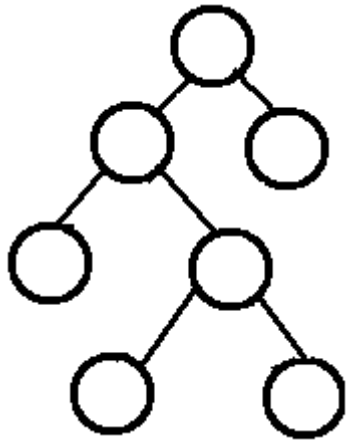
完满二叉树(Full Binary Tree or Strictly Binary Tree) 所有非叶子结点的度都是2。（只要你有孩子，你就必然是有两个孩子。）



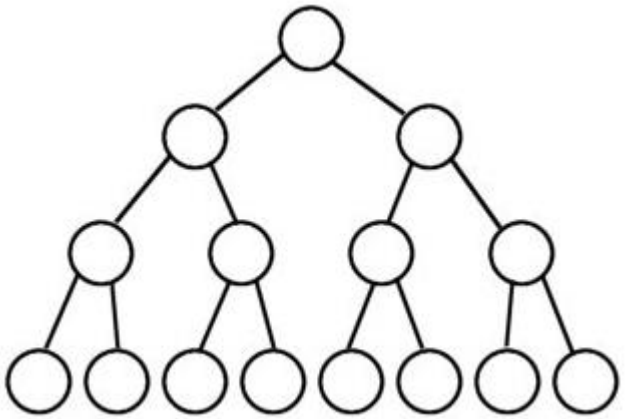
完美二叉树(Perfect Binary Tree)

一棵深度为k且有2k-1个结点的二叉树称为满二叉树。

满二叉树的特点：

（1） 每一层上的结点数都达到最大值。即对给定的高度，它是具有最多结点数的二叉树。

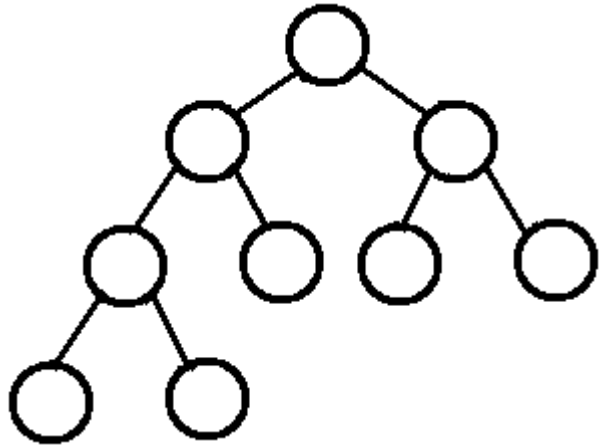
（2） 满二叉树中不存在度数为1的结点，每个分支结点均有两棵高度相同的子树，且树叶都在最下一层上。



如果只是删除满二叉树最底层最右边的连续若干个节点，这样形成的二叉树就是完全二叉树(Complete Binary Tree)

若设二叉树的深度为h，除第 h 层外，其它各层 (1～h-1) 的结点数都达到最大个数，第 h 层所有的结点都连续集中在最左边，这就是完全二叉树。

一棵二叉树至多只有最下面的两层上的结点的度数可以小于2，并且最下层上的结点都集中在该层最左边的若干位置上，则此二叉树成为完全二叉树。



完全二叉树从根结点到倒数第二层满足完美二叉树，

最后一层可以不完全填充，其叶子结点都靠左对齐。

二叉树的常见操作包括树的遍历，即以一种特定的规律访问树中的所有节点。常见的遍历方式包括：

前序遍历(Pre-order traversal)：访问根结点；按前序遍历左子树；按前序遍历右子树。

中序遍历(In-order traversal)：按中序遍历左子树；访问根结点；按中序遍历右子树。特别地，对于二分查找树而言，中序遍历可以获得一个由小到大或者由大到小的有序序列。

后续遍历(Post-order traversal)：按后序遍历左子树；按后序遍历右子树；访问根结点。

以上三种遍历方式都是深度优先搜索算法(depth-first search)。深度优先算法最自然的实现方式是通过递归实现。此外，另一个值得注意的要点是：深度优先的算法往往都可以通过使用栈数据结构将递归化为非递归实现。这里利用了栈先进后出的特性，其数据的进出顺序与递归顺序一致

层次遍历(Level traversal)：首的层；当第i层的所有结点访问完之后，再从左至右依次访问第i+1先访问第0层，也就是根结点所在层的各个结点。层次遍历属于广度优先搜索算法(breadth-first search)。广度优先算法往往通过队列数据结构实现。

