Design Documentation

Group 7

Implementation Managers:
Samuel Jenkins – 550400
Yan Sun – 581250

Test Managers:
Ceris Land – 638014
Codrin Morhan – 649909

Planning and Quality Manager:
Simon Maling – 631526

Design Manager:
Adewale Odunlami – 703257

Customer Interface Manager:
Lloyd Woodroffe – 635428

January 20, 2012

Contents

Chapter 1

Candidate classes and responsibilities

1.1 AddressBook, Contact

AddressBook
Author: Yan Sun
SuperClass: -
SubClasses:-
Responsibilities: Initialise the address book. Change view between calendar and address book. View all current contacts. Manipulate contact details.
Collaborations: Event DigitalOrganiser

Protocols:
`public Boolean addContact(Contact c)`
`public Boolean editContact(Contact c)`
`public Boolean removeContact(Contact c)`
`public Boolean viewContacts()`

Unit Tests: Add a contact. Edit a contact. Remove a contact.

Contact
Author: Ceris Land
SuperClass:-
SubClasses:-
Responsibilities: Create new contacts. Get contact details. Validate contact details.
Collaborations: AddressBook

Protocols:
`public String setName(String n)`
`public String getName()`
`public Boolean validateName()`
`public String setAddress(String a)`
`public String getAddress()`
`public String setEmail(String e)`
`public String getEmail()`
`public Boolean validateEmail()`
`public String setHomeTel(String tel)`
`public String getHomeTel()`
`public String setFaxNo(String fax)`
`public String getFaxNo()`
`public String setURL(String url)`
`public String getURL()`
`public Boolean validateURL()`
`public String setOther(String o)`
`public String getURL()`
`public Boolean validateAll()`

Unit Tests: Test all valid/invalid inputs. Check the contact details are updated.

1.2 Event

Event
Author: Codrin Morhan
SuperClass: -
SubClasses: Deadline Lecture Meal Anniversary HappyHour Accident SocialEvent Meeting Other
Responsibilities: Provide event template. Set event data. Get event data. Validate event data. Have a generic icon. Be an abstract class.
Collaborations: View DigitalAddressBook

Protocols:
`set<attribute>(<attributeDataType> newValue) return void`
`get<attribute>() return <attributeDataType> attributeValue`
`validateDate() return Boolean`
`validateStartingTime() return Boolean`
`validateEndingTime() return Boolean`
`validateAll() return Boolean`

Unit Tests: Date is inside the required interval. Event must have a duration. Check validation works.

1.3 Deadlines

Deadline
Author: Adewale Odunlami
SuperClass: Event
SubClasses: WorkDeadline AssignmentDeadline MarkingDeadline BillPlayment
Responsibilities: Same as *Event* Have a specific icon.
Collaborations: Same as *Event*
Protocols: Same as *Event* except:
Unit Tests: Same as *Event* except: Starting and ending time are the same.

WorkDeadline
Author: Adewale Odunlami
SuperClass: Deadline
SubClasses: -
Responsibilities: Same as *Deadline* Have a specific icon.
Collaborations: Same as *Deadline*
Protocols: Same as *Deadline*
Unit Tests: Same as *Deadline*

MarkingDeadline
Author: Adewale Odunlami
SuperClass: Deadline
SubClasses: -
Responsibilities: Same as *Deadline* Have a specific icon.
Collaborations: Same as *Deadline*
Protocols: Same as *Deadline*
Unit Tests: Same as *Deadline*

AssignmentDeadline
Author: Adewale Odunlami
SuperClass: Deadline
SubClasses: -
Responsibilities: Same as *Deadline* Have a specific icon.
Collaborations: Same as *Deadline*
Protocols: Same as *Deadline*
Unit Tests: Same as *Deadline*

BillPayment
Author: Adewale Odunlami
SuperClass: Deadline
SubClasses: -
Responsibilities: Same as *Deadline* Have a specific icon.
Collaborations: Same as *Deadline*
Protocols: Same as *Deadline*
Unit Tests: Same as *Deadline*

1.4 Lecture, HappyHour, Anniversary

Lecture
Author: Adewale Odunlami
SuperClass: Event
SubClasses: -
Responsibilities: Same as *Event* Have a specific icon.
Collaborations: Same as *Event*
Protocols: Same as *Event* except:
Unit Tests: Same as *Event* except: Duration is longer than fifty minutes. Duration is shorter than three hours.

HappyHour
Author: Adewale Odunlami
SuperClass: Event
SubClasses: -
Responsibilities: Same as *Event* Have a specific icon.
Collaborations: Same as *Event*
Protocols: Same as *Event*
Unit Tests: Same as *Event* except: Duration must be exactly one hour.

Anniversary
Author: Codrin Morhan
SuperClass: Event
SubClasses: Birthday BankHoliday
Responsibilities: Same as *Event* Have a specific icon.
Collaborations: Same as *Event*
Protocols: Same as *Event*
Unit Tests: Same as *Event* except: Duration must be exactly one day.

