国际化流程

国际化流程

1.1 软件开发流程

1.1.1 课程目标

- 掌握软件开发流程理论知识

1.1.2 知识架构树

- 软件开发流程简介
- 软件开发经典瀑布模型

1.1.3 理论知识

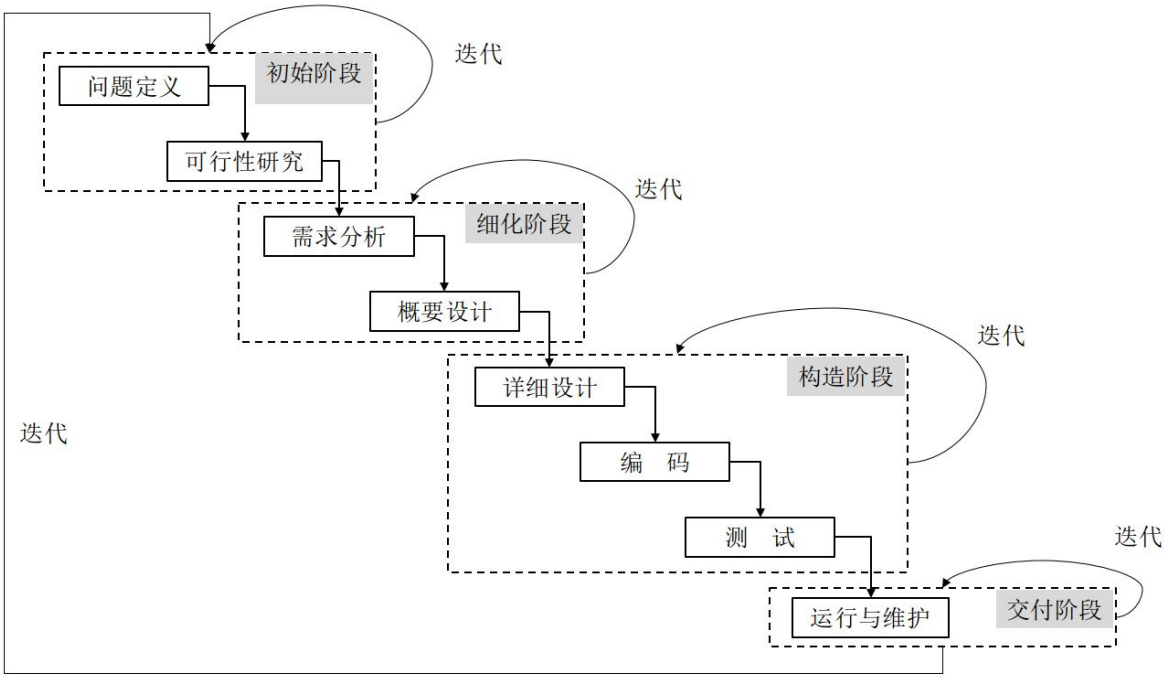
1.1.3.1 开发流程简介

> 软件开发流程是软件从无到有所要经过的过程。整个过程分为不同的阶段，每个阶段需要的工作内容和工作量均有所不同。不管是国内开发，还是欧美开发，还是对日开发，流程基本一致。其中些许的区别在于欧美开发要求英语技能，对日开发要求日语技能，和国内开发相比，多了语言技能项。
>
> 对日开发过程中需要读懂日语式样书，将日语翻译转换成需求，形成代码。这中间还需要遵循一整套开发流程，以保证代码的品质。目前所有的项目开发，都要向规范的开发流程靠拢。

1.1.3.2 软件开发流程阶段

1. **市场调研**
2. **需求分析**
3. **概要设计**
4. **详细设计**
5. **编码**
6. **测试**
7. **上线**
8. **维护**

