Cmd open 计算器

Win r 输入calc

Cmd open 画图

Win r 输入mspaint

Optional<T> optional=Optional.empty();创建空的对象

Optional<T>optional=Optional.of(xx);创建一个option对象，如果是null抛出nullException

Optional<T>optional=Optional.ornull(xx);创建一个optional对象，对象可以是null

Optional<T>optional=Optional.ornull(xx).orElse(xx);创建一个optional对象，对象如果是null返回入参类型的返回值

Optional.get()返回optional对象的提示信息

Optional.isPresent()返回boolean值，null为false。

<\\10.254.161.197\Project Files\zhe.cui>>

Feign与Ribbon通常用于处理服务之间的访问，实现服务通讯之间的负载均衡。

Feign对ribbon与restTemplate进行了封装简化，所以我们在使用过程中可以直接使用feign规则来访问远程服务，实现服务通讯之间的负载均衡。

服务与服务之间连接超时

传统模式 1人肉运维（重启服务器） 2添加延时机制

Activiti5.0是对jbpmn4.0一种延续

Activiti

Act\_ge: 通用表

保存流程引擎，流程图

Act\_re:部署表

Act\_hi:历史纪录表

Act\_id:用户信息表

Act\_ru:运行时表

追求跟已有的东西是两个性质的东西

对称加密 A->A

非对称加密 A->B

对称加密算法的优点是算法公开、计算量小、加密速度快、加密效率高。

对称加密算法的缺点是在数据传送前，发送方和接收方必须商定好秘钥，然后使双方都能保存好秘钥。其次如果一方的秘钥被泄露，那么加密信息也就不安全了。另外，每对用户每次使用对称加密算法时，都需要使用其他人不知道的独一秘钥，这会使得收、发双方所拥有的钥匙数量巨大，密钥管理成为双方的负担。

Maven

Maven\confi\Setting中配置

可以配置多个mirror在mirrors中，按顺序执行下载，EG 第一个仓库没有该jar包则去下一个jar包下载，知道下载成功

或者直接在eclipse pom文件中配置 repository这个标签，会在当前maven项目中采取repostiory中的仓库地址进行下载。

云计算

软，硬

Java结构，回忆

JDK 全部Java需要的jar包，包含JRE JVM

JRE包含JVM，java运行时API

JVM代码执行器，只能读取class文件进行编码

JVM（虚拟机）

目前的jvm，都是采用hotspot vm虚拟机。用来替代JIT（Just in time，即时编译），提高运行。

Java起初将原代码编译为class在虚拟机运行，速度慢。

Hotspot将常用的部分代码编译为native代码，速度快。

Hotsopt属于动态编译。

Jvm在每次运行的时候，会读取JRE所有代码编译成本地代码（字节码）。然后你编写的代码会被JIT即时编译成本地代码（编译器）。

计算机不能直接理解编译高级语言，只能执行机器语言，所以必须将高级语言翻译成机器语言在执行。

编译器

Java源代码编译成class字节码文件。

解释器

用来解释class字节码文件。

I解释型语言

Java是解释型语言，在运行时编译，再由专门的解释器执行。每执行一次，就会编译一次，效率低下。

优点是可以在各大平台执行。

缺点是需要解释环境（安装jvm），运行慢。

II编译型语言

编写的程序在运行之前，需要通过编译器将代码编译成机器语言，然后由操作系统执行。

编译和执行是分开的，但不能跨平台。

例如C编译成exe，在windows可以执行。在其他系统就需要重新编写了。

优点：运行快，同环境对环境要求低

确定：无法移植。（硬伤）

III动态编译

Hotspot VM对字节码的编译不是在程序运行之前，而是在程序运行过程中。

Hotspot VM有一个监视器（profile monitor）,用来监视运行状况。

Java字节码（class文件）用解释的方式加载到VM中。使用频率高，对程序性能影响重要的代码，称之为(Hotspot)。Hotspot VM会将这些热点动态的编译(JIT)成机器码(native code),同时对机器码进行优化，提高效率。使用频率低的代码，VM不会执行。

VI Hotspot VM对字节码处理方式

不编译：运行时编译

编译：字节码编译成本地代码，VM执行时无需再次编译。

编译并优化：字节码编译成本地代码并优化

V 为什么不采用静态编译

提前将字节码编译成本地代码，在VM中运行。

1. 动态编译可以优先优化热点代码
2. 方法调用最浪费资源，因为有很多进出栈操作。动态编译可以把方法调用改为非方法调用，直接将方法代码块嵌入到调用的地方，顺序执行。
3. 对象结构不明确。动态编译可以确定对象之间的继承实现关系，多态是在程序中才能具体确定的。