1.5 Birthday, BankHoliday, Accident, SocialEvent, Concert

Birthday
Author: Codrin Morhan
SuperClass: Anniversary
SubClasses: -
Responsibilities: Same as *Anniversary* Have a specific icon.
Collaborations: View
Protocols: Same as *Anniversary*
Unit Tests: Same as *Anniversary* except: Duration is less than a day.

BankHoliday
Author: Codrin Morhan
SuperClass: Anniversary
SubClasses: -
Responsibilities: Same as *Anniversary* Have a specific icon.
Collaborations: Same as *Anniversary*
Protocols: Same as *Anniversary*
Unit Tests: Same as *Anniversary*

Accident
Author: Adewale Odunlami
SuperClass: Event
SubClasses: -
Responsibilities: Same as *Event* Have a specific icon.
Collaborations: View
Protocols: Same as *Event*.
Unit Tests: Same as *Event* except: Starting and ending time are the same.

SocialEvent
Author: Codrin Morhan
SuperClass: Event
SubClasses: Concert
Responsibilities: Same as *Event* Have a specific icon.
Collaborations: Same as *Event*
Protocols: Same as *Event*
Unit Tests: Same as *Event* except: Duration is less than ten days.

Concert
Author: Codrin Morhan
SuperClass: SocialEvent
SubClasses: -
Responsibilities: Same as *SocialEvent* Have a specific icon.
Collaborations: Same as *SocialEvent*
Protocols: Same as *SocialEvent*
Unit Tests: Same as *SocialEvent* except: Duration is less than a day.

1.6 Meeting, Appointment, Meal, Other, EventList

Meeting
Author: Codrin Morhan
SuperClass: Event
SubClasses: Appointment
Responsibilities: Same as *Event* Have a specific icon.
Collaborations: Same as *Event*
Protocols: Same as *Event*
Unit Tests: Same as *Event* except: Duration is less than a day.

Appointment
Author: Codrin Morhan
SuperClass: Meeting
SubClasses: -
Responsibilities: Same as *Meeting* Have a specific icon.
Collaborations: Same as *Meeting*
Protocols: Same as *Meeting*
Unit Tests: Same as *Meeting*

Other
Author: Codrin Morhan
SuperClass: Event
SubClasses: -
Responsibilities: Same as *Event* Have a generic icon.
Collaborations: View
Protocols: Same as *Event*
Unit Tests: Same as *Event*

Meal
Author: Adewale Odunlami
SuperClass: Event
SubClasses: -
Responsibilities: Same as *Event* Have a specific icon.
Collaborations: Same as *Event*
Protocols: Same as *Event*
Unit Tests: Same as *Event* except: Duration is at most one hour.

EventList
Author: Codrin Morhan
SuperClass:-
SubClasses:-
Responsibilities: Be a list of events. Provide functionality for manipulating a list events.
Collaborations: Event View

Protocols:
`public void addEvent(Event e)`
`public void editEvent(Event e)`
`public void removeEvent(Event e)`
`public vodi removeAllEvents()`
`public void getAllEvents()`
`public void enumerateEvents()`

Unit Tests: Add an event. Edit an event. Remove an event. Remove all events. Enumerate the events.

1.7 Task

Task
Author: Simon Maling
SuperClass: -
SubClasses: -
Responsibilities: Creates itself. Changes itself. Validates itself. Sets itself to complete.
Collaborations: Time Date

Protocols:
`public Boolean setTask()`
`public Task getTask()`
`public Boolean validateTask()`
`public Boolean deleteTask()`
`public Boolean completeTask()`
`public Boolean unCompleteTask()`
`public ArrayList<Task> getIncompleteTasks()`
`public ArrayList<Task> getImportantTasks()`
`public ArrayList<Task> getAllTasks()`
`public ArrayList<Task> orderTasksDeadlineAscending()`
`public ArrayList<Task> orderTasksDeadlineDescending()`
`public ArrayList<Task> orderTasksName()`
`public ArrayList<Task> orderTasksCategory(String cat)`

Unit Tests: Create a task. Complete a task. Uncomplete a task.

1.8 TaskList

TaskList
Author: Simon Maling
SuperClass:-
SubClasses:-
Responsibilities: Create a list of tasks. Show itself. Update itself. Manipulate a list of tasks.
Collaborations: Time Date

Protocols:
`public Boolean setTaskList(TaskList tl)`
`public Boolean getTaskList()`
`public voiid showTaskList()`
`public ArrayList<Task> getIncompleteTasks()`
`public ArrayList<Task> getImportantTasks()`
`public ArrayList<Task> getAllTasks()`
`public ArrayList<Task> orderTasksDeadlineAscending()`
`public ArrayList<Task> orderTasksDeadlineDescending()`
`public ArrayList<Task> orderTasksName()`
`public ArrayList<Task> orderTasksCategory(String cat)`

Unit Tests: Add tasks to the list. Remove tasks from the list. Order the list.