软件开发经典瀑布模型：



1.1.4 综合实验

1.1.5 作业实践

- 掌握软件开发流程理论知识

1.1.6 面试宝典

1.1.7 拓展资料

1.2 对日流程：简介

1.2.1 课程目标

- 掌握对日开发流程
- 掌握对日开发工作内容

1.2.2 知识架构树

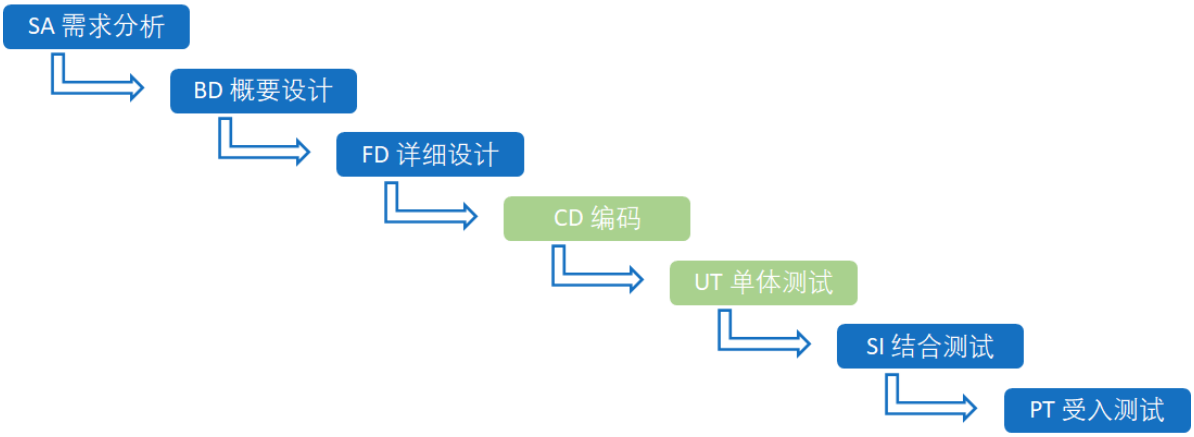
- 对日开发简介
- 开发环境准备
- 项目资料获取
- 工作内容明确
- 项目会议
- 信息安全
- 项目福利
- 开发流程

1.2.3 理论知识

1.2.3.1 对日开发介绍

> 对日流程就是在做对日项目中需要遵守的开发流程。整个流程划分为几个阶段：需求分析、概要设计、详细设计、编码、单体测试、结合测试、受入测试。**每个阶段都有核心产物以及附加产物产出。**核心产物即为每个阶段的核心产出物，例如开发阶段的代码。附加产物即为对核心产物质量保证的确认文档，例如：QA 票、Review 票等。
>
> 整个流程目前已经规范化，每个阶段都有相对固定的工作内容，按部就班，出错的概率相对较小。这也是对日项目品质较高的原因之一。
>
> 对日项目中不可或缺的是日语能力，整个流程过程中都要在日文系统的环境下进行，需要阅读日文文档，将日文内容翻译为中文，在转化为需求进行开发。这也就要求对日开发人员同样也需要具备较强的需求理解和编码能力。需要注意的是对日项目中的日语能力偏重于计算机日语，毕竟是在日文系统环境下进行编码工作。当然基础日语也不可或缺，整个流程中也需要书写一些日文，需要基础语法的支持。

对日软件开发阶段划分：



> 作为初级人员，能参与到的阶段为**编码阶段**和**单体测试阶段。**其余的阶段都需要3年及以上工作经验。

1.2.3.2 开发环境准备

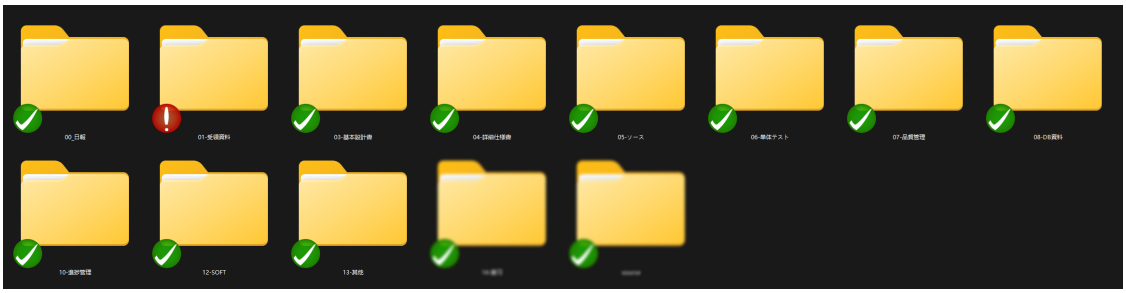
> 入项之后首要工作就是搭建开发环境，不同的项目搭建方式不同，可以划分如下情况：
>
> 1. 参照环境构建手顺（说明文档）搭建环境
>
>   繁琐、费时、出错率高、硬件要求正常、资料易泄露
> 2. **使用虚拟机加载环境镜像搭建环境**
>
>   相对简单、不费时、对硬件要求较高、资料易泄露
> 3. **采用远程方式**
>
>   简单、不费时、出错率低、资料不易泄露

开发环境要求：

1. 电脑设备：建议 8GB 以上运行内存、500GB 以上存储内存；
2. 日文系统（安装语言包、**安装日文系统**）：Windows 平台、Linux 平台；
3. 日文输入法（百度日文输入法、**微软自带输入法**）；
4. 日文版 office（安装语言包、**安装日版 office**）：Excel、Word、Power Point（PPT）；
5. 日文翻译软件：有道词典、百度翻译、灵格斯等；

6. 开发工具：JDK、数据库、数据库连接工具、IDE（idea、eclipse）、版本控制工具（Git、SVN）、虚拟机、文本编辑工具（NP++）等；
7. 网络：内网（局域网）、外网（广域网）；
8. 通讯工具：内网工具（飞秋）、外网工具（QQ、微信等）；

