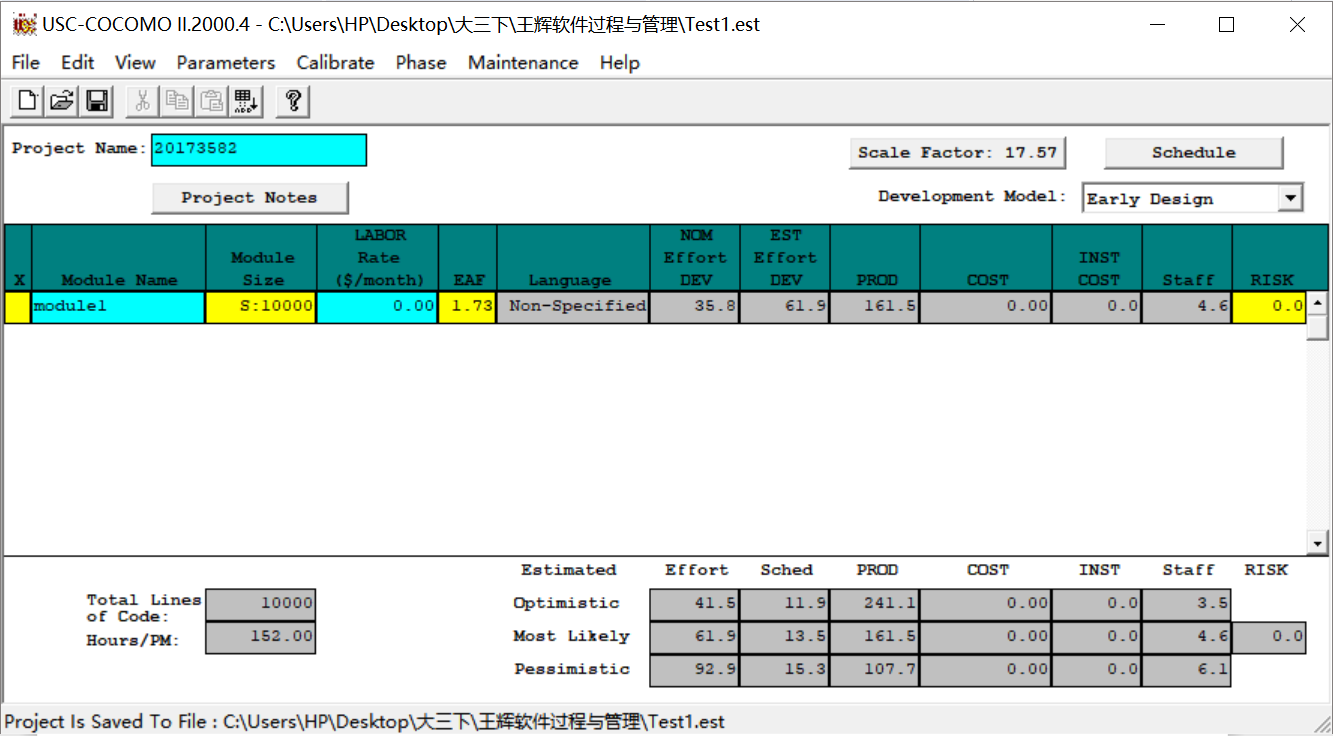
**实验报告**

**实验1：个人软件过程**

**[实验任务一]：**

1. 提交扩展名为est的COCOMO项目文件，并将结果进行截图。



。

[实验任务三]：

代码规范：

|  |  |
| --- | --- |
| ***类型*** | ***规范*** |
| 文件名 | 源文件以其最顶层的类名来命名，大小写敏感，文件扩展名为.java |
| 大括号 | 大括号与if, else, for, do, while语句一起使用，即使只有一条语句(或是空)，也应该把大括号写上。 |
| 块缩进 | 每当开始一个新的块，缩进增加2个空格，当块结束时，缩进返回先前的缩进级别。缩进级别适用于代码和注释。 |
| 注解 | 注解紧跟在文档块后面，应用于类、方法和构造函数，一个注解独占一行 |
| 块注释风格 | 块注释与其周围的代码在同一缩进级别。它们可以是/\* ... /风格，也可以是// ...风格。对于多行的/ ... /注释，后续行必须从开始， 并且与前一行的\*对齐。注释不要封闭在由星号或其它字符绘制的框架里。 |
| 类名，方法名，常量名等 | 以lowerCamelCase风格编写 |
| @override | 只要是合法的，就把@Override注解给用上 |
| 异常处理 | 对捕获的异常不做响应是极少正确的，不能忽视，如果它确实是不需要在catch块中做任何响应，需要做注释加以说明 |
| 排版 | 1、所有全局变量都要写在首位  2、引用包且不能放到代码中  3、当代码大于50个字符的时候换行  4、每一行只能有一条命令 |

代码评审标准：

|  |  |
| --- | --- |
| 完整性检查 | 代码是否完全实现了设计文档中所涉及的所有流程和功能点  代码是否已包含所有所需的业务日志、系统日志、异常日志，日志内容是否完整，日志文件配置是否正确。  代码是否使用缓存等，配置信息是否正确可配置。  代码中是否存在任何没有定义或没有引用到的变量、常数或数据类型等 |
| 一致性检查 | 代码的逻辑是否符合设计文档  代码中使用的格式、符号、结构等风格是否保持一致 |
| 正确性检查 | 代码是否符合制定的标准  所有的变量都被正确定义和使用  所有的注释都是准确的  所有的程序调用都使用了正确的参数个数. |
| 可修改性检查 | 代码涉及到的常量是否易于修改(如使用配置、定义为类常量、使用专门的常量  类等)  代码中是否包含了交叉说明或数据字典，以描述程序是如何对变量和常量进行  访问的  代码是否只有一个出口和一个入口(严重的异常处理除外) . |
| 可预测性检查 | 代码所用的开发语言是否具有定义良好的语法和语义  是否代码避免了依赖于开发语言缺省提供的功能  代码是否无意中陷入了死循环.  代码是否避免了无穷递归 |
| 健壮性检查 | 代码是否采取措施避免运行时错误(如数组边界溢出、被零除、值越界、堆栈溢出等) |
| 结构性检查 | 程序的每个功能是否都作为一个可辩识的代码块存在  循环是否只有一个入口 |
| 可追溯性检查 | 代码是否对每个程序进行了唯一标识  是否有一个交叉引用的框架可以用来在代码和开发文档之间相互对应  代码是否包括一个修订历史记录，记录中对代码的修改和原因都有记录  是否所有的安全功能都有标识 |
| 可理解性检查 | 注释是否足够清晰的描述每个子程序  是否使用到不明确或不必要的复杂代码，它们是否被清楚的注释  使用一些统- -的格式化技巧(如缩进、空白等)用来增强代码的清晰度  是否在定义命名规则时采用了便于记忆，反映类型等方法  每个变量都定义了合法的取值范围  代码中的算法是否符合开发文档中描述的数学模型 |
| 可验证性检查 | 代码中的实现技术是否便于测试  测试代码是否正确，是否覆盖所有流程 |
|  |  |