1.9 View, Calendar

View
Author: Samuel Jenkins
SuperClass:-
SubClasses: MonthView WeekView DayView Multi-WeekView YearlyView MovingView
Responsibilities: Providing the correct view. Providing valid data. Handle user input. Set changes made to the data by the user. Be an abstract class.
Collaborations: Display

Protocols:
`public View createDisplay()`
`public View createDisplay(Object[] oObject)`
`public Boolean show()`
`public Boolean show(Boolean bShow)`

Unit Tests: Check to see that the View can display itself. Check to see that the View can hide itself.

Calendar
Author: Codrin Morhan
SuperClass:-
SubClasses:-
Responsibilities: Display calendar
Collaborations: Event View
Protocols: `public void display()`
Unit Tests: See if it displays itself correctly.

1.10 MonthView, WeekView

MonthView
Author: Lloyd Woodroffe
SuperClass: MovingViews
SubClasses:-
Responsibilities: Creating a correctly formatted monthly view. Showing events during a month in the monthly view. Editing the event data in the monthly view.
Collaborations: Event Period

Protocols:
`public View MonthlyView()`
`public View MonthlyView (Event[] oEvent)`
`public Boolean show()`
`public Boolean show(Boolean bShow)`
`public Boolean addEvent(Event Period oPeriod)`
`public Boolean addEvent(EventPeriod oPeriod, Event oEvent)`
`public Boolean editEvent(EventPeriod oPeriod, Event oEvent)`
`public Boolean removeEvent(EventPeriod oPeriod, Event oEvent)`

Unit Tests: Check that an event is added to an Event Period in the monthly view. Check that a specific event is added to an Event Period in the monthly view. Check that a specific event is edited in a particular Event Period in the monthly view. Check that a specific event is removed from a particular Event Period in the monthly view.

WeekView
Author: Lloyd Woodroffe
SuperClass: MovingViews
SubClasses:-
Responsibilities: Creating a correctly formatted weekly view. Showing event data into a weekly view. Editing the event data in weekly view.
Collaborations:-

Protocols:
`public View WeeklyView()`
`public View WeeklyView (Event[] oEvent)`
`public Boolean show()`
`public Boolean show(Boolean bShow)`
`public Boolean addEvent(Event Period oPeriod)`
`public Boolean addEvent(EventPeriod oPeriod, Event oEvent)`
`public Boolean editEvent(EventPeriod oPeriod, Event oEvent)`
`public Boolean removeEvent(EventPeriod oPeriod, Event oEvent)`

Unit Tests: Check that an event is added to a Event Period in the week view. Check that a specific event is added to a Event Period in the week view. Check that a specific event is edited in a particular Event Period in the week view. Check that a specific event is removed from a particular Event Period in the week view.

1.11 DayView, Multi-WeekView

DayView
Author: Lloyd Woodroffe
SuperClass: MovingViews
SubClasses:-
Responsibilities: Creating a correctly formatted daily view. Showing event data into a daily view. Editing the event data in daily view.
Collaborations:-

Protocols:
`public View DailyView()`
`public View DailyView (Event[] oEvent)`
`public Boolean show()`
`public Boolean show(Boolean bShow)`
`public Boolean addEvent(Event Period oPeriod)`
`public Boolean addEvent(EventPeriod oPeriod, Event oEvent)`
`public Boolean editEvent(EventPeriod oPeriod, Event oEvent)`
`public Boolean removeEvent(EventPeriod oPeriod, Event oEvent)`

Unit Tests: Check that an event is added to a Event Period in the daily view. Check that a specific event is added to a Event Period in the daily view. Check that a specific event is edited in a particular Event Period in the daily view. Check that a specific event is removed from a particular Event Period in the daily view.

Multi-WeekView
Author: Lloyd Woodroffe
SuperClass: MovingViews
SubClasses:-
Responsibilities: Creating a correctly formatted multi-week view. Showing event data into a multi-week view. Editing the event data in multi-week view.
Collaborations:-

Protocols:
`public View MultiWeekView()`
`public View MultiWeekView (Event[] oEvent)`
`public Boolean show()`
`public Boolean show(Boolean bShow)`
`public Boolean addEvent(Event Period oPeriod)`
`public Boolean addEvent(EventPeriod oPeriod, Event oEvent)`
`public Boolean editEvent(EventPeriod oPeriod, Event oEvent)`
`public Boolean removeEvent(EventPeriod oPeriod, Event oEvent)`

Unit Tests: Check that an event is added to a Event Period in the multi-week view. Check that a specific event is added to a Event Period in the multi-week view. Check that a specific event is edited in a particular Event Period in the multi-week view. Check that a specific event is removed from a particular Event Period in the multi-week view.