Maven

在进行软件开发的过程中，无论什么项目，采用何种技术，使用何种编程语言，我们 都要重复相同的开发步骤：编码，测试，打包，发布，文档。实际上这些步骤是完全重复性的工作。那为什么让软件开发人员去重复这些工作？开发人员的主要任务 应该是关注商业逻辑并去实现它，而不是把时间浪费在学习如何在不同的环境中去打包，发布，。。。。

Maven正是为了将开发人员从这些任务中解脱出来而诞生的。Apache Maven 是一种用作软件项目管理和理解工具。它基于项目对象模型（POM）的概念， 可以管理一个项目的构建、报告以及从项目核心信息中生成文档。

Maven能够：

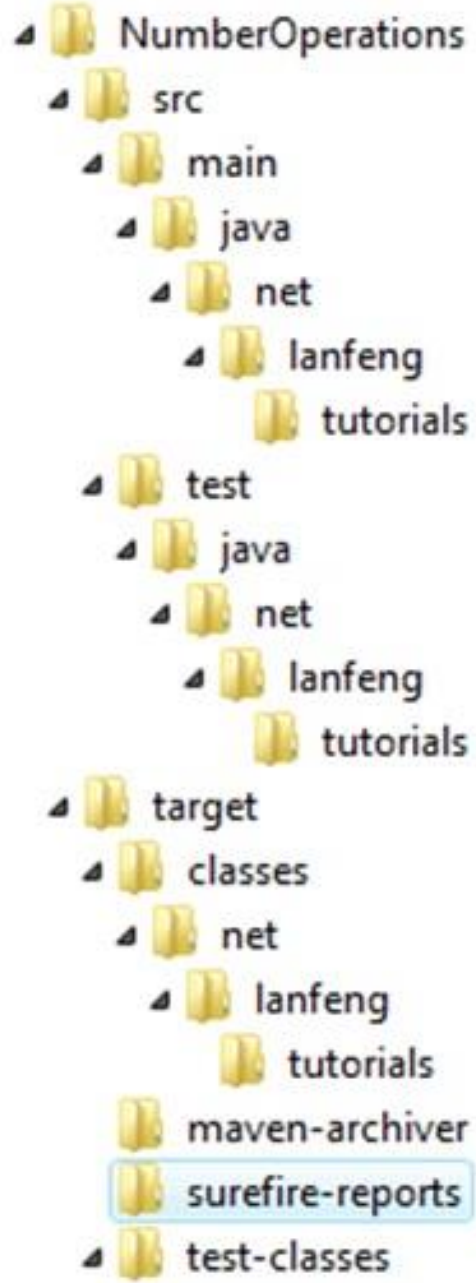
1）理解并管理整个软件开发周期，重用标准的构建过程，比如：编译，测试，打包等。同时Maven还可以通过相应的元数据，重用构建逻辑到一个项目。

2）Maven负责整个项目的构建过程。开发人员只需要描述项目基本信息在一个配置文件中：pom.xml。也就是说，Maven的使用者只需要回答“What”而不是“How”。

# Maven设计原则

1）Convention Over Configuration (约定优于配置)。在现实生活中，有很多常识性的东西，地球人都知道。比如说：如何过马路(红灯停绿灯行)，如何开门，关门等。对于这些事情，人们已经有了默认的约定。

在软件开发过程中，道理也是类似的，如果我们事先约定好所有项目的目录结构，标 准开发过程（编译，测试，。。。。），所有人都遵循这个约定。软件项目的管理就会变得简单很多。在现在流行的很多框架中，都使用了这个概念，比如EJB3和 Ruby on Rails。在Maven中默认的目录结构如下：


NumberOperations
src
main
java
net
lanfeng
tutorials
test
java
net
lanfeng
tutorials
target
classes
net
lanfeng
tutorials
maven-archiver
surefire-reports
test-classes

可以看出以下几个标准的Maven目录：

src：源代码目录。所有的源代码都被放在了这个目录下。在这个目录下又包括了：

1) main：所有的源代码放在这里。对于Java项目，还有一个下级子目录：java. 对于Flex项目则是flex，。。。

2) test：所有的单元测试类放在这里。

target：所有编译过的类文件以及生成的打包文件(.jar, .war, ...)放在这里。

2）Reuse Build Logic (重用构建逻辑)：Maven把构建逻辑封装到插件中来达到重用的目的。这样在Maven就有用于编译的插件，单元测试的插件，打包的插件，。。。Maven可以被理解成管理这些插件的框架。