1.2.3.3 资料获取和确认

> 项目中的资料中类较多，可以划分为：受领资料、进度表、式样书、DB设计、软件工具、日报等。
>
> 00_日報 01-受領資料：项目的学习资料，例如：项目的构建说明等。 03-基本設計書：概要设计文档。 04-詳細仕様書：详细设计文档。 05-ソース：代码目录。 06-単体テスト：单元测试相关目录。 07-品質管理：QA 08-DB資料：数据库设计 10-進捗管理：进度表 12-SOFT：软件 13-其他：其他
>
> 

1. 项目资料一般采用版本控制工具的形式管理，入项的第一天会分配服务器地址和账号密码。
2. 版本控制工具的账号密码不会按照人员划分，一般是一个项目组一个。这也就要求在提交资料时必须填写备注，注明日期、提交人员、提交原因等。
3. **每天到工位的首要任务需要更新资料和代码，以保证开发过程中参照的是最新资料，避免比必要的返工。**
4. **确认资料的正确性和完整性，有问题及时和组长或者项目管理者反馈。**
5. **提交资料务必再三确认，避免提交错误信息，或者与其他人冲突。**

1.2.3.4 工作内容明确

> 明确工作内容，就要看进度表。进度表作为项目中的核心管理文档，是非常重要的，一般采用 Excel 形式管理。**进度表中规划了式样书（作业内容）、担当者（开发者）、开发周期、测试周期、Review 周期、备注等内容。**（对日开发中式样以**“本”**为单位，一个Excel 式样书就是一本式样书。换言之就是一个需求是一个 Excel。）
>
> 担当者（开发人员）要根据进度表中的安排，明确工作内容（多少本式样书、多少个需求），并找到式样书进行浏览（有可能还未提供式样书），估算工作量，同时和进度表中的周期对比，有问题的安排要及时和组长/领导反馈，避免出现任务分配不当、任务重复、任务延迟等情况。
>
> 参照进度表按部就班开展工作，工作完成情况要及时在进度表中体现（如有问题需要标注原因），便于项目管理者实时了解工作动态，安排下一步工作。
>
> 进度汇报时段可以在当天下班之前，也可以在第二天上班的第一时间。
>
> 进度汇报要明确：完成多少、未完成多少、未完成的工作内容需要多长时间可以完成、遇到什么问题、需要什么支持（帮助）等。

1.2.3.5 项目会议

> 项目组的会议也是必不可少的，会议上需要确认当前工作内容和下一步工作安排。参会人员可以是全员（项目组人数较少时）也可以时核心人员（组长、技术担当等）。会议时间不易过长，避免耽误当天工作。所以也就要求开发人员必须详细了解自己的工作情况。会议一般分为晨会和夕会，当然，必要时也可以开展周会。
>
> 晨会：回顾昨天工作、计划今天工作、阐述昨天的遗留问题以及需要的帮助。
>
> 夕会：回顾当天工作、计划明天工作、阐述今天的遗留问题以及需要的帮助。
>
> 周会：回顾本周工作、计划下周工作、阐述本周的遗留问题以及需要的帮助。

1.2.3.6 信息安全

> 信息安全是不得不提的重要环节，所有的项目在上线之前都要求严格保密，条件允许的情况下，入项之前必须签订保密协议，具有法律效益。尤其是对于政府类、银行类的项目，保密更加严格。保密的手段也是多样化，以保证信息的安全。

