Introduction

A command-line-based personal information manager (PIM) is needed for an individual's notetaking, contact management, and event scheduling. The given functionality provides a more convenient methodology for our users to administer their routine activities.

Glossary

Terminology	**Definition**
Personal Information Record (PIR)	A standardized record in the project.
PIM File	A file which contains a PIR. It has the extension ".pim" which can be only read and write by the software.
Note	A subtype of PIR. The classification code is "N".
Task	A subtype of PIR. The classification code is "T".
Event	A subtype of PIR. The classification code is "E".
Contact	A subtype of PIR. The classification code is "C".

User requirements definition

Requirement engineering is performed based on the given user stories. The following concludes the desired services:

1. The system shall store 4 types of personal information records (PIRs), including:
    a. Notes
    b. Tasks
    c. Events
    d. Contacts

2. The input value for each note creation shall include plain text only.

3. The input value for each task creation shall include a description, and a deadline.

4. The input value for each event creation shall include a description, a starting time, and an alarm time.

5. The input value for each contact creation shall include a name, an address, and a phone number.

6. The system shall support modification to an existing PIR.

7. The system shall search for a note by checking whether the content contains a string. (Negation on each criteria is possible.)

8. The system shall search for a task by checking whether the description contains a string, or/and whether the deadline is before, after, or equal to another given point in time. (Negation on each criteria is possible.)

9. The system shall search for an event by checking whether the description contains a string, or/and whether the start time or/and alarm is before, after, or equal to another given point in time. (Negation on each criteria is possible.)

10. The system shall search for a contact by checking whether the name, address, or/and a mobile number contains a string. (Negation on each criteria is

possible.)

11. The system shall print out the information for one specify PIR or all PIRs recorded.

12. The system shall support deletion on a specified PIR.

13. The system shall write PIRs, with a customized file extension name ".pim", to file repository

14. The system shall read files with the extension name ".pim" from the file repository

System architecture

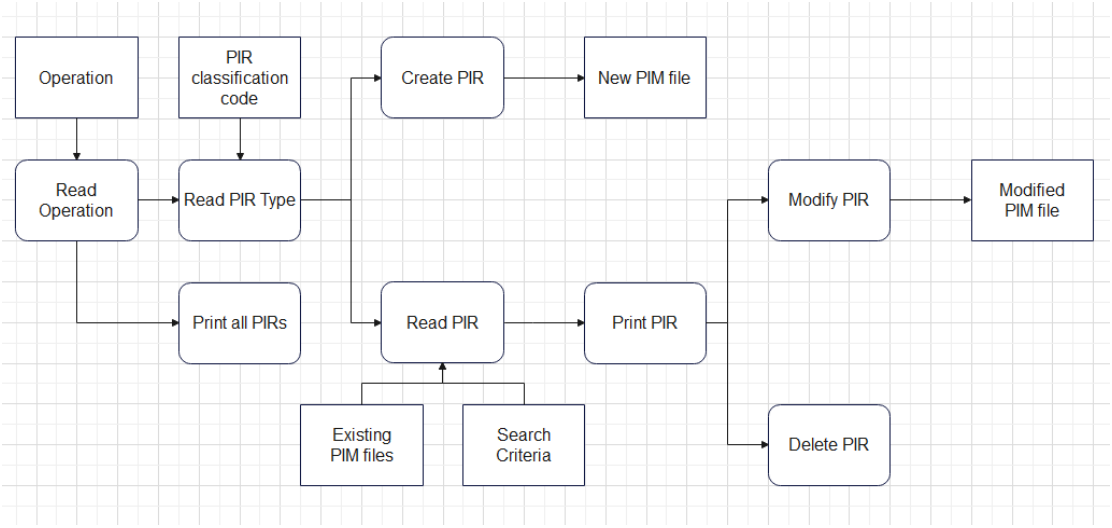

Diagram 1: Pipe and Filter Architecture Design of PIM

The pipe-and-filter architecture is chosen as the major design of PIM. Choosing the architecture consists of few factors:

- Commonly used within a transaction-based data processing software, like the PIM.
- Easy to understand among the developers.
- Requested user interactions are limited.

