

HTWG Konstanz

# Ernährungsplaner 2.0

Anwendung der linearen Optimierung

Julien Hespeler 288008  
Dusan Spasic 288238  
24.6.2015



**HOCHSCHULE  
KONSTANZ**  
TECHNIK, WIRTSCHAFT  
UND GESTALTUNG

## 1 Inhaltsverzeichnis

1	Inhaltsverzeichnis.....	0
2	Einleitung .....	2
3	Aufgabenstellung.....	3
4	Tätigkeiten.....	4
4.1	Refactoring des vorhandenen Codes .....	4
4.2	Implementierung Menü „Solverpfade anpassen“ .....	7
4.3	Implementierung Cplex .....	8
4.4	Lebensmittel dauerhaft löschen und hinzufügen.....	8
4.5	Anpassung der Hilfe-Datei.....	9
4.6	Implementierung Fehlerausgabe .....	10
4.7	Erstellung Installationsdatei .....	10
5	Quick-Installation Guide .....	11
6	Abbildungsverzeichnis.....	12

## 2 Einleitung

Im Rahmen der Vorlesung „Anwendung der linearen Optimierung“ bei Prof. Dr. Michael Grütz sind von den Studierenden in Zweier-Gruppen vorhandene Methoden der Methodenbank zu überarbeiten.

Hierbei wird jeder Zweier-Gruppe eine Methode zugeteilt, bei der die Gruppe vorhandene Fehler beheben, sowie zusätzliche Funktionen implementieren soll.

### 3 Aufgabenstellung

Im Sommersemester 15 sollte die Weiterentwicklung der Methode „Ernährungsplaner 2.0“ erneut aufgegriffen werden. Die Methode wurde zuletzt 2008 Weiterentwickelt und war nicht lauffähig. In Rücksprache mit Serkan Önnisan und Prof. Dr. Michael Grütz wurden folgende funktionale und nicht funktionale Anforderungen definiert, die implementiert werden sollten:

- Der Benutzer muss nach Klick auf den Button „berechnen“ das Ergebnis angezeigt bekommen.
- Der Benutzer muss den Solver-Pfad manuell im Programm ändern können.
- Der Benutzer muss Lebensmittel dauerhaft hinzufügen können.
- Der Benutzer muss Lebensmittel dauerhaft löschen können
- Der Benutzer muss zwischen LP-Solve und Cplex als Solver auswählen können
- Im Fenster muss anstelle der Version „1.0“ die Version „2.0“ angezeigt werden

Nach einer ausführlichen Analyse des vorhandenen Codes und einigen Tests der vorhandenen Methode wurde beschlossen, die folgenden weiteren Anforderungen zu definieren:

- Der Code soll wartbar gemacht werden, um zukünftige Änderungen schneller zu implementieren
- Die Hilfe-Datei soll auf den neusten Stand gebracht werden
- Der Benutzer soll fehlerhafte Eingaben als Pop-Up eingeblendet bekommen
- Der Benutzer soll die neue Version durch einen Installer installieren können

Um oben genannte Anforderungen zu implementieren, wurden mehrere Änderungen an der bestehenden Methode durchgeführt, der LP-Ansatz, sowie die Logik der Berechnungen wurden jedoch nicht verändert.

## 4 Tätigkeiten

### 4.1 Refactoring des vorhandenen Codes

Nach erster Sichtung des Codes wurde festgestellt, dass die Methode aus mehreren Java-Klassen besteht, die alle in nur einem Package gespeichert sind. Es wurde außerdem festgestellt, dass die Klassen „ABESFGUI.java“, „SearchElement.java“ und „JPanelMenueAbout.java“ nie aufgerufen werden und für die Ausführung der Methode nicht benötigt werden.

