# 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.

Regarding how each component of the chosen architecture is implemented, it is as follows:

* **Model**: The four different types of Personal Information Record, namely Contacts, Events, Tasks, and Texts, along with an abstract class named “PIRInterface” that facilitate polymorphism and abstraction.
* **Controller**: The PIM Kernel, responsible for updating the data in the Model, handling requests from the View, and supporting the interactions between the Model and the View. It does not perform basic, yet critical operation based on View requestion.
* **View**: The PIM itself, which presents data to the user and receives user inputs. Meanwhile, it provides user the available functions, such as create, search, export, and load, that they can use to manage their PIRs.

2. Major Code Components Structure and Relationships

The development adopts an Object-Oriented Approach, with the following classes categorized by the component they belong in the MVC pattern:

### Model:

图形用户界面

描述已自动生成

* + PIRInterface: This is an abstract class that defines the common attributes and methods across four different types of personal information record (PIR).

It specifies that all PIRs must include the following attributes:

* + - ID: an integer uniquely identifies a PIR in its corresponding storage
    - Data: an array that stores the actual data
    - TITLES: a map that takes the attributes of a PIR as keys and predefined integer as corresponding values. The purpose of the values is for the PIM kernel print function, which will be elaborated in later part.

The methods that PIRInterface specified for all PIRs are the getters for TITLES, data, ID, and the setters for Data and *NextID*, a static integer attribute for each PIR class that is independent to the NextID in others.

* + Contact, Text, Task, Event: These are the concrete classes that extends PIRInterface represents different types of PIRs. Each class has its own *NextID* which indicates the ID of the next instance to be created, in addition to inheriting all the attributes and methods defined in the PIRInterface.

Meanwhile, they provide a concrete implementation for the NextID setter as there is no NextID in the PIRInterface.

Each of the classes have additional attributes specific to their type:

* + - **Contact**: Name, Email, and Phone Number
    - **Text**: Title, Note
    - **Task**: Title, Description, Due Date
    - **Event**: Title, Description, Staring Time, Alarm

Controller:

PIM (Personal Information Manager): The main entry point of the application, handling initial user

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

图形用户界面

描述已自动生成

图形用户界面

描述已自动生成