

实验说明

一、云平台

登陆网址: <https://console-edu.ucloud.cn/uhost/uhost>

用户名: 学号

密码: 无 (登陆后需修改密码)

二、实验组成

本课程实验主要分为单机实验、分布式实验和编程实验三个部分:

1. 单机实验只需要一台机器即可完成, 可选的环境有:
 - ① 云平台上的 linux 云主机;
 - ② 个人 macOS 计算机;
 - ③ 个人 windows 计算机上的 linux 虚拟机。(可参考[教程](#)在 vmware 中安装 linux 虚拟机)
2. 分布式实验需要多台机器配合完成, 建议使用 linux 云主机进行实验。
3. 编程实验中代码的编写需要在本地 (个人计算机) 完成。(建议使用 windows 系统的同学安装 linux 虚拟机进行实验)

实验一 部署环境准备

1.1 实验目的

- 掌握 Linux 系统免密钥登录的方法
- 学会 Linux 系统中 Java 环境的安装与配置

1.2 实验任务

- 实现本机以及多主机之间的免密钥登录
- 完成 Java 环境的安装与配置

1.3 实验环境

- Ubuntu 操作系统

1.4 实验步骤

1.4.1 本机免密钥登录

(1) ssh 免密钥登录本机的安装和配置

a) 安装 openssh-server

```
1 sudo apt-get install openssh-server #安装ssh-server
```

b) 生成本机公钥

```
1 cd ~  
2 mkdir .ssh #可能提示该文件已存在，不影响其他操作  
3 cd ~/.ssh/  
4 ssh-keygen -t rsa # 生成公钥私钥对，对于系统提示，按回车即可
```

通过上述操作生成的密钥文件所在位置: `~/.ssh/`，用户公钥的位置：“`~/.ssh/id_rsa.pub`”，
用户私钥的位置：“`~/.ssh/id_rsa`”。

c) 将本机公钥发送给本机需要免密 ssh 访问的节点（即本机）

```
1 cat id_rsa.pub >> authorized_keys # 查看公钥内容是否追加写到授权文件
```

d) 执行以下命令来修改授权文件及其所在文件夹的权限

```

1 chmod 600 ~/.ssh/authorized_keys
2 chmod 700 -R ~/.ssh

```

(2) 验证本机免密钥登录是否配置成功

执行以下命令，若没有出现需要输入密码的情况，则代表配置成功。

```
1 ssh localhost
```

1.4.2 多台机器之间免密钥登录

假设现有四台机器，要使得四台机器能够通过“ssh”命令实现彼此之间免密钥登录，执行如下操作：

- (1) 在每台机器上创建一个同名的用户“dase-dis”，此后所有操作都在此用户名下，软件也安装在此用户目录下
- (2) 在每台机器上的“dase-dis”用户下实现本机免密钥登录（参考上文“本机免密钥登录”相关内容）
- (3) 分别修改四台机器的主机名

a) 在第 1 台主机上执行以下命令

```
1 sudo hostnamectl set-hostname ecnu01
```

b) 在第 2 台主机上执行以下命令

```
1 sudo hostnamectl set-hostname ecnu02
```

c) 在第 3 台主机上执行以下命令

```
1 sudo hostnamectl set-hostname ecnu03
```

d) 在第 4 台主机上执行以下命令

```
1 sudo hostnamectl set-hostname ecnu04
```

- (4) 在每台机器上修改/etc/hosts 文件，添加主机名“ecnu01”、“encu02”、“ecnu03”、“ecnu04”和对应的 IP 地址之间的映射，使得每台机器都可以通过主机名登录其他机器。具体操作如下：

a) hosts 配置文件路径：/etc/hosts，该配置文件的作用为将一些常用的网址域名与其对应的 IP 地址关联起来。当用户在浏览器中输入一个需要登录的网址时，系统会首先从 hosts 文件中寻找对应的 IP 地址，一旦找到，系统就会立即打开对应网