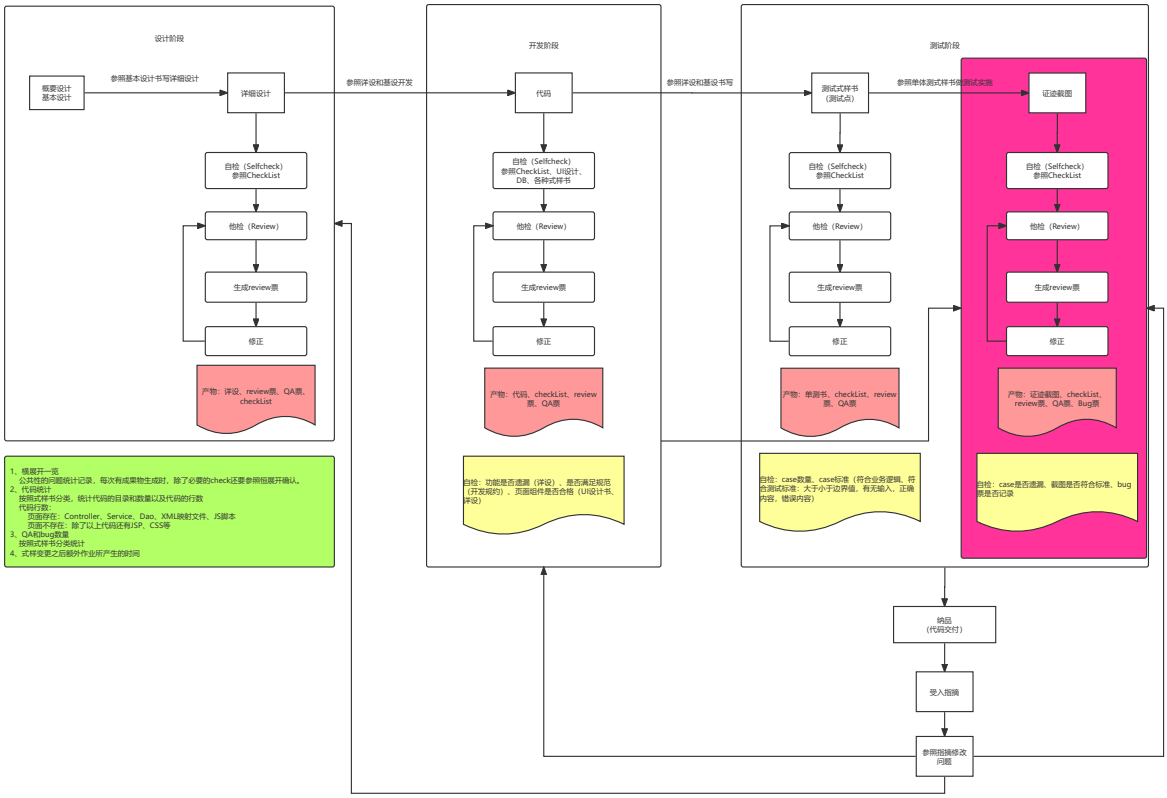
保障信息安全的手段：

1. 门禁：开发人员需要录入指纹或者携带门禁卡方可进入开发场地。
2. 监控：开发场地布置监控设备。
3. 移动设备：禁止携带移动设备（U盘、移动硬盘）进入开发场地，严格的时候禁止携带手机。
4. 电脑设备：禁止使用个人电脑接入公司网络，统一使用公司配备电脑，所有的电脑统一安装杀毒软件，并进行统一部署管理，禁用 USB 接口和串口。
5. 网络：一般以局域网为主。如果有外网禁止登陆微信、QQ 等实时聊天工具，同时禁用邮箱功能，避免资料泄露。
6. 翻译软件：翻译时避免整句整篇翻译，有些翻译软件会收录信息，容易造成资料泄露。建议按单词进行翻译。
7. 系统软件：使用正版授权的系统和软件，避免后门隐患。

1.2.3.7 项目福利

> 为了调动员工积极性，提高工作效率，部分项目在开始之前会举办项目活动，形式不定，可以是聚餐、可以是休闲娱乐。项目完成的出色，在项目中或项目结束后也会举办项目活动，以犒劳员工。当然这中间少不了的就是商务礼仪了。

1.2.3.8 对日开发流程



1.2.4 综合实验

1.2.5 作业实践

1.2.6 面试宝典

1.2.7 拓展资料

1.3 对日流程：开发流程

1.3.1 课程目标

- 掌握对日开发流程
- 掌握开发阶段产物以及含义

1.3.2 知识架构树

- 开发参照文档
- 式样书理解
- 编码制造
- 质量保障
- 自检、他检

1.3.3 理论知识

> 开发即参照相关文档，将用户需求转换为代码并实现功能的过程。

1.3.3.1 开发参照文档

> 开发文档的参照各个文档之间的搭配，比如：详细设计中的表信息，可以搭配 DB 设计书。详细设计中的提示信息，可以搭配信息设计一栏。

1. **开发规约：**开发过程中的命名规约、编码规范等。
    - 文件构造（类文件的组成）：导包、注释、代码、变量声明等的书写顺序。
    - 缩进：不推荐使用 TAB 键，要使用 4 个半角空格。
    - 注释：文档注释、块注释、单行注释、注释内容（功能描述、异常、返回值、作者、版本）。
    - 变量定义：同类型的不建议定义为一行。不同访问修饰符变量的书写顺序（public > protected > private）

```
// 错误
int i, j;
// 正确
int i;
int j;

// public > protected > private
public String a;
protected String b;
private String c;
```

    - 常量定义：所有单词大写，多个单词之间使用下划线分割。

```
// 文件后缀的常量定理
public static final String FILE_SUFFIX = ".jpg";
```

    - 空格/空行：运算符两边加空格。方法与方法之间加空行。提高整体的阅读性。
    - 方法定义：方法名使用动词，且遵循驼峰命名法，例如：setXxxx、getXxxx、findXxxx、editXxxx、updateXxxx 等。
    - 类型转换：使用合适的类型进行转换，避免出现丢失精度的问题。
    - 异常处理：处理合适的异常，不建议处理大异常。
    - 继承/实现关系：避免复杂（乱）的继承/实现关系。
    - 性能处理：挑选合适的判断/循环语句。对于不使用的变量及时释放空间。对于容器（数组、集合）使用完毕之后及时清空。
    - 行代码量：每行的代码不要超过 200（结合项目要求） 个字，超过的地方要换行。在运算符前面换行。在调用方法的点前面换行。

```
// 在运算符前面换行
if(a == b
   && c == d
   || e == f){

}
// 在调用方法的点前面换行。
"".a()
  .b()
  .c();
```

