Project 1 第一个 Java 程序

概述

这个 Project 中你将动手搭建 Java 的编译和运行环境. 同时将编写第一个 Java 程序. 在实验内容中, 你会和这个程序做一些互动. 将看到第一个 Java 编译错误, 第一个 Java 运行时错误以及第一个Bug. 对这些 Java 程序和实验结果你一定会有很多疑惑和问题. 可以尝试自己查资料解决, 也可以把它们记录下来. 在今后的课程中, 我们会逐步找到这些问题的答案. 无论如何, 现在最值得庆祝的是你已经能用 Java 输出那句振奋人心的话: Hello world!

安装 Java 环境

我们将使用命令 `javac` 来编译 Java 程序, 用命令 `java` 运行编译好的 Java 程序. 这两个命令包含在 JDK (Java Development Kit) 中. JDK 有不同的实现, 其中比较常见的是 Oracle JDK 和 OpenJDK.

Winodws + Oracle JDK

下载及安装

1. 进入 Oracle JDK 下载页面
2. 下载 Java SE Development Kit. 目前有8u74和8u73两版本, 可选择其中任意一个.
3. 选择 Windows 对应的文件下载. 注意其中又分为 x86 和 x64 两种. 若你的机器是32位则选择 x86, 若是64位则选择 x64. 下载前需勾选 Accept License Agreement.
4. 打开所下载的 JDK 安装文件, 根据指示设置好安装路径(任意路径, 之后的环境变量配置需用到)进行安装.

设置环境变量 (Windows 7)

安装 JDK 之后可能仍然不能编译 Java 程序. 需要设置的环境变量. 环境变量可以简单的理解为一种"全局变量". 运行在 Windows 下的任何程序可以通过 Windows 提供的函数访问这些变量. 在这里需要通过设置三个环境变量: `JAVA_HOME, classpath, path` 来完成 Java 环境的搭建.

1. 右击"我的电脑", 点击"属性", 选择"高级系统变量", 打开系统属性页面.
2. 点击右下方的"环境变量"选项.

3. 在"系统变量"下, 新建变量名为 `JAVA_HOME` 的变量. 变量值为 jdk 安装时指定的安装目录, 比如 `D:\Java\jdk1.8.0_20` .
4. 仍在"系统变量"下, 新建名为 `classpath` 的变量, 变量值为

```
.;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar
```

   注意: 第一个分号前有一个点号"**.**". Java 在编译和运行时往往依赖一些特定的"库"文件(类似于 C 语言中的 `stdio.h` 等). 这些库文件被放在磁盘上的特定位置 (这里就是JDK在安装时指定的路径). 环境变量 `classpath` 的作用在于告诉 Java 这些库文件所在的位置. 在设置了这个变量之后, 编译和运行 Java 程序就可以通过访问 `classpath` 找到相应的"库". 以上的变量设置具体解释如下:

   - 指定了三个可能存放"库"函数的目录: `.` , `%JAVA_HOME%\lib` 以及 `%JAVA_HOME%\lib\tools.jar` . 它们用分号";"隔开.
   - 表达式 `%JAVA_HOME%` 表示取环境变量 `JAVA_HOME` 的值. 根据之前的设置, 它的值为 `D:\Java\jdk1.8.0_20` . 因此, `%JAVA_HOME%\lib` 实际就是 `D:\Java\jdk1.8.0_20\lib` .
     对 `%JAVA_HOME%\lib\tools.jar` 同理.
   - 点号 `.` 有一个固定含义: 表示程序运行的目录.
   - 这个环境变量的含义是"在每一个Java程序编译和运行时, 到 `D:\Java\jdk1.8.0_20\lib` , `D:\Java\jdk1.8.0_20\lib\tools.jar` , 以及程序所在的目录中查找相应的库函数".
5. 仍在系统变量下, 找到 `path` 变量进行编辑. 在变量值末尾添加

```
;%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin
```

   `path` 变量是为了帮助系统能够找到已经安装好的 `javac` 和 `java` 命令. 与 `classpath` 变量类似, 系统在运行一个命令时, 需要知道这个命令的二进制文件所在的位置. 通过在 `path` 后添加这两个目录, 当系统直接运行 `javac` 或 `java` 时就能在指定位置找到这两个命令.

6. 完成上述配置后, 可检测是否配置成功: 在开始菜单中打开 `cmd` , 进入命令行, 分别输入 `java` 和 `javac` 进行测试.