页，如果没有找到，则系统会将网址提交 DNS 域名解析服务器进行 IP 地址的解析

b) 执行以下命令，打开 hosts 文件

```
1 sudo vim /etc/hosts
```

c) 打开文件后，按 “i” 键进入编辑模式，添加映射信息，映射信息如以下列表所示。映射信息添加完成后，按 “Esc” 键退出编辑模式，并输入 “:wq!” 保存并退出

```
1 #IP地址 主机名
2 10.24.15.11 ecnu01
3 10.24.15.12 ecnu02
4 10.24.15.13 ecnu03
5 10.24.15.14 ecnu04
6 # 注意，以上10.24.15.11、10.24.15.12、10.24.15.13、10.24.15.14
    四个IP地址仅作示范，请分别替换为四台机器实际的IP地址
```

(5) 将主机名为 “ecnu02”、“ecnu03”、“ecnu04” 机器上的公钥发送给主机名为 “ecnu01” 机器上的授权文件。即分别切换到主机名为 “ecnu02”、“ecnu03”、“ecnu04”的机器上执行命令（共执行三次）

```
1 cat ~/.ssh/id_rsa.pub | ssh dase-dis@ecnu01 'cat - >> ~/.ssh/authorized_keys'
```

(6) 将主机名为 “ecnu01” 机器上的授权文件（包含 “ecnu02”、“ecnu03”、“ecnu04”的公钥）发送到主机名为 “ecnu02”、“ecnu03”、“ecnu04”的机器上。即切换到主机名为 “ecnu01”的机器上执行命令，覆盖原有的授权文件

```
1 scp ~/.ssh/authorized_keys dase-dis@ecnu02:/home/dase-dis/.ssh/authorized_keys
2 scp ~/.ssh/authorized_keys dase-dis@ecnu03:/home/dase-dis/.ssh/authorized_keys
3 scp ~/.ssh/authorized_keys dase-dis@ecnu04:/home/dase-dis/.ssh/authorized_keys
```

(7) 验证四台机器是否彼此之间都可以免密钥登录。在每台机器上执行以下命令，若没有出现需要输入密码的情况，则代表配置成功

```
1 ssh dase-dis@ecnu01
2 exit
3 ssh dase-dis@ecnu02
4 exit
5 ssh dase-dis@ecnu03
6 exit
7 ssh dase-dis@ecnu04
8 exit
```

1.4.3 Java 环境配置

(1) 准备工作

在用户目录下下载 jdk1.8 并解压：

```
1 cd /home/dase-local/
2 wget https://repo.huaweicloud.com/java/jdk/8u171-b11/jdk-8u171-linux-x64.tar.gz
3 tar -zxvf jdk-8u171-linux-x64.tar.gz #解压下载好的安装包
4 sudo mv jdk1.8.0_171 /usr/local/jdk1.8 #将文件夹移动到系统目录下
```

(2) 设置环境变量

- 修改全局配置文件，路径：`/etc/profile`

```
1 sudo vim /etc/profile #打开配置文件
```

文件打开后点击“i”进入编辑模式，添加以下配置信息，输入完成后，点击“Esc”退出编辑模式，并输入“:wq!”保存并退出：

```
1 export JAVA_HOME=/usr/local/jdk1.8
2 export JRE_HOME=${JAVA_HOME}/jre
3 export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
4 export PATH=.:${JAVA_HOME}/bin:$PATH
```

- 使配置的环境变量生效

```
1 source /etc/profile
```

(3) 检查是否安装成功

```
1 java -version
```

若出现如图1.1所示的信息，则表示安装成功。

```
dase-local@ecnu01:~$ java -version
java version "1.8.0_261"
Java(TM) SE Runtime Environment (build 1.8.0_261-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.261-b12, mixed mode)
```

图 1.1 Java 安装信息

1.5 思考题

1 hosts 文件的作用是什么？

2 实现免密钥登录有什么作用？

实验二 编程环境准备

2.1 实验目的

- 学会 IDEA 的安装与插件配置
- 掌握使用 IDEA 编写基于 Maven 项目的 Java/Scala 程序
- 学会使用 IDEA 将程序打成 jar 包