Um die Wartbarkeit des Codes zu erhöhen, wurden zuerst alle Klassen-, Variablen- und Methodennamen auf denselben Standard gebracht und mit englischen Namen versehen. Nicht benötigte Klassen wurden gelöscht. An folgenden Klassen wurden Änderungen durchgeführt:

ABESFGUI	→	gelöscht
DiatplanerException	→	gelöscht
Energiebedarf	→	EnergyNeeds
JPanelAusgabe	→	ResultFrame
JPanelAuswahl	→	SelectFoodFrame
JPanelLebensmittelhinzufuegen	→	AddFoodFrame
JPanelLebensmittelloeschen	→	DeleteFoodFrame
JPanelMenueAbout	→	gelöscht
JPanelWerte	→	StartFrame
mainFrame	→	MainFrame
SearchElement	→	gelöscht
SolverMain	→	gelöscht

Abbildung 1: Änderungen an den Java-Klassen während des Refactorings

Die Klassen wurden zusätzlich nach dem 3-Schichten-Prinzip (Model, View, Controller) in Packages unterteilt, sodass die Methode nun folgende Struktur aufweist:

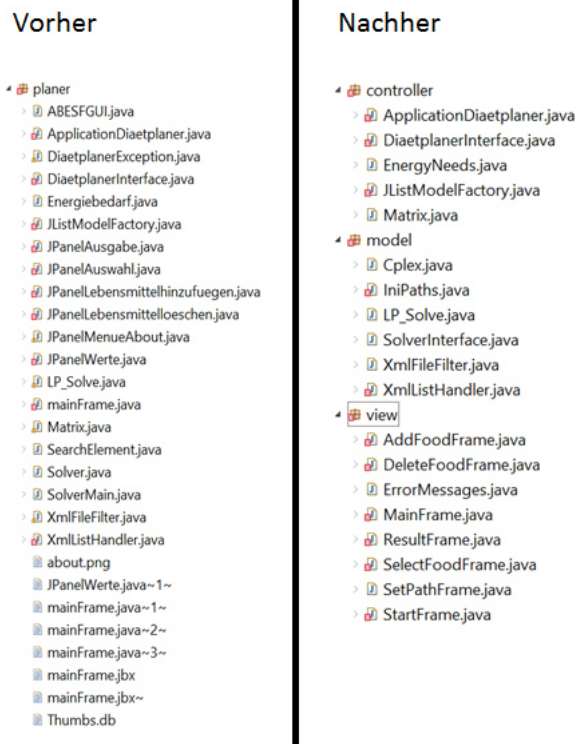


Abbildung 2: Package-Struktur mit Java-Klassen vor und nach dem Refactoring

Zusätzlich wurde in sämtlichen Klassen ein Header eingefügt, der die Funktion der Klasse kurz beschreibt und auch in der neu erzeugten JavaDoc zu finden ist. Somit kann auf den ersten Blick erkannt werden, was die Aufgaben der einzelnen Klassen sind.

```
/**
 * <p>
 * Title: Application Dietplaner
 * </p>
 * <p>
 * Description: Anwendung mit Main-Methode für Programm Diätplaner
 * </p>
 * <p>
 * Copyright: Matthias Siegart Copyright (c) 2003, (Refactoring 2015 by Julien Hespeler, Dusan Spasic)
 * </p>
 * <p>
 * Company: FH Konstanz
 * </p>
 *
 * @author Matthias Siegart, Julien Hespeler, Dusan Spasic
 * @version 2.0
 */
```

Abbildung 3: Header in der Klasse ApplicationDietplaner zum besseren Verständnis

Nach Fertigstellung des Refactorings konnte auch im UML-Klassendiagramm eine deutliche Vereinfachung der Codestruktur erkannt werden.

