " + mydog.voice());
        System.out.println("2 + 2 = " + ExternalSummator.sum(2,2));
        System.out.println("Secret code: " + SecretSafe.SECRET_CODE);
        System.out.println("Finished Step 6");
    }
}
```

Dog.java (Без изменений)

ExternalSummator.java (Без изменений)

secret.jar (взять с сайта)

2. В консоли перейти в корень проекта для шага 6 (lab1/step6)

3. Скомпилировать класс Hello вместе со связанным с ним классом Dog и с подключением внешнего класса ExternalSummator и с подключением jar-файла secret.jar, направив вывод компилятора в каталог bin.

4. Создать командный файл start.bat в корне проекта для шага 6 (lab1/step6)

5. Запустить проект с помощью командного файла.

дополнительно: создать командный файл с установкой переменной CLASSPATH

Шаг 7. Упаковка модулей с байт-кодом в jar-файл

1. Создать следующую структуру каталогов и файлов:

```
lab1/
    step7/
        hello.jar – jar-файл проекта
        src/ – для модулей с исходным кодом
            Hello.java
            Dog.java
        lib/ - для jar-библиотек
            secret.jar
        bin/ – для модулей с байт-кодом
            Hello.class
            Dog.class
            ExternalSummator.class
```

Hello.java

```
public class Hello {
    public static void main(String args[]) {
        System.out.println("Hello!");
        Dog mydog = new Dog();
        System.out.println("Dog: " + mydog.voice());
        System.out.println("2 + 2 = " + ExternalSummator.sum(2,2));
        System.out.println("Secret code: " + SecretSafe.SECRET_CODE);
        System.out.println("Finished Step 7");
    }
}
```

Dog.java (Без изменений)

ExternalSummator.java (Без изменений)

secret.jar (взять с сайта)

2. В консоли перейти в корень проекта для шага 7 (lab1/step7)

3. Скомпилировать класс Hello вместе со связанным с ним классом Dog и классом ExternalSummator (ExternalSummator.class перенести из внешнего каталога в bin! он будет сохранен в jar-архив, т.к. jar-архивы должны содержать только внешние зависимости от других jar-файлов (например,

secret.jar), а не зависимости от отдельных внешних каталогов с модулями байт-кода).

4. С помощью утилиты **jar** создать jar-файл проекта hello.jar в корне каталога lab1/step7/, в который упаковать модули байт-кода.

Шаг 8. Создание запускаемого jar-файла

Исходные данные – получены на шаге 7

1. Сделать jar-файл, созданный на шаге 7,- запускаемым путем модификации файл MANIFEST.MF внутри упакованного jar-архива **hello.jar**

2. Запустить jar-файл из консоли с помощью утилиты **java**

Дополнительно

Выполнить все шаги работы еще в одной операционной системе (например, в Linux).