System Requirement Specifications

1. Create Note

Function:	createNote()
Description:	Creates a new note.
Input:	Plain text for the note content.
Source:	User input.
Destination:	ArrForNote ArrayList.
Action:	Adds a new note to ArrForNote.
Requires:	A string of plain text.
Precondition:	The input is a string of plain text.
Postcondition:	A new note is added to ArrForNote.
Side Effect:	ArrForNote is modified.

2. Create Task

Function:	createTask()
Description:	Creates a new task.
Input:	A description and a deadline for the task.
Source:	User input.
Destination:	ArrForTask ArrayList.
Action:	Adds a new task to ArrForTask.
Requires:	A string for the description and a date for the deadline.
Precondition:	The input includes a string for the description and a valid date for the deadline.
Postcondition:	A new task is added to ArrForTask.
Side Effect:	ArrForTask is modified.

3. Create Event

Function:	createEvent()
Description:	Creates a new event.
Input:	A description, a starting time, and an alarm time for the event.
Source:	User input.
Destination:	ArrForEvent ArrayList.
Action:	Adds a new event to ArrForEvent.
Requires:	A string for the description, and dates for the starting time and alarm time.
Precondition:	The input includes a string for the description and valid dates for the starting time and alarm time.

Postcondition:	A new event is added to ArrForEvent.
Side Effect:	ArrForEvent is modified.

4. Create Contact

Function:	createContact()
Description:	Creates a new contact.
Input:	A name, an address, and a phone number for the contact.
Source:	User input.
Destination:	ArrForContact ArrayList.
Action:	Adds a new contact to ArrForContact.
Requires:	Strings for the name, address, and phone number.
Precondition:	The input includes strings for the name, address, and phone number.
Postcondition:	A new contact is added to ArrForContact.
Side Effect:	ArrForContact is modified.

5. Search Interface

Function:	searchInterface()
Description:	Provides an interface for searching PIRs.
Input:	User's choice of PIR type to search and search criteria.
Source:	User input.
Destination:	SearchService methods.
Action:	Calls the appropriate SearchService method based on the user's choice.
Requires:	A valid choice of PIR type and valid search criteria.
Precondition:	The user provides a valid choice of PIR type and valid search criteria.
Postcondition:	The appropriate SearchService method is called and the search results are processed.
Side Effect:	None.

6. Print All Interface

Function:	printAllInterface()
Description:	Prints out the information for all recorded PIRs into a .pim file.
Input:	Filename for the output file.
Source:	User input.
Destination:	Output file in the file repository.
Action:	Writes all PIRs to the output file.

Requires:	A valid filename.
Precondition:	The user provides a valid filename.
Postcondition:	All PIRs are written to the output file.
Side Effect:	A new file is created in the file repository.

7. Make Directory

Function:	makeDir()
Description:	Creates a directory for the file repository if it doesn't already exist.
Input:	None.
Source:	None.
Destination:	File system.
Action:	Creates a new directory.
Requires:	None.
Precondition:	The directory does not already exist.
Postcondition:	The directory is created.
Side Effect:	A new directory is created in the file system.

8. Reset

Function:	reset()
Description:	Clears all PIRs.
Input:	None.
Source:	None.
Destination:	ArrForNote, ArrForTask, ArrForEvent, ArrForContact.
Action:	Clears all PIRs from ArrForNote, ArrForTask, ArrForEvent, ArrForContact.
Requires:	None.
Precondition:	None.
Postcondition:	ArrForNote, ArrForTask, ArrForEvent, ArrForContact are empty.
Side Effect:	ArrForNote, ArrForTask, ArrForEvent, ArrForContact are modified.

9. Search Note

Function:	searchNote()
Description:	Searches for a note by checking whether the content contains a string.
Input:	A string to search for in the note content.
Source:	User input.
Destination:	ArrForNote ArrayList.

Action:	Searches ArrForNote for notes whose content contains the input string.
Requires:	A string to search for.
Precondition:	The input is a string.
Postcondition:	Returns an ArrayList of notes whose content contains the input string.
Side Effect:	None.

10. Search Task

Function:	searchTask()
Description:	Searches for a task by checking whether the description contains a string, or/and whether the deadline is before, after, or equal to another given point in time.
Input:	A string to search for in the task description and/or a date to compare with the task deadline.
Source:	User input.
Destination:	ArrForTask ArrayList.
Action:	Searches ArrForTask for tasks that match the input criteria.
Requires:	A string to search for and/or a date to compare with the task deadline.
Precondition:	The input includes a string and/or a valid date.
Postcondition:	Returns an ArrayList of tasks that match the input criteria.
Side Effect:	None.