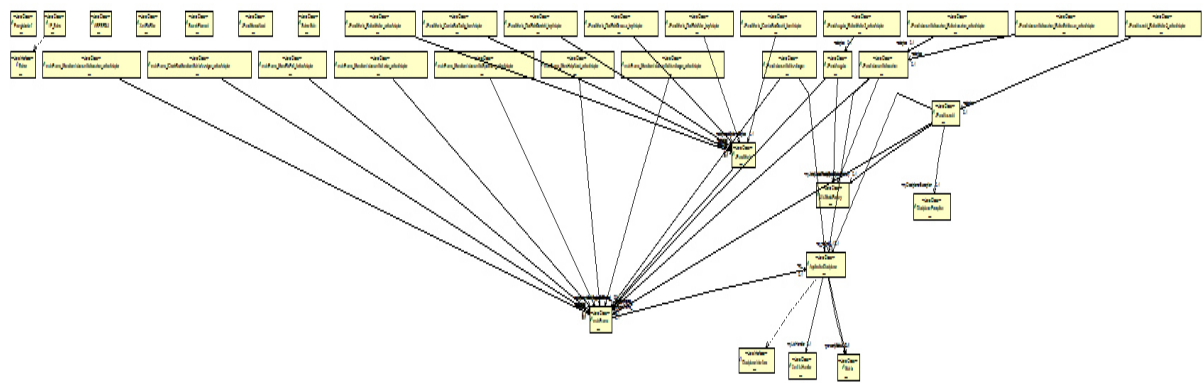


Abbildung 4: UML-Klassendiagramm der Methode vor dem Refactoring

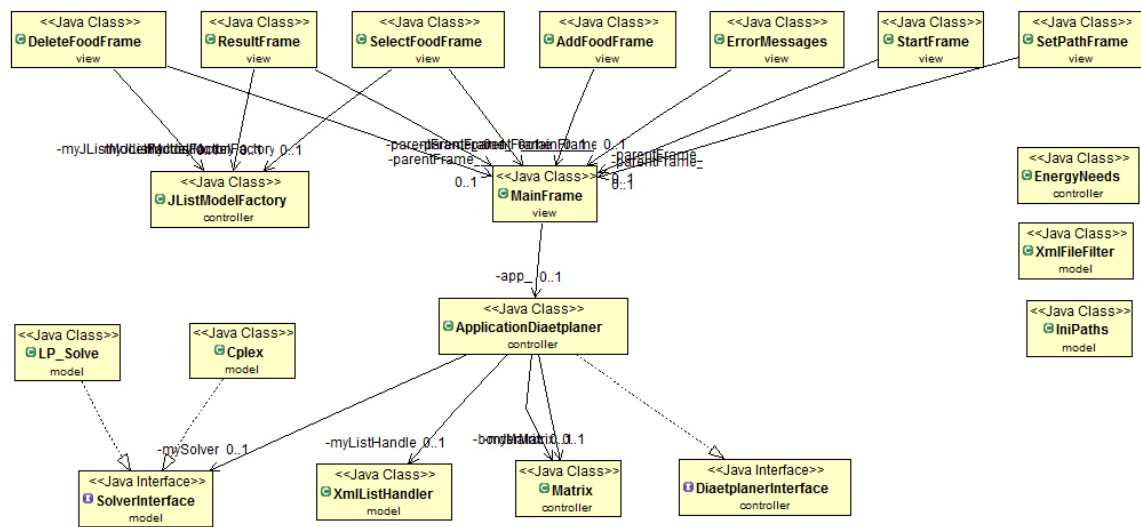
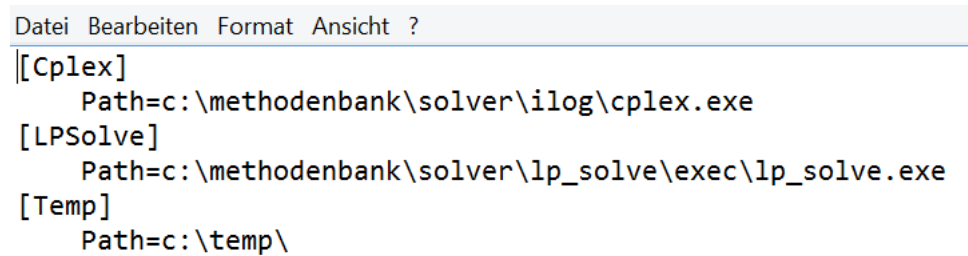


