**DoIT AG**
Melvin Lupp
Thaneswaran Jotheeswaraguru

Erstellt: 27.01.2016
Bearbeitet: 22.03.2016
Klasse: INF13a

# *DoIT*

## Tool zur alltäglichen Aufgabenverwaltung

## Objektorientiertes Design (OOD)

Verfasser: Melvin Lupp | Thaneswaran Jotheeswaraguru
Version: 1.0
Status: zur Prüfung
Datum: 22.03.2016

Zugerstrasse 13
6030 Ebikon
Schweiz

Tel. +41 41 445 34 34
Fax +41 41 445 38 86
www.schindler.com

**DoIT AG**
Melvin Lupp
Thaneswaran Jotheeswaraguru

Erstellt: 27.01.2016
Bearbeitet: 22.03.2016
Klasse: INF13a

# Projektmitarbeiter

| Name | Vorname | Email-Adresse |
|---|---|---|
| Lupp | Melvin | melvin.lupp@dava.ch |
| Jotheeswaraguru | Thaneswaran | t.jotheeswaraguru@gmail.com |

# Änderungskontrolle

| Version | Datum | Wer | Bemerkung |
|---|---|---|---|
| 0.1 | 27.01.2016 | | draft |

# Prüfung

| Version | Datum | Wer | Bemerkung | Visum |
|---|---|---|---|---|
| 0.1 | 27.01.2016 | | draft | |

**DoIT AG**
Melvin Lupp
Thaneswaran Jotheeswaraguru

Erstellt: 27.01.2016
Bearbeitet: 22.03.2016
Klasse: INF13a

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1. Zweck des Dokumentes

Im folgenden Dokument handelt es sich um das **Objektorientierte Design (OOD)** zu unserem Projekt **DoIT**.
Das Objektorientierte Design basiert auf den unter **1.2.** aufgeführten Referenzen.

## 1.2. Referenzierte Dokumente

| Nr. | Datum | Name | Link |
|---|---|---|---|
| **[1]** | 18.09.2015 | Projektantrag | Siehe Mail |
| **[2]** | 17.11.2015 | Anforderungsspezifikation | Siehe Mail |
| **[3]** | 26.01.2016 | Objektorientierte Analyse | Siehe Mail |