3）Declarative Execution (声明式执行)：Maven中所有的插件都是通过在POM中声明来定义的。Maven会理解所有在POM中的声明，并执行相应的插件。

src/main/java - 存放项目.java文件；

src/main/resources - 存放项目资源文件；

src/test/java - 存放测试类.java文件；

src/test/resources - 存放测试资源文件；

target - 项目输出目录；

pom.xml - Maven核心文件（Project Object Model）；

下载Maven：http://maven.apache.org/

2) 解压缩下载的zip文件到本地目录下，比如：D:\Maven

3) 添加D:\Maven\bin到环境变量PATH中

4) 在命令行下运行:

mvn -version  或者 mvn –v

一些Maven命令

编译: mvn compile

单元测试: mvn test

构建并打包： mvn package

清理： mvn clean

安装 mvn clean install

maven 项目会有一个 pom.xml 文件， 在这个文件里面，只要你添加相应配置，他就会自动帮你下载相应 jar 包，不用你铺天盖地的到处搜索你需要的 jar 包了。

```
<dependency>
  <groupId>junit</groupId> 项目名
  <artifactId>junit</artifactId> 项目模块
  <version>3.8.1</version> 项目版本
  <scope>test</scope>
</dependency>
```

maven都会通过，项目名-项目模块-项目版本来maven在互联网上的代码库中下载相应jar包。

所有的POM文件要求有projec节点和三个必须字段：groupId, artifactId,version。

在仓库中项目的标识为groupId:artifactId:version。

POM.xml的根节点是project并且其下有三个主要的子节点：

节点　　描述

groupId　项目组织的Id。通常在一个项目或者一个组织之中，这个Id是唯一的。例如，某个Id为com.company.bank的银行组织包含所有银行相关的项目。

artifactId 项目的Id，通常是项目的名字。例如，consumer-banking。artifactId与groupId一起定义了仓库中项目构件的路径。

version　项目的版本。它与groupId一起，在项目构件仓库中用作区分不同的版本，例如：

com.company.bank:consumer-banking:1.0

com.company.bank:consumer-banking:1.1.

Eclipse 提供了一个极佳的插件 m2eclipse ， 可以无缝地把 Maven 和 Eclipse 集成在一起。

m2eclipse 的一些特性列出如下：

你可以从 Eclipse 中运行 Maven 目标操作。

你可以使用 Eclipse 自身的控制台查看 Maven 命令的输出。

你可以使用 IDE 更新 Maven 依赖。

你可以从 Eclipse 中启动 Maven 构建。

为基于 Maven pom.xml 文件的 Eclipse 构建路径做依赖管理。

解决来自 Eclipse 工作空间的 Maven 依赖，而无需安装依赖到本地 Maven 仓库中（需要依赖的项目在同一个工作空间中）。

自动从 Maven 远程仓库中下载所需依赖及源码。

提供了向导，可供创建 Maven 新项目和 pom.xml 文件以及为已存在的项目开启 Maven 支持。

提供了对 Maven 远程仓库中依赖的快速搜索。

SQL 的小题目，有兴趣可以看一下

有三个表

tblStudentNo:

| studentNo | lastName | firstName |
|---|---|---|
| 79004651 | Somers | Curtis |
| 96000001 | Unknown | |
| 96000002 | Day | Green |
| 96000003 | Oven | Beth |
| 96000004 | Lisp | Franz |

tblLabMark:

| studentNo | labNo | labMark |
|---|---|---|
| 79004651 | 1 | 6 |
| 79004651 | 2 | 7 |
| 96000001 | 1 | 6 |
| 96000002 | 2 | 7 |
| 96000003 | 2 | 8 |
| 96000005 | 2 | 7 |

tblTestMark:

| studentNo | testNo | testMark |
|---|---|---|
| 79004651 | 1 | 70 |
| 79004651 | 2 | 60 |
| 96000001 | 1 | 50 |
| 96000001 | 2 | 50 |
| 96000002 | 1 | 60 |
| 96000003 | 1 | 70 |
| 96000003 | 2 | 80 |
| 96000002 | 2 | 90 |

要求：

写一个SQL语句，可以很长，可以嵌套，要求如下：

Calculate term marks assuming the equation:
((Lab#1/10)*15) + ((Lab#2/10))*15) +((Midterm#1/100)*30)+((Midterm#2/100)*40)
Select the firstName, lastName and final mark of the each student using the above equation.