- 类/方法代码量：每个类/方法不建议太长，例如：类 500 行，方法 200 行，超过的部分要拆开写。
- ……

2. **UI 基准设计书**：页面的规划文档，对页面风格统一的定义。包括：页面之间的跳转（关系），页面的样式（风格），页面组件的布局等。

```
按钮风格：修改：蓝色 添加：绿色 删除：红色 查询：青色
表格风格：
超链接风格：
信息提示风格：
...
```

3. **DB 设计书**：数据库设计文档，包含表结构和字段（名称、属性、长度等），表之间的关系（ER 图）等。
4. **Session 设计书**：用户登录成功存放在 Session 中的信息。
5. **提示信息一览**：系统的提示信息、警告信息、错误信息等的定义。通常是 k-v 形式存储（properties 文件），Message ID 对应一个提示信息。

```
LMS-000001=系统异常，请联系管理员！
LMS-000002=网络异常，请稍后再试！
LMS-000003=添加成功
LMS-000004=修改成功
LMS-000005=删除成功
LMS-000006=操作成功
....
```

6. **进度表**：整个工作内容安排，包含项目整体周期、每个人的任务分配等。
7. **横展开一览**：项目组共性问题、错误率高的问题的记录文档。
8. **概要设计**：需求大概的功能描述文档。基础功能、功能之间的关系等。
9. **详细设计**：需求的详细描述，具体每一步的操作，用到的表、字段等。

1.3.3.2 式样书理解

> 理解式样书是对日开发过程中非常重要的一个环节，这个阶段需要开发人员参照相关文档，翻译式样书（将日文描述转化为需求，再将需求转化为代码）。这个过程对于初次接触的开发者来说是有些难度的，因为他们把大部分的精力都放在了日文的直译上，忽略了需求，殊不知需求才是重点。**所以在理解翻译式样书的时候切记不要逐字逐句翻译，以涵盖意思为主（添加、删除、修改、查询），弄清楚需求，肯定和否定，被动和主动即可。项目中行业的专有名词不建议翻译，只需要作为名词理解。**
>
> 另外，对日项目中需要参照的文档较多，不知道这些文档之间的关联关系，所以很多开发者面对这些文档无无从下手。**首先需要知道这些文档的作用，包含了哪些内容；其次按照式样书的书写顺序，需要什么就从指定的文档中找即可。**例如：式样中描述了SQL，SQL 包含表的信息，所以要参照 DB 设计书。
>
> 式样理解也是对开发人员经验的考验和提升，有经验的开发人员通过粗略的阅读就能知道大概的需求。

式样理解步骤：

1. 整体翻看式样书（一般是 excel），了解式样书的组成（sheet 组成）以及每个部分的作用（sheet 作用），找到核心部分。
2. 参照静态页面、概要设计和详细设计，宏观了解需求（页面的样子，大概什么功能）。
3. 以需求和式样核心部分为出发点，参照相关内容逐步翻译式样（切记不可逐字逐句翻译）转化需求，形成编码思路。
4. 宏观角度查看是否有式样（需求文档）问题，例如：缺表、缺字段、需求描述不清、需求遗漏、需求重复等问题。如有这些问题需要提 QA 票。

1.3.3.3 编码制造

> 在理解了式样之后，参照相关文档进行开发制造，也是开发阶段的核心工作。开发制造要求开发人员能整正确理解需求开发，同时也要求个人的理解，因为式样书上的需求并不完全正确。

工程目录：

```
控制层：Controller
业务层：Service
持久层：Dao/Mapper
实体类：数据模型，主要作用是封装数据和数据传输
    和DB传输数据：Entity/Bean/Model（和DB的字段一致，只多不少）
    和页面传输数据：DTO（和页面的字段一致，只多不少）
异常：Exception
工具包：Util
注解：Annotation
拦截器：Interceptor
```

文件命名：

> 类名、页面名称，依据式样书中的技能ID和驼峰命名法加上分层标识即可。

注释编写技巧：

1. 尽可能从式样书中复制，进行修改。
2. 可以采用符号标识。
3. 通过关键字标识。
4. 借助翻译软件。

编码流程：

1. 创建类，命名参照式样书中的技能ID。
2. 导入静态页面，一般情况下项目会提供静态 HTML、CSS 样式、共性脚本。
3. **参照式样和开发规约等文档按部就班开发。**
4. 遇到问题及时和组长沟通，提出 QA。

1.3.3.4 QA 质量保障

> QA（Quality Assurance），质量保证，其定义为“为了提供足够的信任表明实体能够满足品质要求，而在质量管理体系中实施并根据需要进行证实的全部有计划和有系统的活动。”
>
> 在对日项目中 QA 是非常重要的质量保障手段，整个项目周期（从入项到退项）都可以通过 QA 和客户方沟通，答疑解惑，以保障软件的质量。
>
> QA 在项目中存在放在【品质管理、QA】目录，是 一个 Excel 文件。