2.2 实验任务

- 完成 IDEA 的安装和激活，并配置 Maven 插件
- 完成 Java 环境的安装，并在 IDEA 中完成对 Java 的配置
- 完成 Scala 环境的安装，并在 IDEA 中完成对 Scala 的配置
- 完成将创建的 Maven 项目打成一个 jar 包

2.3 实验环境

- 操作系统：Ubuntu 18.04

2.4 实验步骤

2.4.1 IDEA 的安装与激活

- (1) 下载并解压 IntelliJ IDEA 的 Ultimate 版本

```
1 cd ~  
2 wget https://download.jetbrains.com/idea/ideaIU-2020.2.3.tar.gz #下载安装包  
3 tar -xvf ideaIU-2020.2.3.tar.gz #解压下载好的安装包
```

- (2) 具体安装与激活过程

- 进入 IDEA 解压后的文件夹，并运行 IDEA 安装指导程序

```
1 cd ~/idea-IU-202.7660.26/  
2 ./bin/idea.sh #运行IDEA安装指导程序
```

- 勾选 “I confirm that I have read and accept the terms of this User Agreement”，并点击 “Continue”
- 选择是否 “DATA SHAREING”，点击 “Don't Send”

- 在 IDEA 官网 (<https://account.jetbrains.com/login>) 利用学校邮箱注册一个账户，并登录学校邮箱操作，获得 License ID
- 利用注册账号的用户名或者邮箱激活 IDEA，如图2.1所示，激活后的界面如图2.2 所示

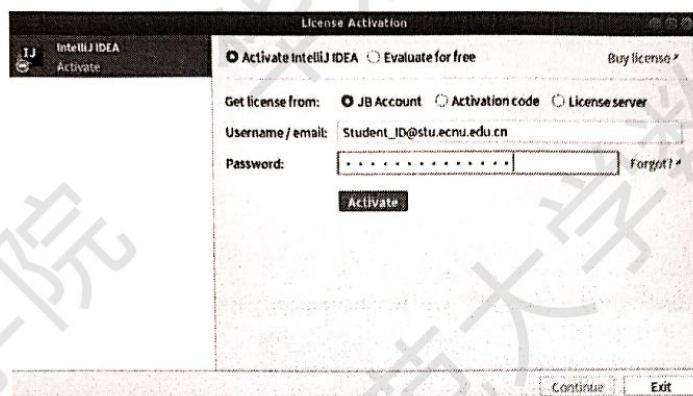


图 2.1 IDEA 激活界面



图 2.2 IDEA 开始界面

2.4.2 IDEA 中 Java 环境的安装与配置

- (1) 打开 IDEA，如若进入工程界面，依次选择 File->Close Project，返回 IDEA 主界面
- (2) 在 IDEA 主界面中选择 “Configure” -> “Structure for New Projects”
- (3) 在弹出页面中选中左侧 “Platform settings” 中的 “SDKs”，此时点击中栏中的“+”号，选择 “Download JDK...”，如图2.3所示
- (4) 在弹出页面中选中如图2.4所示的 JDK 来源与版本，点击 Download