11. Search Event

Function:	searchEvent()
Description:	Searches for an event by checking whether the description contains a string, or/and whether the start time or/and alarm is before, after, or equal to another given point in time.
Input:	A string to search for in the event description and/or dates to compare with the event start time and alarm.
Source:	User input.
Destination:	ArrForEvent ArrayList.
Action:	Searches ArrForEvent for events that match the input criteria.
Requires:	A string to search for and/or dates to compare with the event start time and alarm.
Precondition:	The input includes a string and/or valid dates.
Postcondition:	Returns an ArrayList of events that match the input criteria.

Side Effect: None.

12. Search Contact

Function: searchContact()

Description: Searches for a contact by checking whether the name, address, or/and a mobile number contains a string.

Input: A string to search for in the contact name, address, and/or mobile number.

Source: User input.

Destination: ArrForContact ArrayList.

Action: Searches ArrForContact for contacts that match the input criteria.

Requires: A string to search for.

Precondition: The input is a string.

Postcondition: Returns an ArrayList of contacts that match the input criteria.

Side Effect: None.

13. Process Note Search Results

Function: processNoteSearchResults()

Description: Modifies or deletes the results of a note search.

Input: An ArrayList of notes that match the search criteria.

Source: searchNote() method in SearchService class.

Destination: User interface.

Action: Displays the search results to the user and allows the user to modify the selected note.

Requires: An ArrayList of notes.

Precondition: The input is an ArrayList of notes.

Postcondition: The search results are displayed to the user and the selected note is modified as per user's choice.

Side Effect: The selected note in ArrForNote may be modified or deleted.

14. Process Task Search Results

Function: processTaskSearchResults()

Description: Modifies or deletes the results of a task search.

Input: An ArrayList of tasks that match the search criteria.

Source: searchTask() method in SearchService class.

Destination: User interface.

Action: Displays the search results to the user and allows the user to modify the selected task.

Requires: An ArrayList of tasks.
Precondition: The input is an ArrayList of tasks.
Postcondition: The search results are displayed to the user and the selected task is modified as per user's choice.
Side Effect: The selected task in ArrForTask may be modified or deleted.

15. Process Event Search Results
Function: processEventSearchResults()
Description: Modifies or deletes the results of an event search.
Input: An ArrayList of events that match the search criteria.
Source: searchEvent() method in SearchService class.
Destination: User interface.
Action: Displays the search results to the user and allows the user to modify the selected event.
Requires: An ArrayList of events.
Precondition: The input is an ArrayList of events.
Postcondition: The search results are displayed to the user and the selected event is modified as per user's choice.
Side Effect: The selected event in ArrForEvent may be modified or deleted.

16. Process Contact Search Results
Function: processContactSearchResults()
Description: Modifies or deletes the results of a contact search.
Input: An ArrayList of contacts that match the search criteria.
Source: searchContact() method in SearchService class.
Destination: User interface.
Action: Displays the search results to the user and allows the user to modify the selected contact.
Requires: An ArrayList of contacts.
Precondition: The input is an ArrayList of contacts.
Postcondition: The search results are displayed to the user and the selected contact is modified as per user's choice.
Side Effect: The selected contact in ArrForContact may be modified or deleted.

17. Import File
Function: importFile()
Description: Imports PIRs from a .pim file.

Input:	The name of the .pim file.
Source:	User input.
Destination:	ArrForNote, ArrForTask, ArrForEvent, ArrForContact ArrayLists.
Action:	Reads the .pim file and adds the PIRs to the appropriate ArrayLists.
Requires:	The name of a .pim file.
Precondition:	The .pim file exists in the file repository.
Postcondition:	The PIRs from the .pim file are added to the appropriate ArrayLists.
Side Effect:	ArrForNote, ArrForTask, ArrForEvent, ArrForContact are modified.

18. Error Handling

Function:	Various functions.
Description:	Handles errors and exceptions that occur during system operation.
Input:	Errors or exceptions.
Source:	Various functions.
Destination:	System logs or user interface.
Action:	Catches errors or exceptions, logs them, and displays appropriate error messages to the user.
Requires:	Errors or exceptions.
Precondition:	An error or exception occurs.
Postcondition:	The error or exception is handled.
Side Effect:	System operation may be affected.