1.12 YearlyView

YearlyView
Author: Samuel Jenkins
SuperClass: MovingViews
SubClasses:-
Responsibilities: Creating a correctly formatted yearly view. Showing event data into a yearly view. Editing the event data in yearly view.
Collaborations:-

Protocols:
`public View YearlyView()`
`public View YearlyView (Event[] oEvent)`
`public Boolean show()`
`public Boolean show(Boolean bShow)`
`public Boolean addEvent(Event Period oPeriod)`
`public Boolean addEvent(EventPeriod oPeriod, Event oEvent)`
`public Boolean editEvent(EventPeriod oPeriod, Event oEvent)`
`public Boolean removeEvent(EventPeriod oPeriod, Event oEvent)`

Unit Tests: Check that an event is added to a Event Period in the yearly view. Check that a specific event is added to a Event Period in the yearly view. Check that a specific event is edited in a particular Event Period in the yearly view. Check that a specific event is removed from a particular Event Period in the yearly view.

1.13 MovingView, EventData

MovingView
Author: Samuel Jenkins
SuperClass:-
SubClasses: DayView WeeklyView Multi-WeekView MonthlyView YearlyView
Responsibilities: Update view regularly.
Collaborations:-
Protocols: `public Boolean updateDisplay()`
Unit Tests: Check that the view is updated without error at timely intervals.

EventData
Author: Samuel Jenkins
SuperClass: Views
SubClasses:-
Responsibilities: Get the data to be held from the Events.
Collaborations: DayView WeeklyView Multi-WeekView MonthlyView YearlyView Period
Protocols: `public EventPeriod[] getAllEventData(Period period)`
Unit Tests: Check that all events for a view are added to that period.

1.14 Date

Date
Author: Ceris Land
SuperClass: -
SubClasses:-
Responsibilities: Get date of system clock. Set the current date. Display current date. Compare two dates. Compare date to today.
Collaborations: DigitalOrganizer Calendar Event EventList AddressBook Task

Protocols:
`public DateFormat getDate()`
`public Boolean setDate()`
`public Boolean displayDate()`
`public Boolean isDateAfter(DateFormat d)`
`public Boolean isDateAfter(DateFormat d1, DateFormat d2)`
`public Boolean isDateBetween(DateFormat d1, DateFormat d2)`

Unit Tests: Test if date is got correctly. Test if date is after or between given dates.

1.15 DigitalOrganiser

DigitalOrganiser
Author: Yan Sun
SuperClass:-
SubClasses:-
Responsibilities: Control the main flow of execution of the application. Controls user input. Initialise the application. Initialise all other necessary resources. Exit cleanly; save data upon exit. Get and store who the current user is.
Collaborations: UserInterface User Date Time Store

Protocols:
`public Boolean Initialise()`
`public DateFormat getTime()`
`public DateFormat getDate()`
`public Boolean setCurrentUser(User u)`
`public User getCurrentUser()`
`public Boolean initialiseCalendar(User u)`
`public Boolean initialiseEvent(User u)`
`public Boolean initialiseAddressBook(User u)`
`public Boolean initialiseAll(User u)`
`public Boolean initialiseTask(User u)`
`public Boolean exit(Boolean b)`
`public Boolean exit()`

Unit Tests: Possible only when all the subsystems are available.

1.16 Time

Time
Author: Ceris Land
SuperClass:-
SubClasses:-
Responsibilities: Get time. Set time. Display time. Compare time. Compare time to current time. Determine if time is between two times.
Collaborations: DigitalOrganiser UserInterface Event EventList Task

Protocols:
`public DateFormat getTime()`
`public Boolean setTime()`
`public Boolean displayTime()`
`public Boolean isTimeAfter(DateFormat d)`
`public Boolean isTimeAfter(DateFormat d1, DateFormat d2)`
`public Boolean isTimeBetween(DateFormat d1, DateFormat d2)`

Unit Tests: See if time is got correctly. Test if a time is after/between.

1.17 Store, File

Store
Author: Simon Maling
SuperClass: -
SubClasses: File
Responsibilities: Save data. Backup data. Restore data. Get a saved file. Be an abstract class.
Collaborations: DigitalOrganiser User

Protocols:
`public int getStoreType()`
`public Boolean setStoreType(String s)`
`public String getStoreLocation()`
`public Boolean setStoreLocation(String s)`
`public Boolean save()`
`public Boolean backup()`
`public Boolean restore()`
`public Boolean load()`

Unit Tests: Save a file. Exit program and check file exists. Try backupping.

File
Author: Simon Maling
SuperClass: Store
SubClasses:-
Responsibilities: Decide which file to load. Create a new file. Write to a file. Save file. Delete file. Restore backup file.
Collaborations: DigitalOrganiser User

Protocols:
`public Boolean newFile(String s)`
`public Boolean inputFile()`
`public Boolean writeFile()`
`public Boolean saveFile()`
`public Boolean appendFile()`
`public Boolean deleteFile()`
`public Boolean restoreFile()`

Unit Tests: Test all the methods.