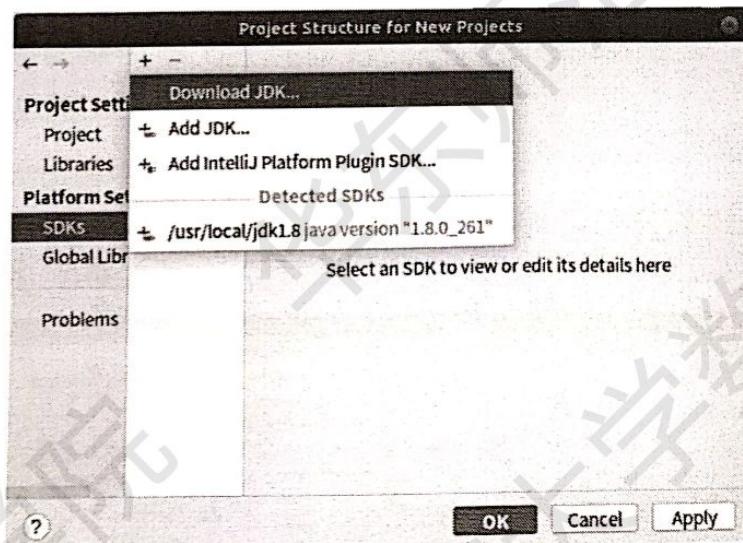


图 2.3 选择下载 JDK

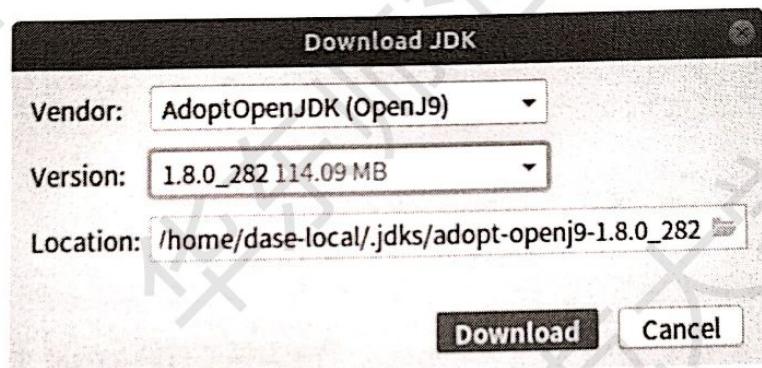


图 2.4 选择 JDK 来源与版本

(5) 下载完成后，在如图2.5所示的界面中点击 OK

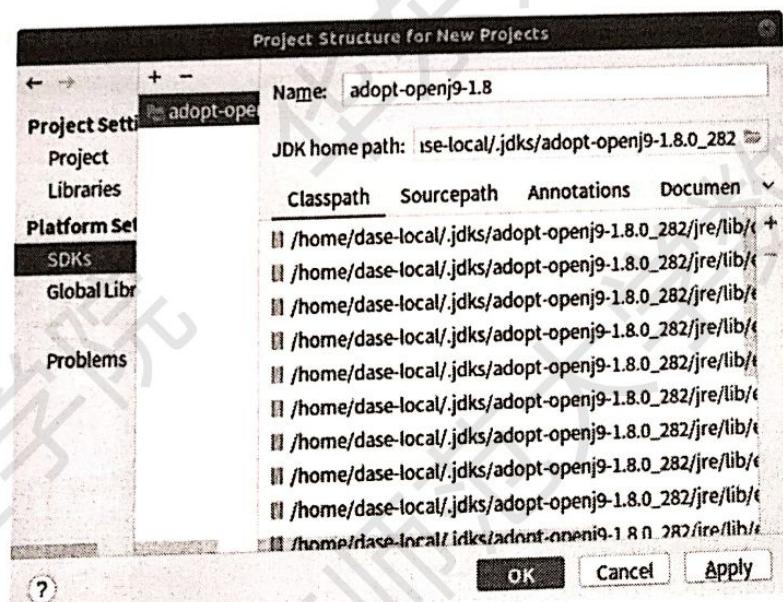


图 2.5 使用下载完成的 JDK

2.4.3 基于 Maven 项目编写 Java 程序

具体创建过程如下：

(1) 更新 maven 插件

IDEA 中集成了 maven 插件。在 IDEA 开始界面点击 “Configure” -> “Setting” 进入配置界面，如图2.6 所示。

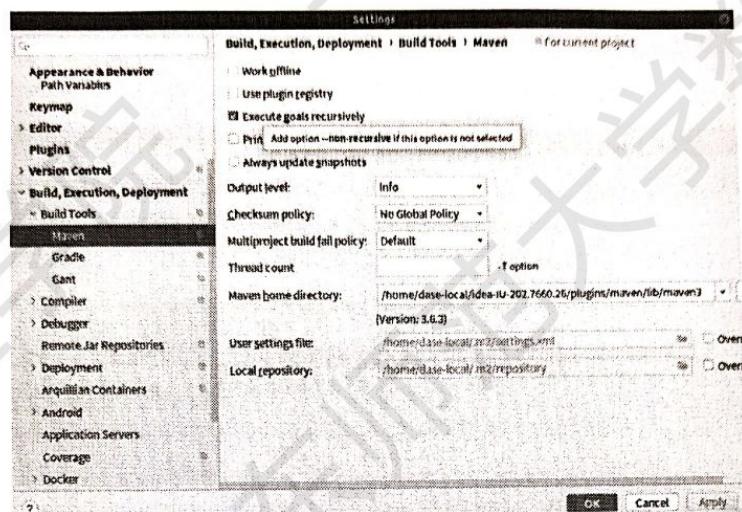


图 2.6 Maven 插件配置界面

(2) 新建一个 Maven 项目并调试运行

- 在 IDEA 开始界面点击 “+ NEW Project”
 - 选择相应配置

在弹出来的界面中，选择“Maven”、已安装的 JDK，并点击“next”，如图2.7所示。



图 2.7 新建 Maven 项目

- 输入项目信息