Linux + OpenJDK

在 Linux 上安装 JDK 同样比较容易. 这里将以 Ubuntu 为例, 其他发行版本请查阅相关文档. 在终端下执行 `javac` 与 `java` , 确认是否已经安装 JDK. 若否, 则执行如下两条命令进行安装:

```
sudo apt-get update
sudo apt-get install openjdk-8-jdk
```

第一个程序 HelloWorld.java

完成你的第一个 Java 程序需要三个步骤: 编写源代码, 编译, 运行. 下面以 Windows 为例.

编写源码

1. 打开一个文本编辑器 (例如记事本), 写入下面的程序

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

2. 将其保存为HelloWorld.java. 假设保存路径为"D:\HelloWorld\"

编译

1. 打开 `cmd` , 进入 HelloWorld.java 所在的目录

```
D:
cd D:\HelloWorld\
```

2. 输入以下命令进行编译

```
javac HelloWorld.java
```

3. 若编译错误, 将会输出**编译错误信息**. 若编译成功, 则没有信息输出, 同时生成一个 HelloWorld.class 文件 (.class 文件为编译好的 Java ByteCode, 执行时由 Java 虚拟机执行其中代码).

运行

1. 打开 `cmd` , 进入 HelloWorld.java 所在的目录

```
D:
cd D:\HelloWorld\
```

2. 输入以下命令运行程序

```
java HelloWorld
```

3. 若运行正确, 将在屏幕输出 `Hello World!` . 否则将输出运行错误信息.

说明

- Linux 步骤. 基本与上述步骤相同.
  1. 编写源码: 可以使用任意文本编辑器(如gedit). 假设文件 HelloWorld.java 保存在目录 `~/HelloWorld/` .
  2. 编译: 打开终端, 执行以下两条命令

     ```
     cd ~/HelloWorld
     javac HelloWorld.java
     ```

  3. 运行: 打开终端, 执行以下两条命令

     ```
     cd ~/HelloWorld
     java HelloWorld
     ```

- 文本编辑器. 程序源代码在编译之前和普通文本文件没有区别(比如一段诗歌, 一篇小说), 都可以看成字符串. 因此可以选用任何你喜欢的文本编辑器来编写程序 (我们鼓励使用文本编辑器来完成所有的实验项目). 但有一些编辑器更适合编写源代码. 以下我们推荐一些编辑器.
  - Windows: Sublime Text, Notepad++.
  - Linux: Sublime Text, Vim, Emacs. (Ubuntu 可以用apt install 安装).
- 集成开发环境 (Integrated Development Environment, IDE). 虽然文本编辑器可以胜任这门课程所有的实验, 对于一些规模较大的项目, 使用功能更为强大的集成开发环境能够帮助你事半功倍. 开源社区中有许多优秀的 Java 集成开发环境, 我们推荐下面两个:
  - Eclipse
  - Intellij
- 编译工具 (Java build tools). 我们使用JDK javac 作为主要的编译工具. 同样, 对于大规模的项目, 有其他更强大的编译工具. 以下为两种常用的编译工具:
  - Apache Ant
  - Apache Maven

实验内容

1. 创建 HelloWorld.java, 编译并执行.
2. 分别删除第一行的 public, 第二行的 public, static, void, args, String 分别会发生什么?
3. 如果错误的拼写了 public, static, void, args, String 分别会发生什么?
4. 如果错误的拼写了 System, out, println 分别会发生什么?
5. 将第二行替换成为 `public static void main()` 会发生什么?

6. 将文件重命名为 HalloWorld.java 会发生什么?
7. 将 HelloWorld.java 中的每个空格替换为两个空格, 每行间增加一些空行, 删除每行前的空格分别会发生什么?
8. 编写程序 Hi.java

```
public class Hi {
    public static void main(String[] args) {
        System.out.print("Hi, ");
        System.out.print(args[0]);
        System.out.println(". How are you?");
    }
}
```

执行以下命令分别会发生什么? 你能理解 args 的含义吗?

```
java Hi
java Hi @#$%
java Hi 1024
java Hi Bob
java Hi.java Bob
java Hi.class Bob
java Hi Alice Bob
```

修改以上程序, 使它能够接收3个命令行参数, 并倒序输出它们. 比如执行 `java Hi Alice Bob Carol`, 输出 `Hi Carol, Bob, Alice.`.

9. 编写程序输出10次 "Hello World!".
10. Java 语言的变量类型 (int, float, double), 表达式(运算符, 逻辑表达式等) 以及控制结构 (条件语句, 循环语句等) 和 C 语言比较相似. 你能否用使用 C 语言的知识写一些稍微复杂的 Java 程序呢?