QA 文件组成：

1. **QA号**：是QA问题的代号。

2. **要件：**出现问题的文档名字或者技能的ID
3. **问题：**描述具体问题和自己的见解。
4. **発行者、発行日**：问题提出的人员和日期。
5. **添付資料**：附件，文字描述不清楚的可以采用附件（画图、表格）的形式提交。
6. **回答内容**：客户方对问题的回答。
7. **QA 状态：**已提出、已回答、再提出、关闭。

QA 的产生阶段：整个项目周期。

1. 式样理解阶段：明显的文档书写问题以及逻辑问题可以提出 QA
2. 代码开发阶段：业务逻辑问题、数据库问题等可以提出 QA
3. 测试阶段：开发阶段没有发现，测试阶段测试出的问题可以提出 QA

QA 的提出流程：

1. 发现问题，总结问题。
2. 和组长确认问题是否需要提出 QA。
3. 在 QA 管理文档中描述具体问题，并给出合理的解决方案。
4. 在相关文档（进度表、代码、测试式样书等）中记录 QA 号，汇报进度时也需要汇报 QA。Java文件中可以使用保留关键字`TODO`标记QA

```
// TODO QA 1001
```

5. 等待 QA 回答（注意：等待的过程可以做其他没有做完的事情，或者做下一个功能）。
6. 依据 QA 回答内容解决问题，不能解决的可以再次提出（可以在原有QA中继续追加，也可以重新开一个QA）。
7. **关闭 QA，删除相关文档中的 QA 记录。**

常见的 QA 问题类型：

1. 式样书描述模糊不清。
2. 式样书叙述功能遗漏、重复。
3. 式样书叙述功能违背正常逻辑。
4. 式样书叙述表结构和实际表结构有出入（表不存在、字段不存在、类型不匹配、长度不一致等）。
5. 复杂且专业的业务逻辑，项目组无法实现。

QA 编写技巧：

1. 套模板，参照其他的QA
2. 借助翻译软件（逐字翻译）
3. 将QA的中文写法写出来，将词语替换成日文，语法找组长帮忙改
4. **编写QA核心思路：找具体位置，描述问题，给出解决方案。**

1.3.3.5 自检（Self Check）

> 自检是对开发的代码进行自我审查的阶段，通过项目组提供的审查文档，挨个项目确认是否满足要求。没有问题的画【〇（まる）】，有问题的画 【×（ばつ）】，跟当前功能不想管的画【-】。
> **自检文档也是开发阶段的重要产物。**
>
> **自检主要从开发规约（格式、注释、编码规范等）、功能实现两个方面入手。**

自检文档标识：

> 〇：表示该项目在代码中存在，且正确。

> ×：表示该确认项在代码中存在，但错误。
>
> -：表示该确认项在代码中不存在。

1.3.3.6 他检（Review）

> 每个开发者都会有先入为主的思想，对个人代码中的问题并不能及时发现。Review 是安排有经验人员（组长、其他资历较高的组员），以三方的角度再次从开发规约（格式、注释、编码规范等）、功能实现两个方面入手，结合 Review 者的经验对代码进行再次检查，并将所发现的问题记录形成文档，这个文档就是 **Review 票**，也称为**指摘（してき）票**。Review 票也是开发阶段的一个产物。
>
> 开发人员需要根据 Review 票中记录的问题进行代码修正，修正结束后， Review 者需要继续检查，在保证问题全部修正的基础上，不再出现新的问题。如果出现新的问题，要继续 Review 记录，让开发人员继续修改，如此循环往复，直至没有问题，开发阶段才算结束。

代码 Review 的技巧：

1. Review 之前先更新最新资料（式样书、QA票、代码等）。
2. 先看代码规范问题（文件名字、属性命名、注释等）和一眼能确认的问题。
    - 类角度

```
1、是否报错
2、多余的导包信息
3、类注释、类名、属性注释、方法注释、空行、缩进
4、变量命名、空格
5、功能
    5.1 先执行，看是否有明显报错。
    5.2 对照式样书看逻辑是否正确，不要逐字逐句看，看大概逻辑。
```

    - 页面角度：页面布局、缩进、空行、注释（不要出现HTML注释）
3. 再集合式样书和相关文档看业务逻辑。

Review 票的编写的技巧：

1. 边看代码，边记录；
2. 在时间紧迫的情况下，可以在文本编辑器中记录，然后发给开发者进行修改。Review 者要整理 Review 票（中文翻译为日文）提交。
3. 在时间不紧张的情况下，要求在 Review 票中挨个记录。

对应指摘的技巧：

1. Review 票的问题点从上往下看，但是修改代码要从下往上改。
2. 先修改代码，再填写 Review 票。

1.3.3.7 开发阶段收尾

1. 整理开发阶段的所有产物（**代码**、QA、Self Check 文档、Review 票）进行提交
2. 及时更新进度，在进度表中填写开发结束时间，并给组长汇报。
3. **开发阶段结束有可能还存在 QA，这个QA是对功能影响较小，也可以继续下一个阶段的任务，等待 QA 回答之后修改代码即可。注意，一定要在相关文档中体现 QA。**

