实验十 软件体系结构设计（二）

实验目的：

1. 体系结构风格和视图特点

2. 研究经典软件体系结构案例

3. 继续补充和修改自己项目的SAD

实验内容：

1. 对比书上各种软件体系结构风格和视图特点，思考自己项目属于哪种设计风格？

网上搜索最新的软件体系结构资料，如MVC、Kruchten 4+1视图等。

2. 参阅课本和网上资料，研究经典软件体系结构案例KWIC。

An Introduction to Software Architecture，4.1节

On-the-Criteria-To-Be-Used-in-Decomposing-Systems-into-Modules（Example System 1）

http://www.cs.cmu.edu/~ModProb/index.html

针对KWIC和自己项目，参考课本ch5 表5-3，小组成员每人给几种不同的体系结构风格设计打分，评最佳。

3. 补充和修改自己项目的SAD

记录项目及小组每个人工作的进度、里程碑、工作量的跟踪图或表，将其保存到每个小组选定的协作开发平台上，每周更新。

1. 对比书上各种软件体系结构风格和视图特点，思考自己项目属于哪种设计风格？

网上搜索最新的软件体系结构资料，如MVC、Kruchten 4+1视图等。

人事管理系统项目属于简约明了风格，这种风格以简单、直接、明了为主要特点，适用于操作流程相对简单、功能相对单一的应用系统。人事管理系统虽然功能较多，但每个功能模块的界面和操作都尽可能简洁明了，方便用户快速上手和使用。

MVC（Model-View-Controller）：

MVC是一种软件设计模式，用于将应用程序的逻辑、数据和界面显示进行分离。这种模式将应用程序分为三个核心部分：

Model（模型）：代表数据和业务规则。它负责在数据库中存储数据，处理与数据相关的逻辑（如验证规则）。

View（视图）：负责数据的显示。它知道如何从模型中获取数据，但不知道数据来自何处或如何被修改。

Controller（控制器）：是模型和视图之间的协调者。它接收用户的输入，决定调用哪个模型来处理请求，并选择哪个视图来显示结果。

MVC模式的主要优点是它提高了代码的可维护性、可重用性和可扩展性。通过将逻辑、数据和界面显示分离，MVC使得应用程序的各个部分可以独立开发和测试。

Kruchten的4+1视图模型：

Kruchten的4+1视图模型是一种软件体系结构描述方法，它将软件体系结构分为五个视图：逻辑视图、开发视图、进程视图、物理视图和场景视图（或用例视图）。

逻辑视图：关注系统的功能和结构，描述系统的模块、类、组件和它们之间的关系。它使用UML（统一建模语言）中的类图、组件图和包图等工具来表示。

开发视图：侧重于软件模块的组织和管理，以及开发环境中的软件源代码的组织结构。它描述了软件在开发环境中的分解情况，例如包和库等。

进程视图：关注系统的运行特性，包括系统并发性和分布性。它描述了系统组件之间的通信时序，数据的输入输出等。

物理视图：描述系统的运行环境，包括硬件、网络和分布式系统的节点等。它关注系统如何部署到物理环境中。

场景视图：也称为用例视图，它是其他四个视图的中心，描述了系统的最终需求和交互设计。它使用UML中的用例图来表示。

4+1视图模型的主要优点是它提供了多个角度来描述软件体系结构，使得开发团队可以更好地理解和沟通系统设计。每个视图关注系统的不同方面，通过提供特定的抽象层次，使得开发人员能够专注于他们关心的特定领域。

KWIC（Keyword-In-Context）索引系统是软件体系结构研究的经典案例。以下是对KWIC系统的一些详细研究：

1. **基本功能**：
   1. KWIC系统接受一些行作为输入，每行包含若干词，每个词由若干字符组成。
   2. 这些行可以循环移位，即每行的第一个词被删除并接到行末，形成新的行。
   3. KWIC系统的目标是输出所有行的各种移位情况，并且这些输出按照字母表顺序排列。
2. **体系结构设计**：
   1. **面向过程风格**：这是一种传统的体系结构风格，它以功能为中心，通过一系列的子程序（或模块）来实现系统的功能。在KWIC系统中，面向过程风格可以将各个功能模块划分为不同的子程序，例如输入模块、处理模块和输出模块。这种风格的优点是结构清晰，易于理解和维护，但缺乏灵活性和可重用性。
   2. **面向对象风格**：与面向过程风格不同，面向对象风格将系统划分为多个对象，每个对象都具有属性和方法。在KWIC系统中，面向对象风格可以将输入、处理和输出等功能划分为不同的对象，对象之间通过消息传递来实现协作。这种风格的优点是可重用性和灵活性强，易于扩展和维护。
3. **质量属性**：
   1. **可修改性**：系统应易于修改以适应算法的变化、数据表示方式的变化以及系统功能的可扩展性。
   2. **性能**：系统的时空复杂性应保持在可接受的范围内。
   3. **可复用性**：系统构建时应考虑组件的复用性，以减少开发时间和成本。
4. **非质量属性**：
   1. 输入：系统应能够处理用户指定的英文文本文件（.txt），包括指定文件的目录和文件名。
   2. 移位：系统应能够按照要求执行行的循环移位操作。
   3. 排序：系统应能够按照字母表顺序对输出进行排序。
   4. 输出：系统应能够将处理后的结果输出到用户指定的文件中。
5. **设计模式**：
   1. 在实现KWIC系统时，可以使用设计模式来提高代码的可复用性、可维护性、可读性和稳健性。例如，适配器模式可以用于将不同接口的类进行适配，使得它们能够协同工作。
6. **解决方案**：
   1. **面向过程方案**：虽然简单明了，但可扩展性不强。
   2. **基本面向对象方案**：将系统划分为输入、移位、排序和输出四个基本功能类，通过Kwic类对它们进行整合。这种方案具有更好的灵活性和可重用性。
   3. **基于事件通知的方案**：系统中的各个功能并不是直接被调用的，而是通过组件通知或广播事件信息来触发。这种方案提高了系统的灵活性和可扩展性。