DoIT AG
Melvin Lupp
Thanes Jotheeswaraguru

Erstellt am:	25.01.2016
Bearbeitet:	26.01.2016
Klasse:	INF13a

DoIT | Objektorientierte Analyse

Tool zur alltäglichen Aufgabenverwaltung

Verfasser:	Melvin Lupp | Thanes Jotheeswaraguru
Version:	1.0
Status:	in Bearbeitung
Datum:	22.11.2015

BBZW
Kottenmatte 4
6210 Sursee
Schweiz

Tel. +41 41 925 13 00
Fax +41 41 925 13 20
beruf.lu.ch

DoIT AG
Melvin Lupp
Thanes Jotheeswaraguru

Erstellt am: 25.01.2016
Bearbeitet: 26.01.2016
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	13.10.2015		draft

Prüfung

Version	Datum	Wer	Bemerkung	Visum
0.1	13.10.2015		draft	

DoIT AG
Melvin Lupp
Thanes Jotheeswaraguru

Erstellt am: 25.01.2016
Bearbeitet: 26.01.2016
Klasse: INF13a

Inhaltsverzeichnis

DoIT AG
Melvin Lupp
Thanes Jotheeswaraguru

Erstellt am: 25.01.2016
Bearbeitet: 26.01.2016
Klasse: INF13a

1. Einleitung

1. Zweck des Dokumentes

Folgende Dokument ist eine OOA, das im Zusammenhang ist mit dem Projekt DoIT "ToDo" steht. OOA steht in diesem Kontext für objektorientierte Analyse. Diese Analyse basiert auf unsere schon vorher geschriebenen Dokumente "Projektantrag" und "Anforderungsspezifikation".

2. Referenzierte Dokumente

Nr.	Datum	Name	Link
[1]	18.09.2015	Projektantrag	Siehe Mail
[2]	17.11.2015	Anforderungsspezifikation	Siehe Mail