1.3.4 综合实验

1.3.5 作业实践

1.3.6 面试宝典

1.3.6.1 常见面试题

- 开发阶段都有写产物？并说明产物的作用。
- 开发需要参照的文档有哪些?
- QA 在什么阶段产生?
- QA 文档包含哪些项目?
- 在项目中提过哪些QA?
- 项目中遇到问题询问组长之后不用提QA，此时应该怎么办?
    - 按照式样书和组长的建议开发即可。
- 自检文档都要做哪些项目的确认?
    - 格式：命名、注释、语法选择...
    - 功能：多的、少的、错的...
- 简述一下开发流程。
    1. 更新资料，根据进度表确认个人工作内容。
    2. 理解式样书，需要搭配 DB 资料、UI 设计等相关文档。如果发现问题，可以提 QA。
    3. 参考开发规约进行代码的开发。过程中遇到问题可以提 QA（并不是所有的问题都要提）。
    4. 参照自检文档对代码进行检查。
    5. 代码的Review，以及修正。
    6. 更新进度。
- 单词：
    - 开发（開発「かいはつ」、実装「じっそう」、コーディング）、测试（テスト）
    - 添加（追加「ついか」）、修改（修正「しゅうせい」）、删除（削除「さくじょ」）、查询（検索「けんさく」）
    - 跳转（遷移「せんい」）、展示（表示「ひょうじ」）、条件（条件「じょうけん」）

1.3.7 拓展资料

1.4 对日流程：测试流程

1.4.1 课程目标

1.4.2 知识架构树

1.4.3 理论知识

1.4.3.1 测试前导知识

测试的原则：

- 保证测试的覆盖程度，但穷举测试是不可能的
- 所有的测试都应追溯到用户需求

- 越早测试越好，测试过程与开发过程应是相结合的
- 测试的规模由小而大，从单体测试到系统测试
- 为了尽可能地发现错误，应该由独立的第三方来测试
- 不能为了便于测试擅自修改程序
- 既应该测试软件该做什么也应该测试软件不该做什么

测试方案与分析：

1. 交叉测试：两个开发人员互相测试；
2. 专职测试：项目组安排专人做测试，也是理想化的测试方式；
3. 自己自测：开发者测试自己测试自己的功能；

	时间	问题
交叉测试	需要重新理解式样，耗费时间。	三方思想，可以发现非主观问题。
专职测试	需要重新理解式样，耗费时间。	三方思想，可以发现非主观问题。
自己自测	不需要再次理解式样，耗时较短。	主观思想，不容易发现问题。

1.4.3.1 单体测试式样书做成

测试式样书自检：

1. 确认式样书上的功能是否都覆盖
2. 正常系和异常系
3. 测试点数量是否达标/超标
4. 测试式样书上标识的技能名等信息是否有误

测试式样书中的测试点：

1. 根据需求的负责都进行推断（1000行代码：100测试点：10bug）
2. 测试点不够的情况下，可以将一些大的测试点拆开写。
3. 测试点量超的离谱的情况下要择优选择。

1.4.3.2 测试实施

> 就是根据测试式样书中点测试点，一个一个进行测试。并将页面的操作过程，通过图片的形式记录在 Excel 文档中，这个文档称为证迹截图（エビデンス）。证迹截图是测试实施阶段的核心产物。

造数据：

1. 如果原有数据能满足要求直接使用即可。
2. 造数据不能动原有数据（不能去修改原有数据），将原有数据复制一份，按照需求修改。
3. 既要造满足条件的数据，也要造不满足条件的数据（包含查询条件和关联条件）。

测试截图工具：

1. 项目组提供工具
2. Snipaste 
3. FastStone Capture Portable（红绿小工具） 

4. 腾讯截图工具

截图的要求：

1. 图片的质量不能太高，以能看清楚为准。
2. 图片中只能出现功能相关的内容（网页的主体部分）。例如：任务栏、输入法的悬浮框、浏览器的地址栏（任务栏）、通讯工具的隐藏框等。
3. 图片中可以使用红框对关键点进行标注，不要直接画在图上，使用Excel的框标记。

证迹文档的要求：

1. 可以使用红框对测试点的关键信息进行标注。
2. 图片之间如果有关联关系，使用箭头进行指向。
3. 对于有数据操作类的测试，需要将测试前后的数据都贴出来进行对比。
4. 对于时间相关的测试数据结果和图片上的时间要对应。
5. 一个步骤一个截图，不能跳着截图。

测试技巧：

1. 测试位数的点，可以才有有规律的一段内容进行重复。
2. 有数据的测试，要提前将数据粘贴到证迹文档中。

证迹的自检：

证迹的他检：

1.4.3.3 Bug

> Bug也叫障害，在测试的实施阶段产生。和测试点的预期结果不一样的情况都可以称为Bug。比如：页面报错、后台报错、页面布局问题、页面格式问题等。有Bug的情况下需要记录Bug问题（Bug票、障害票），并让让开发人员进行修改，然后再次对当前测试点进行测试。

