系统要求

2、假设公司组织结构为：

--总结理（工资5万）

----技术部门经理（工资2.5万）

------开发人员A（工资7500）

------开发人员B（工资6000）

----销售部门经理（工资底薪5000，完成业绩提成5000，未完成无提成）

------销售人员A（工资3500，完成业绩提成4000，未完成无提成）

------销售人员B（工资3000，完成业绩提成4000，未完成无提成）

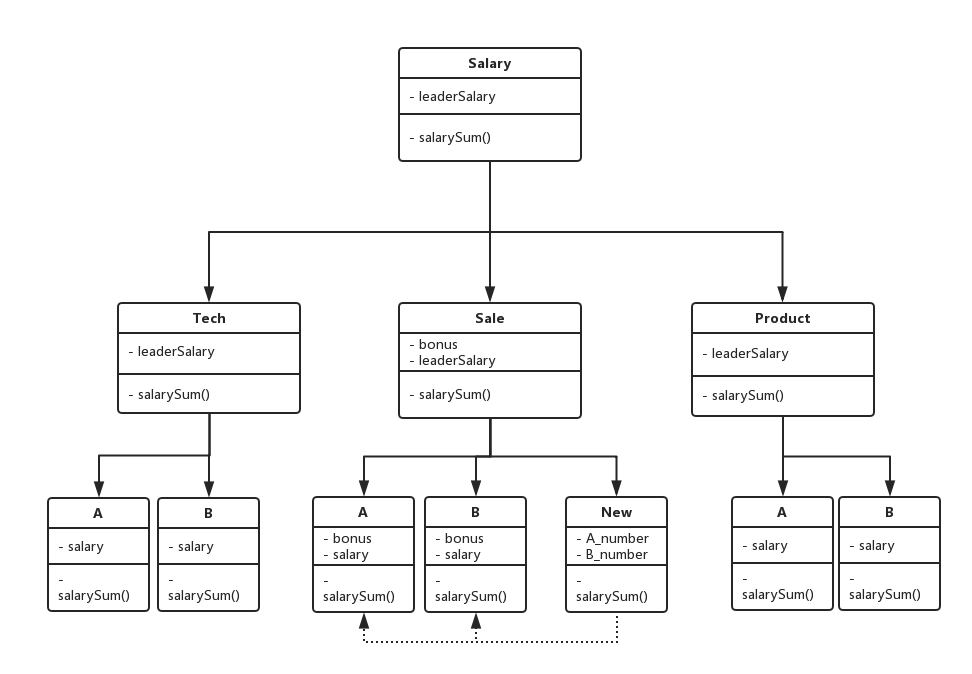
----产品部门经理（工资1.5万）

------产品设计人员A（工资8500）

------产品测试人员B（工资5000）

总经理直接领导技术部经理和销售部经理，技术部经理直接领导开发人员A和开发人员B。 销售部经理直接下属员工2名，随着公司规模增大，销售部门会新增销售员工。请考虑将来人员的变动性，计算整个公司的今年总工资状况。

UML类图



类图使用PrcessOn（https://www.processon.com）在线生成工具完成。

程序采用抽象工厂模式进行设计。

主要包含11个类，Salary、Tech、Tech\_A、Tech\_B、Product、Product\_A、Product\_B、 Sale、Sale\_A、Sale\_B、New。各类之间关系见上图。

这里需要注意，对于产品部门和技术部门，我们从题目中可以观察到其逻辑上是等价的，因此我们在程序的实现部分没有对产品部门的工资计算进行实现，并在以下文档中不再过多说明。

此外由于各类之间相似度较高，我们以下简化对各个类的说明。

Salary类用于对总经理的抽象，包含一个方法，一个属性：

一个属性为：

1. leaderSalary：标定其本身的薪资；

一个方法为：

（1） salarySum()：计算其本身与其依赖的各个类的薪资总和；

Tech类为技术部门的抽象，包含一个方法，一个属性；

Sale类为销售部门的抽象，包含一个方法，两个属性

两个属性为：

1. bonus：标定其本身的提成；
2. leaderSalary：标定其本身的薪资；

一个方法为：

1. salarySum()：计算其本身与下属新旧员工的薪资之和；

New类为新员工薪资计算功能的抽象，包含一个方法，两个属性。

两个属性为：

1. A\_number：记录新招入的A类销售人员的数目；
2. B\_number： 记录新招入的B类销售人员的数目；

一个方法为：

（1） salarySum()：计算新招入销售人员的薪资；

系统实现

该版本的系统使用C++实现。工程包含十七个文件（main.cpp new.cpp new.h salary.cpp salary.h tech.cpp tech.h tech\_a.cpp tech\_a.h tech\_b.cpp tech\_b.h sale.cpp sale.h sale\_a.cpp sale\_a.h sale\_b.cpp sale\_b.h）。

基于上述同样的原因，我们对各个文件的功能描述进行简化。

其中：

1. main.cpp: 包含简单用户引导界面实现；
2. salary.cpp & salary.h: 包含对总经理类的实现，从总经理的薪资计算开始自顶向下的薪资计算过程；
3. tech\_a.cpp & tech\_a.h：包含A类技术人员类的实现，开始自底向上薪资计算;
4. new.cpp & new.h：包含新员工薪资计算类的实现，依据输入决定新员工的数量并进行薪资计算；

薪资计算系统实现过程：从总经理的薪资计算，自顶向下从领导到员工计算整个公司的薪资，在计算销售部门的薪资时，提示输入各个类型新员工的数量并计算新员工的数量。

需要注意的实现细节如下：

1. 销售部门人员的提成是否存在由随机数模2的结果决定；
2. 我们默认新加入销售部门的员工的薪资类型与当前存在的薪资类型种类相同；
3. 我们默认大小领袖与员工的数目只有一位，且不会发生变化；

系统运行结果