在弹出来的界面中，填写“Name”（项目名）、“Location”（项目存储位置）、GroupId（项目组织唯一的标识符，实际对应 Java 的包的结构，是 main 目录里 java 的目录结构名称）、ArtifactId（项目的唯一标识符，实际对应项目的名称）和 Version（项目的当前版本），点击“Finish”。如图 2.8 所示，新建了一个名为“HelloWorld”的项目。

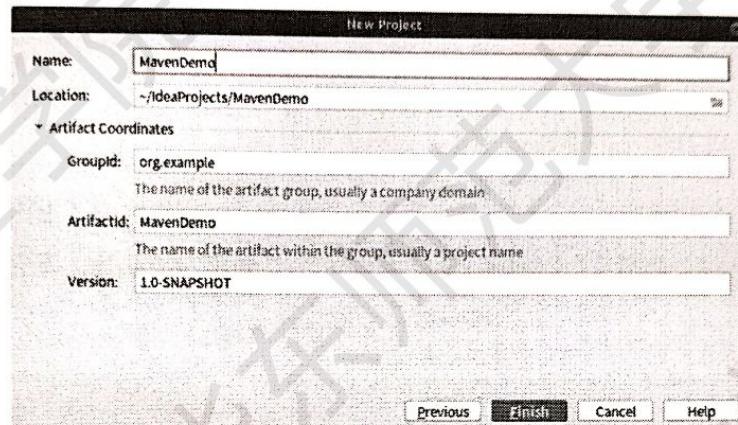


图 2.8 Maven 项目信息

- 新建 Java 类

在项目的 /src/main/java 路径下，新建一个名为“HelloWorld”的类。

```

1 public class HelloWorld {
2     public static void main(String[] args) {
3         System.out.println(args[0]);
4     }
5 }
```

- 配置运行环境

- 在菜单界面选择“Run”->“Edit Configurations”
- 在弹出的窗口中左侧点击“+”，选择“Application”
- 在新建的页面中，填写“Name”（应用名称）、“Mainclass”（主类的名称）、“Program arguments”（运行参数，即 args[x] 的值，多个值之间用空格隔开），如图 2.9 所示
- 点击“Apply”后点击“OK”