Bug的管理形式：

1. 采用Excel文档的形式管理
2. 采用Bug管理系统，例如：禅道、公司自己研发的系统。

Bug产生之后要做的事情：

1. 记录Bug生成Bug票和Bug号。
2. 在测试文档中对应的测试点后面标记Bug号。
3. 因为式样问题产生的Bug，可以同时提出QA。

Bug数量要求：

1. Bug 的数量不是越多越好，也不是越少越好。需要根据需求的复杂度进行推算（1000行代码：100测试点：10bug）
2. Bug 的数量不够时，可以将开发阶段出现的问题作为Bug

Bug票填写内容：

1. **Bug号（如果是系统会自动生成；Excel 需要手动自增；）**
2. **机能名、机能ID**
3. **问题发生的过程（什么情况下做了什么操作出现了什么现象）**
4. 提出者、提出日期
5. **指向者（当前机能的开发人员）**
6. 优先级
7. Bug的修改内容
8. Bug的状态
9. Bug产生的原因（代码问题、式样问题、自身问题等）

Bug修复后的测试：

1. 只测和 Bug 相关的的测试点。
2. 之前产生 Bug 的截图证据要留存。新测试的可以新开一个Sheet 进行截图，并说明 Bug 情况。

二重送信制御：

表单重复提交的控制。表单重复提交会导致数据重复添加的问题。

二重送信制御实现：

1. 通过脚本控制点击提交之后将提交按钮禁用掉。
2. 采用拦截器的形式对提交拦截判断。

1.4.4 综合实验

1.4.5 作业实践

1.4.6 面试宝典

1.4.7 拓展资料

1.5 对日流程：纳品

> 按照和客户约定的是时间，按时将开发阶段和测试阶段的产物整体分类打包，交付给客户方。客户方对成果进行复查，如果查出问题，会反馈给项目组，进行修改。那么这之间会有一个**受入指摘**发送给项目组，项目组安排开发人员进行对应，对应完成之后再次测试，再次纳品。
>
> 内部指摘：项目内的指摘；
>
> 外部指摘：客户方的指摘；

纳品的特例：

1. 带 QA 纳品，也属于正常。
2. 先提交代码，随后再提交测试结果。

1.6 对日流程：拓展

开发IDE（idea）：

1. 格式化代码：ctrl + alt + L
2. 复制行：ctrl + D
3. 删除行：ctrl + Y
4. 撤回：ctrl + Z
5. 反向撤回：ctrl + shift + Z
6. 生成方法：alt + Insert
7. 抽取方法：ctrl + alt + M
8. 代码生成：ctrl + alt + T
9. 文本搜索：ctrl + F
10. 替换：ctrl + R
11. 重命变量/文件名：shift + F6
12. 全局搜索：shift * 2
13. 提示：alt + 回车
14. 大小写转换：ctrl + shift + U
15. 打开IDE设置：ctrl + alt + S
16. 打开工程设置：ctrl + shift + alt + S

17. Debug：F8（下一步/进入）F9（下一个断点）

版本控制工具（Git、SVN）：

1. 提交规则：提交时添加注释，标明提交时间、提交人员、提交原因
2. 更新规则：实时更新
3. 公共文件操作：
    1. 更新
    2. 加锁，加注释
    3. 修改
    4. 提交，加注释
4. 冲突文件操作：
    1. 先备份
    2. 删除冲突内容
    3. 更新最新资料
    4. 添加个人修改内容
    5. 提交，加注释
5. 恢复文件：丢弃修改内容，恢复到当前版本。

文本编辑工具（NP++）；

1. 纵向造作：alt + 鼠标下拉
2. 全文检索：ctrl + F
3. 全文替换：ctrl + H
4. 大写转换：ctrl + shift + U
5. 小写转换：ctrl + U
6. 上移行：ctrl + shift + 上键
7. 下移行：ctrl + shift + 下键
8. 文件夹检索：ctrl + shift + F

Excel 操作：

1. 系统时间公式：now()
2. 行号公式：row()
3. 求和公式：sum()
4. 求和快捷键：alt + 等号
5. 平均值公式：average()
6. 统计公式：count()，countif()，countifs()
7. 查找公式：vlookup()
8. 填充当前时间：ctrl + ;
9. 添加行：ctrl + 加号
10. 删除行：ctrl + 减号
11. 调出单元格格式设置：ctrl + 1
12. 单元格内容画删除线：ctrl + 5
13. 另存为：F12
14. 行列转换：选择性粘贴 ---- 转置
15. 单元格获取光标：F2
16. 单元格内换行：alt + 回车

代码统计：

> 统计个人开发了多少行代码，自己写的都要统计。

	统计	不统计
类/接口	控制类、业务类、持久类、拦截器类、异常类等	实体类
静态资源	JS脚本	静态页面、CSS