1.18 UserInterface, User

UserInterface
Author: Simon Maling
SuperClass:-
SubClasses:-
Responsibilities: Initiates GUI. Updates GUI. Control display of calendar,address book and tasks.
Collaborations: DigitalOrganiser Event AddressBook TaskList

Protocols:
`public void showCalendar()`
`public void showCalendar()`
`public void showTasks()`

Unit Tests: Testing depends on other classes/subsystems.

User
Author: Yan Sun
SuperClass:-
SubClasses:-
Responsibilities: Create a new user. Store user name and password. Delete an user. Record who is the current user. Associate events,calendar and address book with user. Check password.
Collaborations: DigitalApplication

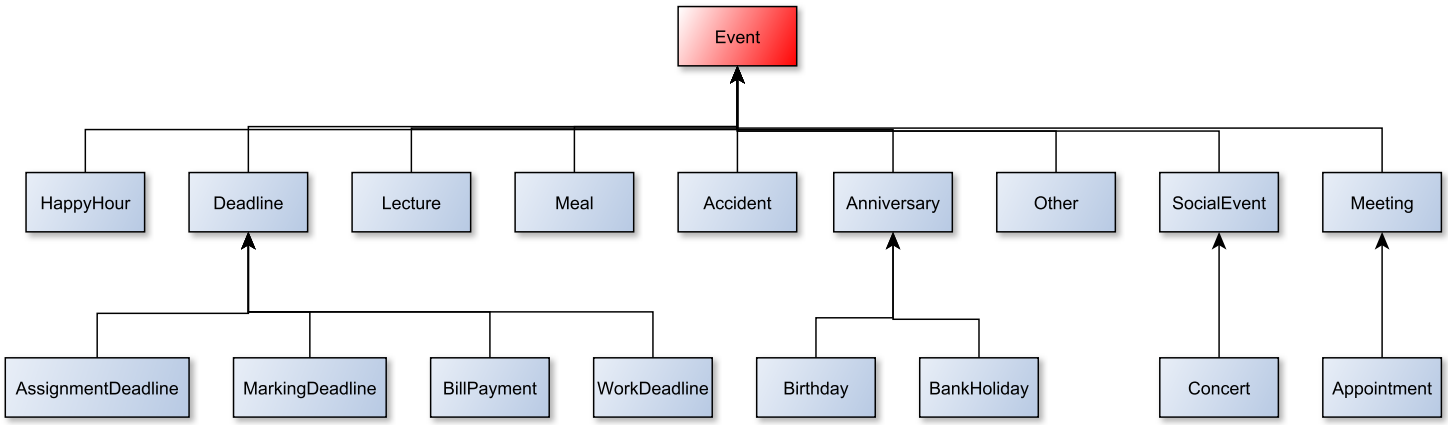
Protocols:
`public User getUser()`
`public Boolean newUser()`
`public Boolean deleteUser()`
`public Boolean setPassword()`
`public Boolean comparePassword(String pass)`

Unit Tests: Create an user. Delete an user. Check user validity.

Chapter 2

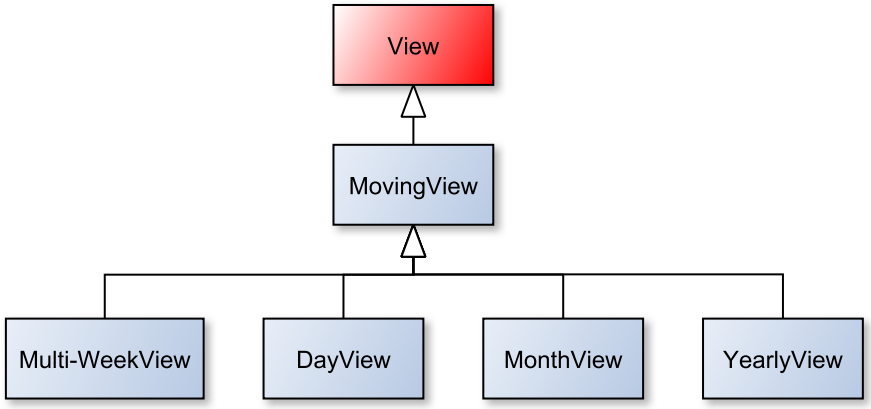
Class Hierarchy Diagrams and Descriptions

