**Jdk的下载和安装**

官网下载，双击安装

控制台输入java –version 来确定安装成功

如第二步不成功，则需要配置一下环境变量

JAVA\_HOME 配置为JDK安装根目录，后续可以将%JAVA\_HOME%作为变量使用

CLASSPATH %JAVA\_HOME%\lib —

Path %JAVA\_HOME%\bin; 方便使用java命令&工具

**intellij的下载 安装和激活**

<https://www.jetbrains.com/idea/download/> 上下载 idea （当前下载的为2017.3.5）

激活

1[下载破解补丁](http://idea.lanyus.com/jar/JetbrainsCrack-2.7-release-str.jar)

**2.补丁copy到 intellij的安装目录**

**3.** 打开并编辑 idea.exe.vmoptions和idea64.exe.vmoptions这两个文件，在上述**两个文件的最后一行**均加上下面这段代码   
-javaagent:G:\idea\IntelliJ IDEA 2017.3.4\bin\JetbrainsCrack-2.7-release-str.jar(路径修改为对应的安装目录)

**4.** 启动idea，选择Activation Code，并输入如下激活码

BIG3CLIK6F--iygsIMXTVeSyYkUxAqpHmymrgwN5InkOfeRhhPIPa88FO9FRuZosIBTY18tflChACznk3qferT7iMGKm7pumDTR4FbVVlK/3n1ER0eMKu2NcaXb7m10xT6kLW1Xb3LtuZEnuis5pYuEwT1zR7GskeNWdYZ0dAJpNDLFrqPyAPo5s1KLDHKpw+VfVd4uf7RMjOIzuJhAAYAG+amyivQt61I9aYiwpHQvUphvTwi0X0qL/oDJHAQbIv4Qwscyo4aYZJBKutYioZH9rgOP6Yw/sCltpoPWlJtDOcw/iEWYiCVG1pH9AWjCYXZ9AbbEBOWV71IQr5VWrsqFZ7cg7hLEJ3A==-

参考 <https://blog.csdn.net/qq_27686779/article/details/78870816>

**建立控制台程序helloworld**

建立helloworld项目

启动Intellij idea，选择Create New Project，选择对应的jdk版本，点next

勾选 Create project from template ,选择Command Line App,点Next

填写项目名称，路径等信息

在main函数中输出控制的代码

参考 https://blog.csdn.net/qq\_27093465/article/details/52795186

**将控制台程序输出为jar包**

选中java项目，选择 File -> Project Structure

在弹出的窗口中左侧选中"Artifacts"，点击"+"选择jar，然后选择"from modules with dependencies"

选择MainClass 中 指定当前项目的Main方法，确定ok,

当需要输出jar包时，选择“Build - Build Artifacts”下的“Build”或者“Rebuild”即可生成最终的可运行的jar

**字体控制**  File -> Setting ->Editor ->Font 设置Size

**智能提示忽略大小写** File -> Setting -> Editor ->General->Code Completion

设置 Case sensitive completion 为 None

**安装和配置Maven**

在官网上下载maven，解压到任意目录（例如：C:\Apache\Maven\Maven-3.5.0）

**镜像配置**

Maven的官方仓库服务器是在国外  
访问很不稳定。需要修改配置文件Maven根目录\conf\settings.xml,或者复制settings.xml文件改名作为自定义的配置文件，以适应不同场景的需要，这里就复制一份并更名为：settings-my.xml  
在配置文件settings-my.xml中mirrors节点增加配置：

<**mirror**>

<**id**>alimaven</**id**>

<**name**>aliyun maven</**name**>

<**url**>http://maven.aliyun.com/nexus/content/groups/public/</**url**>

<**mirrorOf**>central</**mirrorOf**>

</**mirror**>

创建本地包仓库文件夹，例如（C:\Apache\Maven\Repository）

在intellij上配置maven

File-Settings-Build.Exectution.Deployment-> Maven配置如下三项：

Maven Home Directory：C:\Apache\Maven\Maven-3.5.0  
User settings file：C:\Apache\Maven\Maven-3.5.0\conf\settings-my.xml  
Local repository：C:\Apache\Maven\Repository

参考： <https://ken.io/note/java-quickstart-sde>

**新建Maven项目**

File->New->Project

选择Maven，然后选择Maven项目模板maven-archetype-quickstart

填写 GroupId Artifactid

选择setting file 和 repository ，配置当上一步配置的目录

等待intellij 下载maven的各种jar包（大概20分钟左右）

**Maven项目引用第三方的jar包**

在 pom.xml文件中添加 如下 依赖

<**dependency**>  
 <**groupId**>com.fasterxml.jackson.core</**groupId**>  
 <**artifactId**>jackson-databind</**artifactId**>  
 <**version**>2.7.0</**version**>  
</**dependency**>