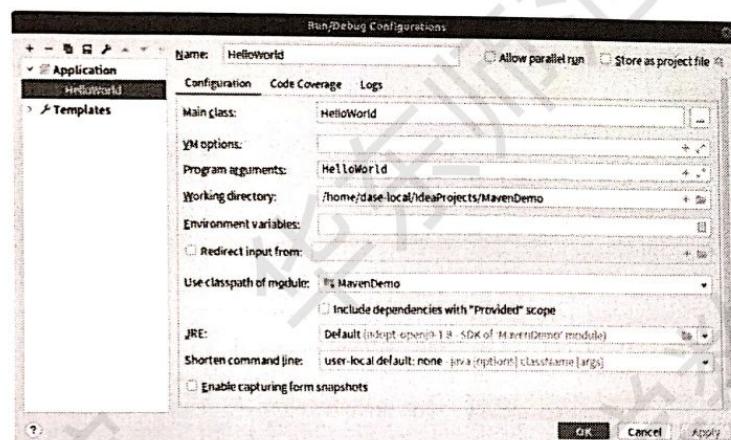


图 2.9 运行参数配置

- 在 IDEA 中直接运行

在菜单界面选择“Run”->“Run ‘HelloWorld’”，控制台输出如图 2.10 所示。

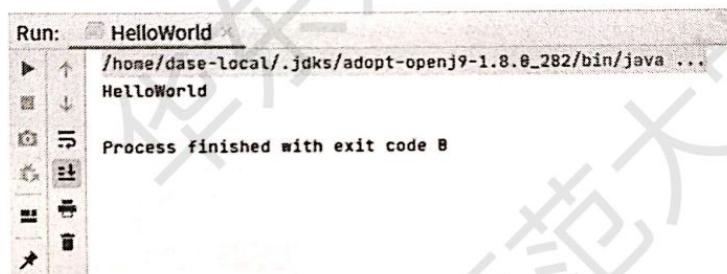


图 2.10 IDEA 控制台输出结果

2.4.4 IDEA 中 Scala 环境的安装与配置

具体安装与配置过程如下：

(1) 为 IDEA 安装 Scala 插件

在菜单中选择“File”->“Settings”，在弹出窗口的左侧边栏选择“Plugins”，在搜索栏输入“Scala”，如图 2.11 所示。选择“Install”进行安装，随后选择“RestartIDE”重启 IDEA。

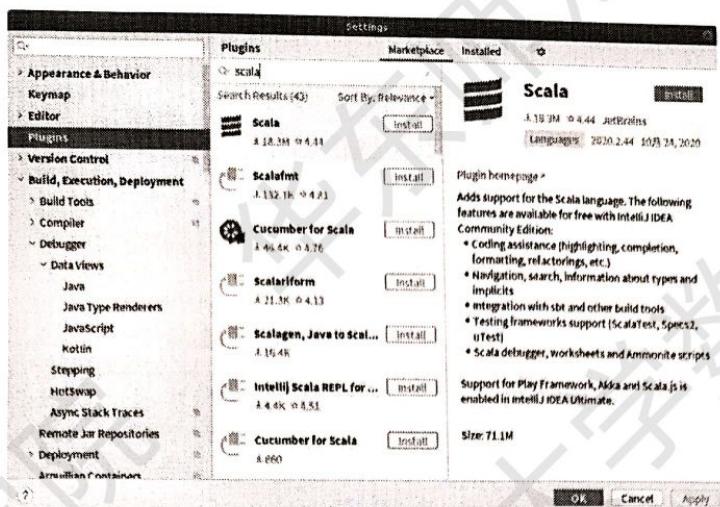


图 2.11 搜索 Scala 插件

(2) 安装 Scala SDK

- 返回 IDEA 主界面，依次选择“Configure”->“Structure for New Projects”，点击中栏的“+”号，选择“Scala SDK”选项，如图2.12所示

