Design Document for Personal Information Manager (PIM)

## 1. Architecture Description

The architecture pattern chosen for the PIM is Model-View-Controller (MVC). The main reasons this pattern is adopted are as follows:

1. Facilitating the presentation of data in various views: different views of data can be catered to different user needs, such as browsing a list of records or inspecting a specific record in detail.
2. Enhancing the flexibility of the codebase: making the code more adaptable for implementing the requirements outlined in the user stories.
3. Avoiding complexity arise from console I/O: allowing unit tests to purely focusing on the functionality of the model itself.

The model of

Model: Personal Information Records (PIR) which include contacts, events, tasks, and texts, along with an interface named “PIRInterface”.

Controller: The PIM Kernel, responsible for handling the business logic and interactions between the Model and the View.

View: The PIM itself, which presents data to the user and handles user interactions.

2. Major Code Components Structure and Relationships

The development adopts an Object-Oriented Approach, with the following components:

PIM (Personal Information Manager): The main entry point of the application, handling initial user interaction and delegating tasks to the PIM Kernel.

PIM Kernel: Acts as the Controller in the MVC pattern, managing the flow of data between the Model (PIRs) and the View (PIM interface).

PIRInterface: An interface implemented by various PIR types such as Contact, Event, Task, and Text. It defines common functionalities and attributes shared across different PIRs.

Contact: A class representing contact information, implementing the PIRInterface.

Event: Represents event details, also implementing the PIRInterface.

Task: A class for task management, following the PIRInterface.

Text: Handles plain text records, adhering to the PIRInterface.

Utils: A utility class providing supporting functions and tools for the application.

Each component plays a specific role in the system:

The PIM serves as the interface for user interaction.

The PIM Kernel (Controller) manages requests, processes data, and communicates between the Model and View.

PIRInterface and its implementations (Contact, Event, Task, Text) constitute the Model, holding data and business logic.

Utils provides auxiliary functions that assist in various operations across the application.

图形用户界面

描述已自动生成