2. Objektorientierte Analyse

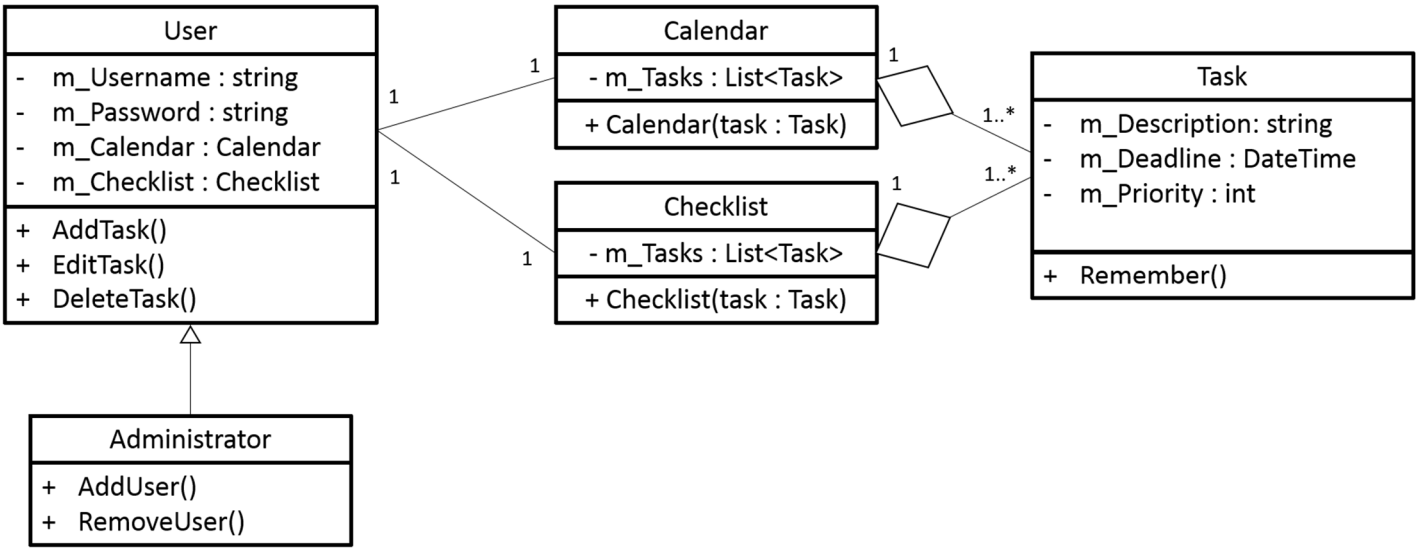
1. Benutzer

Nachdem sich der **User** erfolgreich **angemeldet** hat, kann er alle **grundlegende Funktionen** benutzen z.B. Aufgaben verwalten. Spezifischer beschrieben, kann er **Aufgaben** neu **hinzufügen**, die Vorhandenen **bearbeiten** und erledigte/unnötige Aufgaben **löschen**. Diese werden mit anderen offenen Instanzen des Programm **synchronisiert**.

2. Administrator

Nach der **erfolgreichen Anmeldeverfahren** kann der Admin alle Funktionen zugreifen, was auch der Benutzer hat. Zusätzlich kann er aber noch **User hinzufügen** bzw. **entfernen** von der jeweiligen Checkliste und Kalender. Ebenso kann nur der Admin ein Export/Import-Ablauf durchführen.

3. Klassendiagramm



1. Objektklassen

Wir benutzen für unser Programm 5 Objektklassen;

- User
  - Administrator
- Calendar
- Checklist
- Task

2. Assoziationen

Die Assoziationen sehen wie folgt aus:

- Die Klasse Administrator erbt von der Klasse User, da der Administrator auch ein Benutzer ist, welcher über mehr rechte verfügt
- Ein User hat genau einen Kalender und eine Checklist zugeteilt (1:1 Beziehung)
- Die Klassen Calendar und Checklist sehen ziemlich ähnlich aus, da sie genau die gleichen Sachen darstellen sollen, jedoch auf verschiedene Arten. Ein Kalender / eine Checklist kann mehrere Tasks beinhalten, welche angezeigt werden sollen.

3. Attribute

User
Variablen

- Private string m_Username //Beinhaltet den Benutzernamen
- Private string m_Password //Beinhaltet das Passwort
- Private Calendar m_Calendar //Der Kalender des Benutzers
- Private Checklist m_Checklist //Die Checklist des Benutzers

Methoden

- Public void AddTask() //Fügt einen neuen Task zum Benutzer
- Public void EditTask() //Bearbeitet einen existierenden Task
- Public void DeleteTask() //Löscht einen existierenden Task

Administator
Methoden

- Public void AddUser() //Fügt einen neuen Benutzer hinzu
- Public void RemoveUser() //Löscht einen Benutzer

Calendar
Variablen

- Private List<Task> m_Tasks //Die Tasks, die der Kalender anzeigen soll

Methoden

- Public void Calendar(Task) //Erweiterter Konstruktor

Checklist
Variablen

- Private List<Task> m_Tasks //Die Tasks, die die Checklist anzeigen soll

Methoden

- Public void Checklist(Task) //Erweiterter Konstruktor

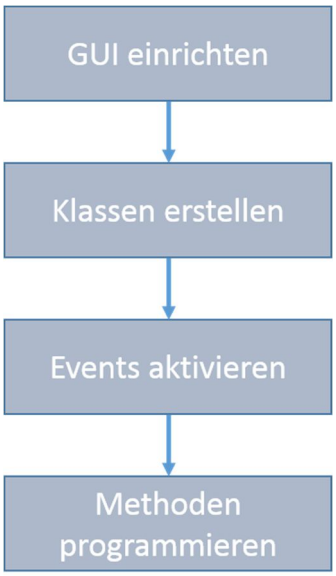
Task
Variablen

- Private string m_Description //Taskbeschreibung
- Private DateTime m_Deadline //Frist des Tasks
- Private int m_priority //Setzt die Priorität des Tasks

Methoden

- Public void Remember() //Erinnerung für den Benutzer

4. Aktivitätsdiagramm



5. Analyse der Persistenzmachung

Variablen, deren Werte lang beständig sein sollen, werden in öffentlichen statischen Variablen gespeichert, damit auf sie jederzeit von überall zugegriffen werden kann.

3. Glossar

Begriff	Erklärung / Bezeichnung
Persistenz	Persistenz ist die Fähigkeit eines Objektes, über die Ausführungszeit eines Programms zu leben.