2.1 Types of Events



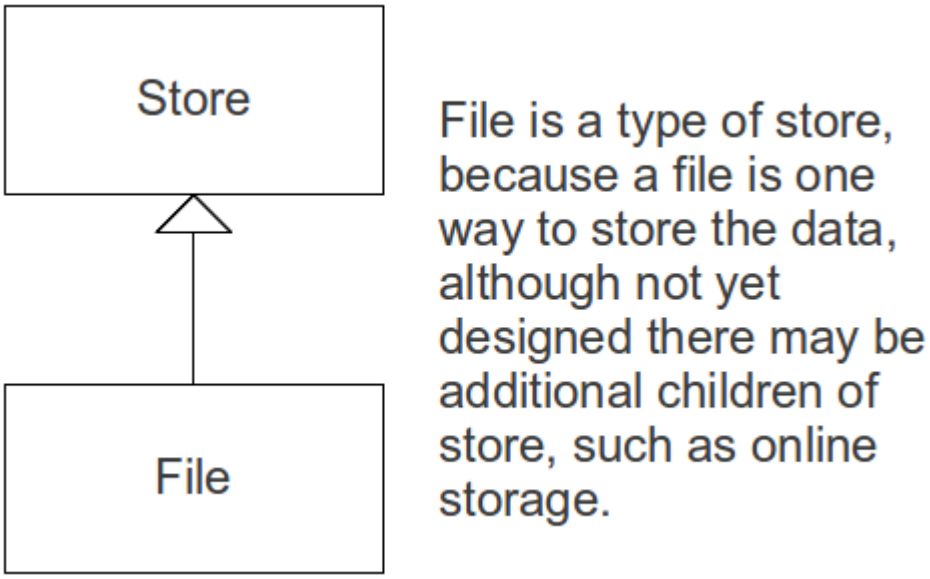
HappyHour/Deadline/Lecture/Meal/Accident/Anniversary/Other/SocialEvent/Meeting *is-kind-of* Event.
AssignmentDeadline/MarkingDeadline/BillPayment/WorkDeadline *is-kind-of* Deadline.
Birthday/BankHoliday *is-kind-of* Anniversary.
Concert *is-kind-of* SocialEvent.
Appointment *is-kind-of* Meeting.

2.2 Types of Views



Moving view *is-kind-of* View.
Multi-Week/Day/Month/Yearly View *is-kind-of* MovingView.

2.3 Types of Storage

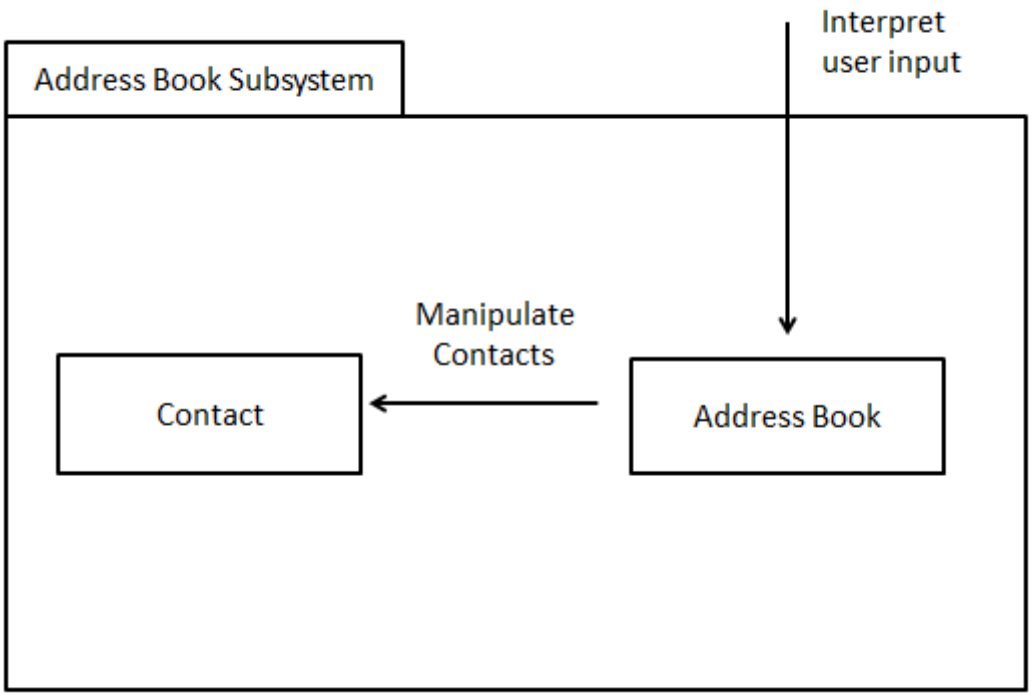


File *is-kind-of* Store.