**引用本地的jar包**

File->Project Structure->Modules->Dependencies-> + 号->添加 JARs or directories->选择 Jar 包->Apply

在代码中import 对应的package后，直接使用对应的类

**引用当前project中的module**

File->Project Structure->Modules->Dependencies-> + 号->添加 Module Dependency->选择Module->Apply

在代码中import 对应的package后，直接使用对应的类

**普通项目引用 maven**

File->Project Structure->Modules->Dependencies-> + 号->Library->From Maven(关键字搜索)->Apply

**Log4j的使用**

引入jar包

在src 目录下新建log4j.properties 文件，在文件中复制以下内容

log4j.rootLogger=debug,myfile

log4j.appender.myfile=org.apache.log4j.FileAppender  
log4j.appender.myfile.File=log\\log4j.txt  
log4j.appender.myfile.layout=org.apache.log4j.PatternLayout  
log4j.appender.myfile.layout.ConversionPattern=%d{yyyy-MM-dd HH\:mm\:ss} [%t] [%c] [%p] [%L]- %m%n

此配置会将日志输出到 文本文件

新建TraceHelper类，输出以下内容，

**import** org.apache.log4j.Logger;  
  
**public class** TraceHelper {  
  
 **private static** Logger *logger* = Logger.*getLogger*(TraceHelper.**class**);  
  
 **public static void** TraceInfo(String I\_Info)  
 {  
 *logger*.debug(I\_Info);  
 }  
}

TraceInfo 方法为输出日志的方法

**Jackson的使用**

Intellij中scr目录下的文件默认会打包到jar包中

**控制台的输入输出**

输出：System.***out***.println(**"请选择任务种类："**);

输入：Scanner scan = **new** Scanner(System.***in***); String readLine = scan.nextLine();

**基本类型的数据转换**

string -> int Integer.*parseInt*(str)

int -> string String.valueOf(i)

char[] -> string char ch[]=new char[]{'5','a','B'}; String str=String.valueOf(ch);

**String -> float** Float.parseFloat(str)

float-> string

**String -> Date** java.text.SimpleDateFormat formatter =new SimpleDateFormat("yyyy-MM-dd");  
String s= "2011-07-09"; Date date =  formatter.parse(s);

**Date->String** java.text.SimpleDateFormat formatter =new SimpleDateFormat("yyyy-MM-dd");  
String date = formatter.format(new Date());

**String-> BigDecimal** String StrBd="1048576.1024"; BigDecimal bd=new BigDecimal(StrBd);

**BigDecimal->String** bd=bd.setScale(2, BigDecimal.ROUND\_HALF\_UP);

String OutString=bd.toString();

**Long->int** long lll = 300000;   int ii = (int)lll; 或 int iii= new Long(ll).intValue();

int -> long long lll =(long)i; long l=new Long((long)i);

string -> Boolean Boolean.parseBoolean(“true”) Boolean.parseBoolean(“false”)

**集合类**

1）如果要求线程安全，使用Vector，Hashtable

2）如果不要求线程安全，使用ArrayList，LinkedList，HashMap

3)如果要求键值对，则使用HashMap，Hashtable

4）如果数据量很大，又要求线程安全考虑Vector

1．ArrayList: 元素单个，效率高，多用于查询

2．Vector: 元素单个，线程安全，多用于查询

3．LinkedList:元素单个，多用于插入和删除

4．HashMap: 元素成对，元素可为空

5．HashTable: 元素成对，线程安全，元素不可为空

**参考：**[**https://www.cnblogs.com/LittleHann/p/3690187.html**](https://www.cnblogs.com/LittleHann/p/3690187.html)

**https://www.cnblogs.com/ZhuRenWang/p/4773647.html**

**小技巧**

**得到 jar包的当前路径**

**public static** String GetLocalPath()  
{  
 String R\_Result = **null**;  
  
 String jarWholePath = Main.**class**.getProtectionDomain().getCodeSource().getLocation().getFile();  
 **try** {  
 jarWholePath = java.net.URLDecoder.*decode*(jarWholePath, **"UTF-8"**);  
 } **catch** (UnsupportedEncodingException e) { System.***out***.println(e.toString()); }  
 R\_Result = **new** File(jarWholePath).getParentFile().getAbsolutePath();  
  
 **return** R\_Result;  
}

**得到换行字符串** System.*getProperty*(**"line.separator"**) 来

**得到 当前日期的字符串** **new** SimpleDateFormat(**"yyyy-MM-dd HH:mm:ss"**).format(**new** Date())

**文件的读写**

读取和写入文本文件 BufferedReader BufferedWriter 针对字符流

读取和写入二进制文件 BufferedInputStream BufferedOutputStream 针对字节流

2. 3. java 访问数据库

4. 和c#对比，总结语言的接口（单独一个文档）