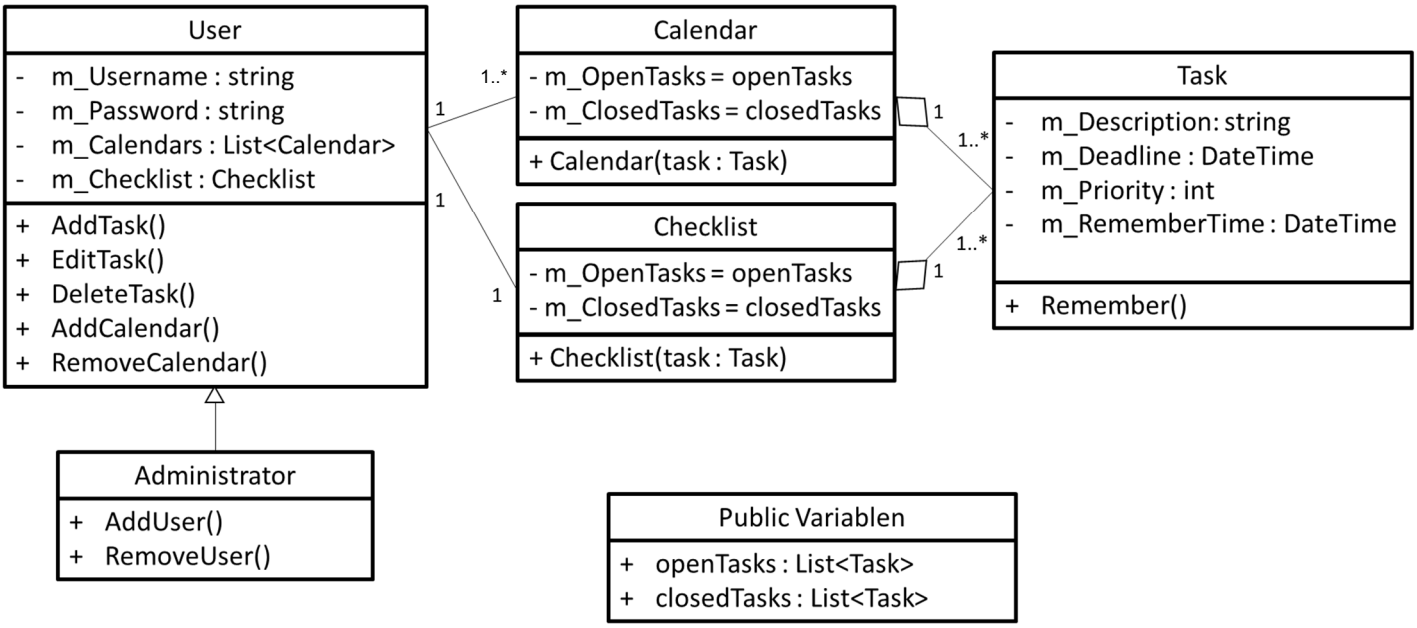
# 2 Objektorientiertes Design

## 2.1. Problembeschreibung

Bisher haben wir folgendes Problem erkannt:

- Wir sind nicht ganz sicher, ob das UML Klassendiagramm im Dokument OOA zu 100% die schlussendliche Klassenstruktur wiederspiegelt.

## 2.2. UML Klassendiagramm




### 2.2.1. Erklärung der Änderungen zum UML in der OOA

**Neue Membervariablen**

- m_OpenTasks / m_ClosedTasks
  - Um offene und geschlossene Tasks zu unterscheiden haben wir die Variable m_Tasks in zwei neue Variablen unterteilt.
- Public Variablen (openTasks / closedTasks)
  - Damit wir nicht genau dieselben Variablen bei den Klassen Calendar und Checklist deklarieren müssen, speichern wir die offenen / geschlossenen Tasks in die öffentlichen Variablen openTasks und closedTasks. Die dazugehörigen Membervariablen in den Klassen beinhalten dann diese Werte.
- Task.m_RememberTime
  - Möchte der Benutzer an einen bestimmten Task erinnert werden, kann er die Uhrzeit für die Erinnerung setzen. Der Wert liegt standardmässig auf dem Wert der Variable m_Deadline