Chapter 3

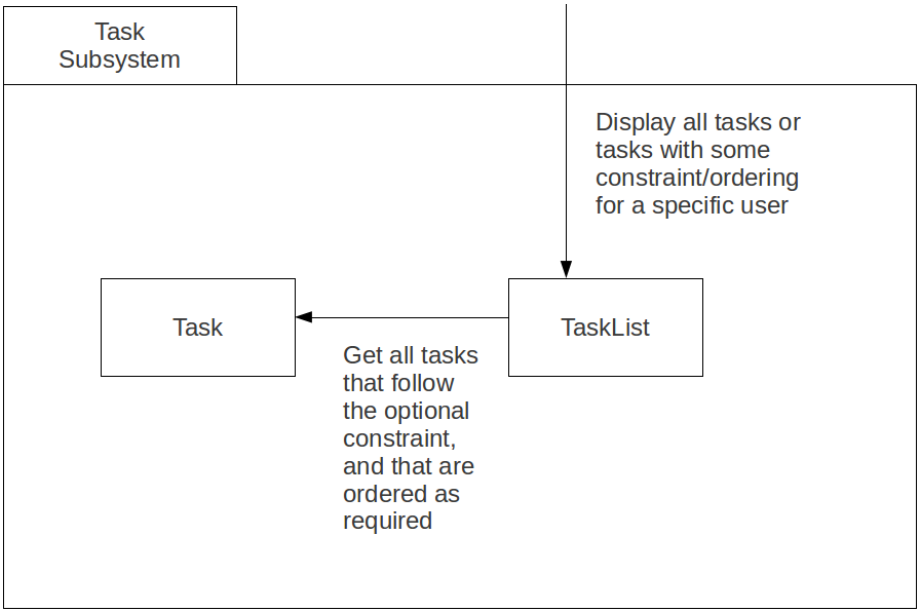
Sub-system Diagrams and Descriptions

3.1 Address Book Sub-system



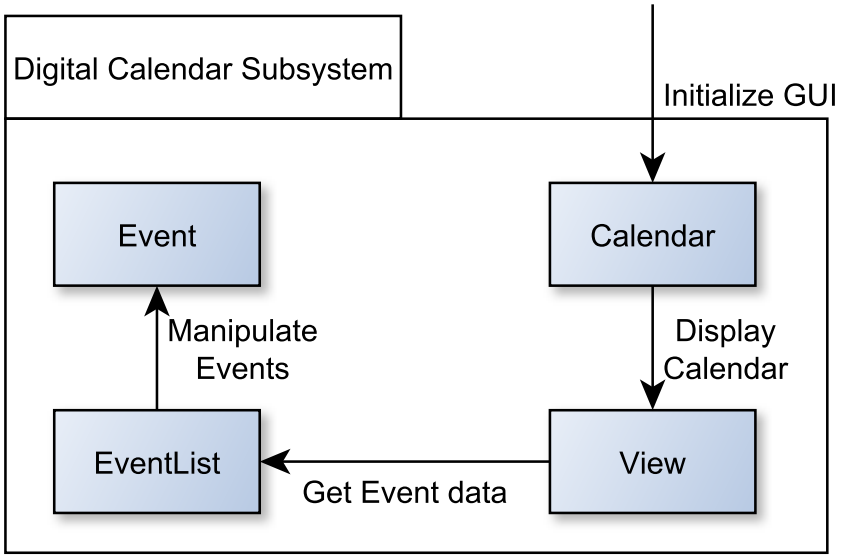
Contact *is-part-of* AddressBook.

3.2 Task Sub-system



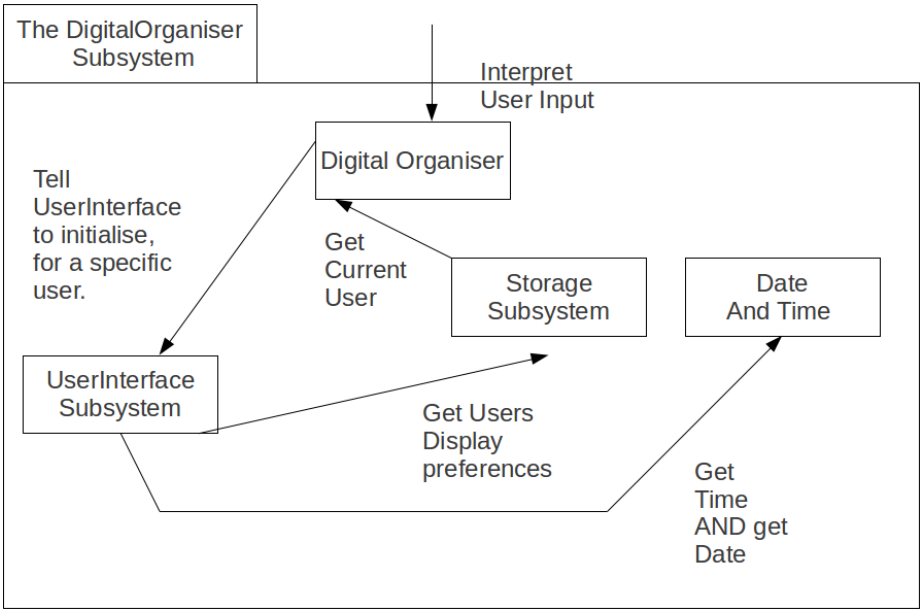
Task *is-part-of* TaskList.