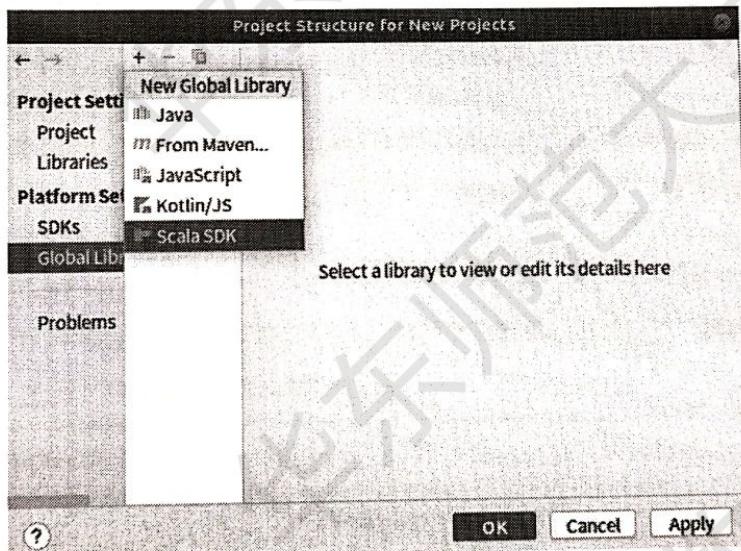


图 2.12 选择为新建项目配置 Scala SDK

- 点击 Scala SDK 选项后，在弹出的界面中选择 Download，进入如图2.13所示界面，选择 OK

2.4.5 基于 Maven 项目编写 Scala 程序

(1) 新建 Maven 项目

按照2.4.3节内容，新建一个 Maven 项目。

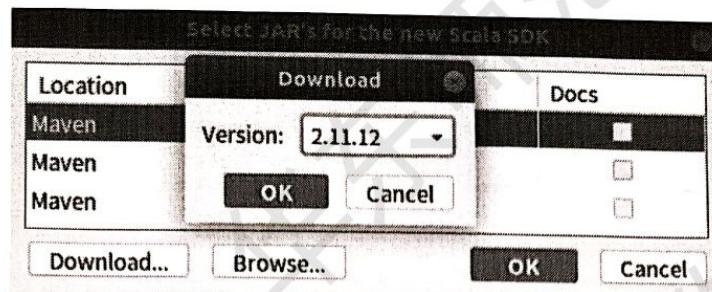


图 2.13 选择 Scala SDK 对应版本

(2) 创建 Scala 代码路径

展开项目目录，右键点击 `src/main`/目录，依次选择“New”->“Directory”，输入文件夹名称“scala”。接着右键点击 `scala` 目录，依次选择 `Mark Directory as -> Sources Root`。`scala` 文件夹颜色会变为蓝色。

(3) 导入 Scala 相关配置

依次选择“File”->“Project Structure...”，在弹出的页面中选择“Platform Settings”下的“Global Libraries”。右键选中中栏中的 `scala-sdk-2.11.12`，选择 `Add to Modules...`，接着选择“OK”。

(4) 新建 object “Hello World”

右键单击 `src/main/scala` 目录，依次选择“New->Scala Class”，输入文件名称“`HelloWorld`”，选择“Object”。输入如下代码：

```

1 object HelloWorld {
2     def main(array:Array[String]): Unit ={
3         println("HelloWorld");
4     }
5 }
```

(5) 配置运行环境

- 在菜单界面选择“Run”->“Edit Configurations”
- 在弹出的窗口中左侧点击“+”，选择“Application”
- 在新建的页面中，填写“Name”(应用名称)、“Main class”(主类的名称)、“Program arguments”(运行参数，即 `args[x]` 的值，多个值之间用空格隔开)，如图2.14 所示
- 点击“Apply”后点击“OK”