**Geänderte Membervariablen**

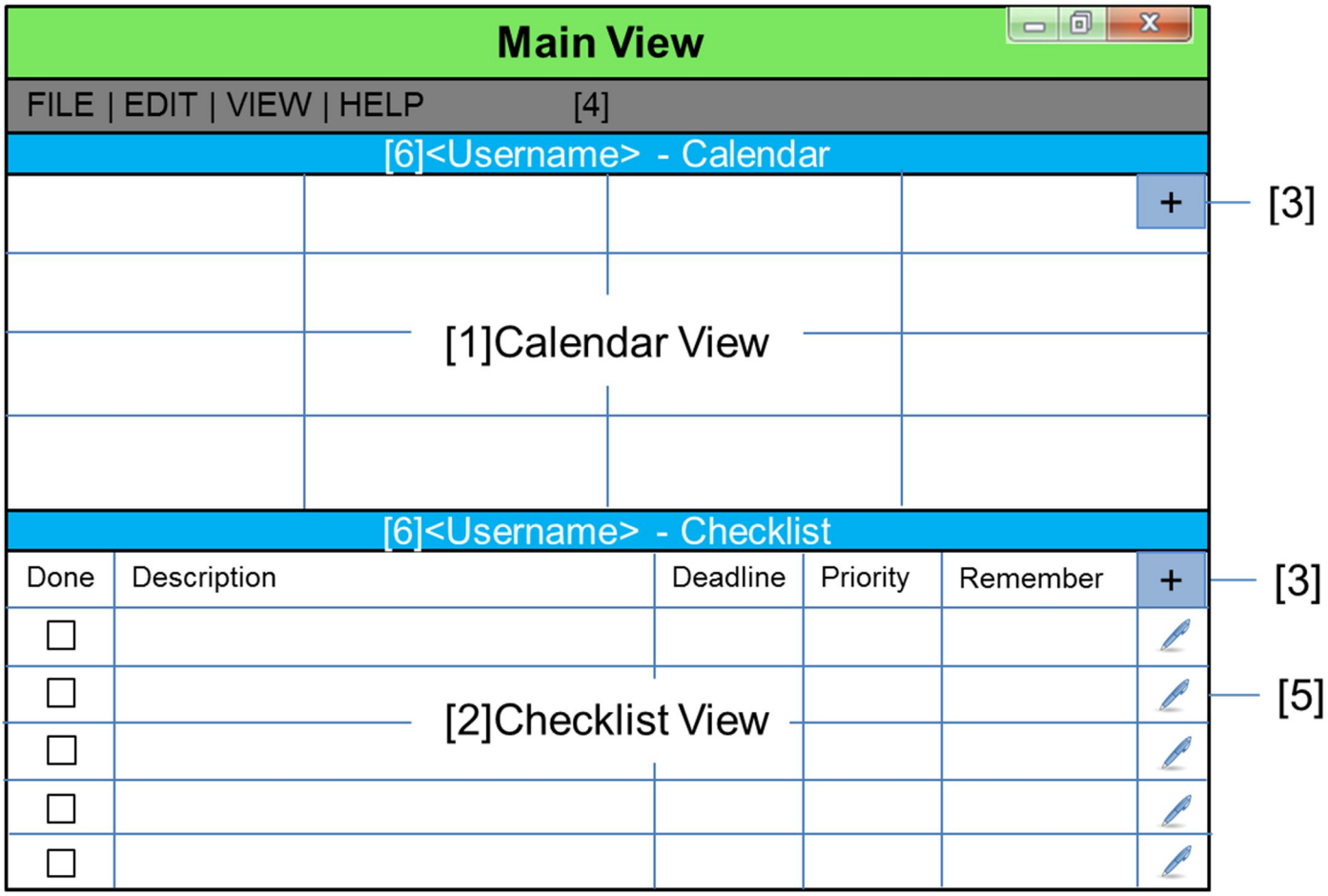
- User.m_Calendar → User.m_Calendars
  - Ein Benutzer kann über mehrere Kalender verfügen, deshalb brauchen wir auch eine Liste statt nur eine Variable vom Typ Calendar.

**Neue Methoden**

- User.AddCalendar()
  - Fügt einen Kalender zur Variable User.m_Calendars hinzu.
- User.RemoveCalendar()
  - Entfernt einen Kalender von der Variable User.m_Calendars.

## 2.3. Gestaltung der Benutzeroberfläche

Die folgende Skizze zeigt einen Prototyp für unser Hauptfenster, in dem Alle wichtigen Funktionen zusammengefasst werden.



### 2.3.1. Menüstruktur

In unserem Hauptmenü haben wir voraussichtlich vier Menükategorien; File, Edit, View und Help.

**FILE**

- Integrate Calendar
- Save Calendar Data
- Create new Calendar
- Delete Calendar…

**EDIT**

- Add new Task
- Edit Task
- Delete Task

**VIEW**

- Maximize Calendar / Checklist
- Hide Calendar / Checklist
- Filter…
- Search…

**HELP**

- Documentation
- Report Bugs
- About

### 2.3.2. Funktionen der einzelnen Controls

**[1] – Kalender-Ansicht**
Die Kalender-Ansicht zeigt alle Tasks strukturiert in einem Kalender inklusive einer detaillierteren Ansicht des einzelnen Tasks wenn man ihn fokussiert.