3.3 Digital Calendar Sub-system



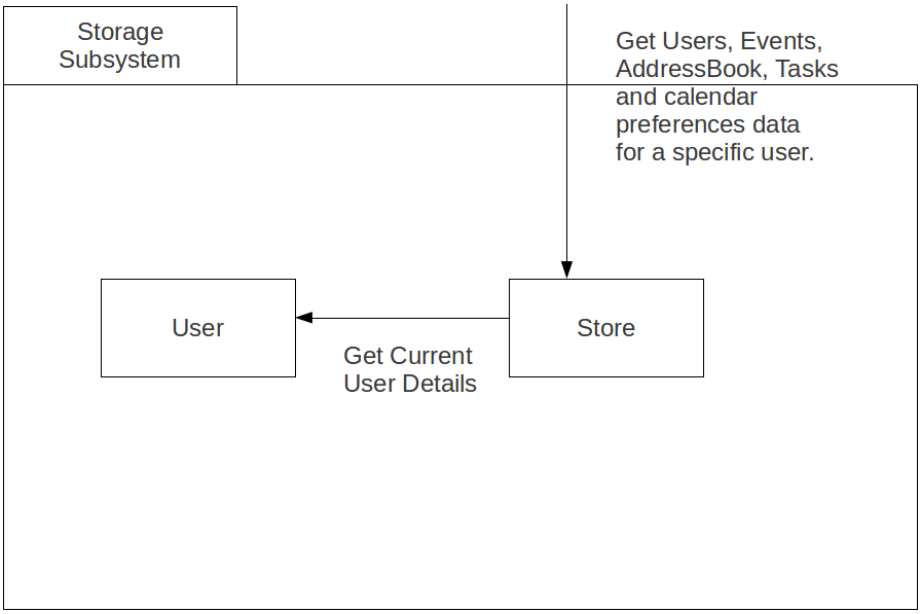
Event *is-part-of* EventList. EventList *is-part-of* Calendar Views. Calendar Views *is-part-of* Calendar.

3.4 Digital Organiser Sub-system



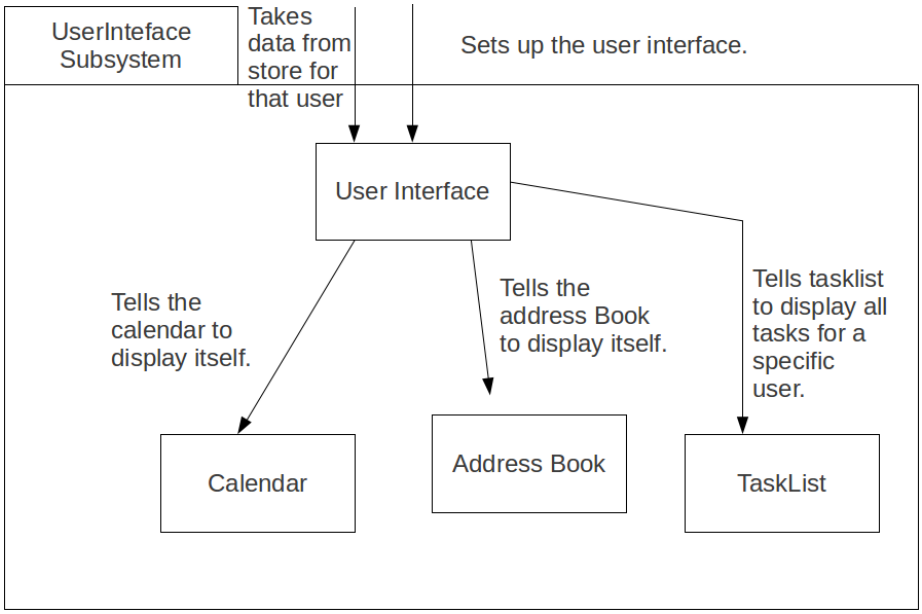
UserInterface *is-part-of* DigitalOrganizer.
The Storage Sub-system *is-part-of* DigitalOrganizer.
The Storage Sub-system *is-part-of* UserInterface.
Date and Time Sub-system *is-part-of* UserInterface.

3.5 Storage Sub-system



User *is-part-of* the Store.

3.6 User Interface Sub-system



TaskList *is-part-of* UserInterface.
AddressBook *is-part-of* UserInterface.
Calendar *is-part-of* UserInterface.

Chapter 4

Data File Format

We will use serialization for our files and we will use this to save objects directly to the files. We will use an ascii text file, the file will be named as either usernameCURRENT or usernameBACKUP[DATE]. The first name for when the file is the current file being used and the second one for when a backup is being made.

The file: Saving to the file will be done by the store class. The file will be saved to by saving the users store object to the file. This will include the date created as a date period object, and the last saved date also as a date period and time period object (this will be replaced by the current date every time the file is saved). The store holds everything for the specific user.

The store object will include: The user object containing the username, and preferences of that user. Backups will be exactly the same except they will have backup and the backup date in the filename. This will mean that there are multiple files (one for each users).