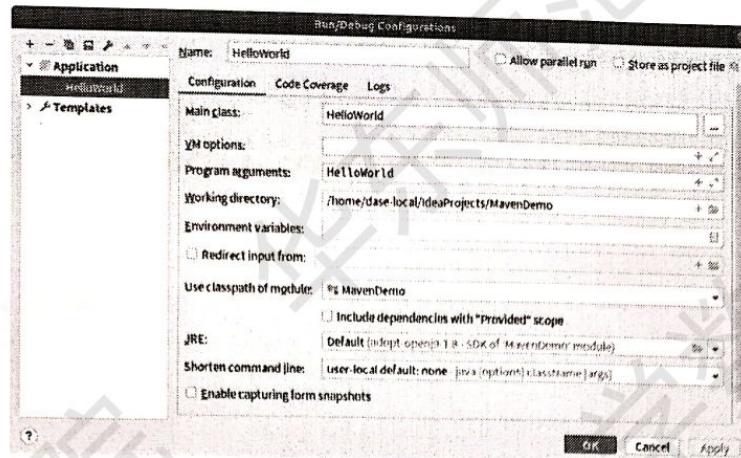


图 2.14 运行参数配置

(6) 在 IDEA 中直接运行

在菜单界面选择“Run”->“Run ‘HelloWorld’”，控制台输出如图 2.15 所示。

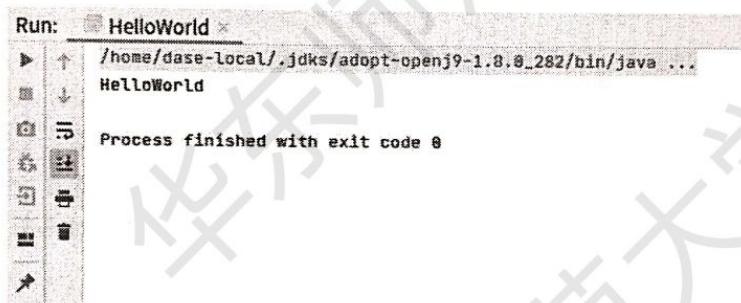


图 2.15 IDEA 控制台输出结果

2.4.6 使用 IDEA 将程序打包为 jar 包

以上述基于 Maven 项目编写的 Java 程序为例，具体打包过程如下：

- (1) 选中 Java 项目工程名称，在菜单中选择“File”->“Project Structure”
- (2) 在弹出的窗口中左侧选中“Artifacts”，点击“+”按钮，选择“jar”->“Empty”
- (3) 配置 jar 包参数

在打开的界面中配置相关参数，如图 2.16 所示，配置了名为“HelloWorld”的 jar 包，存放在“/home/dase-local/IdeaProjects/HelloWorld/out/artifacts/HelloWorld”目录下。配置完成后，点击“Apply”后点击“OK”。

- 填写“Name”（生成 jar 包的名称）
- 点击“Output Layout”下的“+”按钮，选择“Module Output”，然后选择需要打包的“Module”，即将编译生成的 class 文件打包到 jar 包中
- 选中 jar 包，点击“Create Manifest File”，选择项目根目录
- 选中 jar 包，填写“Main Class”（主类）

- 勾选“Include in project build”，即 IDEA 每次运行项目时，都会自动执行打包流程，更新 jar 包
- 点击“Apply”后点击“OK”

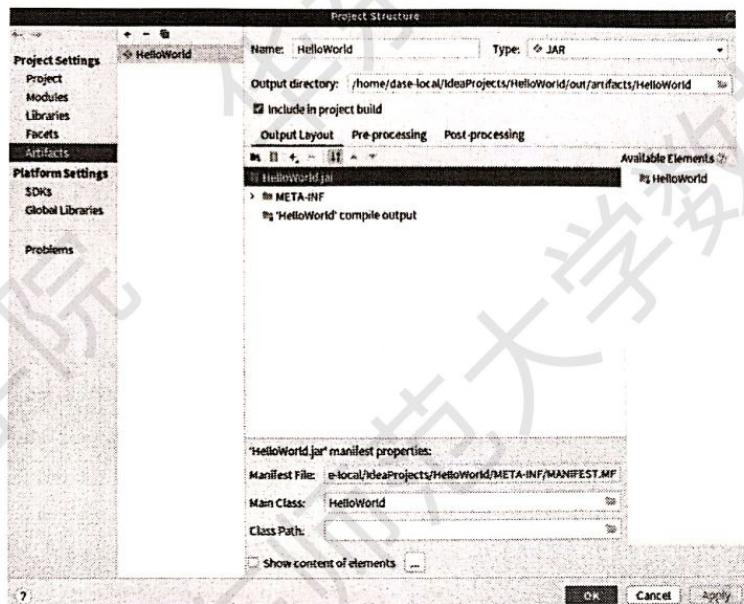


图 2.16 jar 包配置界面

(4) 在主界面构建项目以生成 jar 包

- 方法一：重新运行 Java 程序，项目自动打包生成 jar 包
- 方法二：在菜单中选择“Build”->“Build Artifacts”，然后选择“Build”或者“Rebuild”，生成 jar 包

2.5 思考题

- 图2.9中的 VM option 是什么？
- JVM 是什么？它是操作系统中的进程吗？
- 同一个工程中的代码是否可以同时使用 Java 和 Scala 两种语言编写？