**[2] – Checklist-Ansicht**
Die Checklist-Ansicht zeigt alle Tasks strukturiert in einer ListView-Control, welche als Checklist umfunktioniert wird. Sie enthält fünf Spalten;

- Done
  - Wenn ein Task abgearbeitet wurde, kann man ihn als erledigt markieren, wodurch er von der „Open Tasks“-Checklist heruntergenommen wird. (Man kann ihn später immer noch ansehen).
- Description
  - Die Beschreibung des Tasks (wird bei der Task-Erfassung angegeben).
- Deadline
  - Die Frist des Tasks (Optional).
- Priority
  - Die Proirität des Tasks (Optional).
- Remember
  - Falls der Benutzer eine Erinnerung einstellt wird diese in dieser Spalte angezeigt.

**[3] – <Add>-Button**
Mit dem Add Button wird ein neuer Task erfasst. Diese Funktion ist auch im Menustrip enthalten.

**[4] – Menustrip**
Siehe Punkt **2.3.1**

**[5] – Edit Button**
Mit dem Edit Button kann man existierende Tasks ändern und seinen Bedürfnissen anpassen.

### 2.3.3. Wichtige Dialoge zur Laufzeit

Um eine ToDo-Applikation zu erstellen, braucht es natürlich einige Dialogfenster um wichtige Daten zu erfassen. Folgende Dialoge sind wichtig für das Programm:

| Wird ausgelöst durch | Dialogtyp | Aufgabe |
|---|---|---|
| Neuer Task / Task bearbeiten | Neue Form zur Eingabe | Nimmt Daten für einen neuen Task entgegen (Beschreibung, Frist, Priorität) |
| Task / Kalender löschen | MessageBox | Fragt nach, ob der Task / Kalender wirklich gelöscht werden soll |
| Kalender integrieren | OpenFileDialog | Der Benutzer kann damit sein File suchen, welches Kalenderdaten enthält |
| Kalender speichern | SaveFileDialog | Der Benutzer sucht einen geeigneten Speicherplatz, wo danach eine XML Datei generiert wird, welche alle Kalenderdaten des ausgewählten Kalenders enthält |
| Task.Remember() | MessageBox | Wenn der Benutzer in einem Task die Erinnerungsfunktion aktiviert, erscheint zur angegebenen Uhrzeit eine Erinnerung |
| Administrator.AddUser() | Neue Form zur Eingabe | Nimmt Daten auf und erstellt daraus einen neuen Benutzer |
| Administrator.RemoveUser() | Neue Form mit Benutzerliste zum auswählen | Der ausgewählte Benutzer wird entfernt |

## 2.4. Persistenzmachung der Daten und Benutzereinstellungen

Wir speichern die Daten in eine XML Datei ein.

### 2.4.1. Einspeicherung in eine XML Datei

Die XML Datei hat die folgende Tagstruktur:

```
<Users>
  <User>
    <Username>
    <Password>
    <Calendars>
      <Calendar>
        <OpenTasks>
          <Task>
            <Description>
            <Deadline>
            <Priority>
            <RememberTime>
          </Task>
          …
        </OpenTasks>
        <ClosedTasks>
          <Task>
            <Description>
            <Deadline>
            <Priority>
            <RememberTime>
          </Task>
          …
        </ClosedTasks>
      </Calendar>
      …
    </Calendars>
    <Checklist>
      <OpenTasks>
        <Task>
          <Description>
          <Deadline>
          <Priority>
          <RememberTime>
        </Task>
        …
```

```
        </OpenTasks>
        <ClosedTasks>
          <Task>
              <Description>
              <Deadline>
              <Priority>
              <RememberTime>
           </Task>
          …
        </ClosedTasks>
      </Checklist>
   </User>
   …
</Users>
```

# 3 Glossar

| Begriff | Erklärung / Bezeichnung |
|---|---|
| Persistenz | Persistenz ist die Fähigkeit eines Objektes, über die Ausführungszeit eines Programms zu leben. |
| OOA | Objektorientierte Analyse |
| OOD | Objektorientiertes Design |
| Assoziation | Beziehung zwischen zwei oder mehreren Classifiern |
| Controls | Interaktionselemente zur Benutzersteuerung |
| ERM | Entity Relationship Model |
| UML | Unified Modeling Language |