Abbildung 5: UML-Klassendiagramm der Methode nach dem Refactoring

## 4.2 Implementierung Menü „Solverpfade anpassen“

Zum einfachen Anpassen der Solverpfade wurde auf eine Lösung mit einer INI-Datei zurückgegriffen. Hierzu wurde eine INI-Datei „paths.ini“ erzeugt, in der sämtliche notwendige Pfade permanent gespeichert werden und somit auch nach Programmneustart wieder ausgelesen werden können.



```
Datei Bearbeiten Format Ansicht ?
[[Cplex]
    Path=c:\methodenbank\solver\ilog\cplex.exe
[LPSolve]
    Path=c:\methodenbank\solver\lp_solve\exec\lp_solve.exe
[Temp]
    Path=c:\temp\
```

Abbildung 6: Screenshot aus der Datei paths.ini

Zum Schreiben und Lesen der INI-Datei wurde eine neue Klasse „IniPaths.java“ erzeugt, welche für jeden Pfad eine Getter- und eine Setter-Methode beinhaltet. Um eine einheitliche Lösung für alle Java-Methoden zu erhalten, wurde diese Lösung auch an alle anderen Gruppen verteilt, sowie Herr Önnisan für zukünftige Gruppen übergeben.

```
public static String getLpSolvePath() {
    String result = "c:\\methodenbank\\solver\\lp_solve\\exec\\lp_solve.exe";
    String lpSolverPath = paths.leseString("LPSolve", "Path");
    if (lpSolverPath != null) {
        result = lpSolverPath;
    }
    return result;
}

public static void setLpSolvePath(String newLpSolvePath) {
    if (newLpSolvePath != null) {
        paths.setzeString("LPSolve", "Path", newLpSolvePath);
        paths.schreibeINIDatei(iniPath, true);
    }
}
```

Abbildung 7: Getter und Setter für den LP-Solve Pfad in der Klasse "IniPaths"

Um die Solverpfade im Programm anpassen zu können, wurde der View-Komponente eine weitere Klasse „SetPathFrame.java“ hinzugefügt, welche eine Oberfläche zum Eintragen der Pfade erzeugt. Diese Oberfläche wird durch einen Klick auf „Pfade anpassen“ erzeugt. „Pfade anpassen“ wurde in der Klasse „MainFrame.java“ als neues Element in die Statusleiste eingefügt und kann über den Menüpunkt „Solver“ ausgewählt werden.



Wird nach dem Eintragen eines neuen Pfads der Button „Speichern“ geklickt, so wird in der Klasse „IniPaths.java“ die jeweilige Getter-Methode aufgerufen, die den eingetragenen Pfad in die INI-Datei speichert.

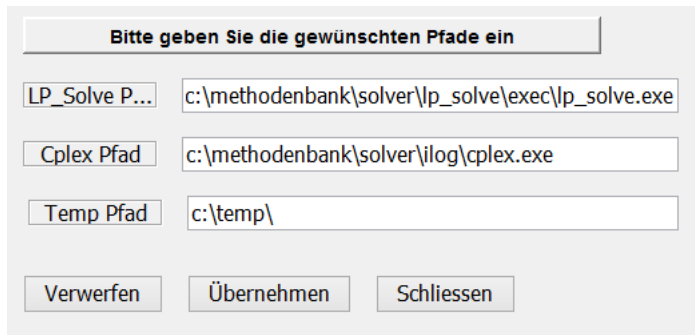


Abbildung 8: Fenster zum Setzen der Pfade in der Methode

### 4.3 Implementierung Cplex

Als zusätzliche Aufgabe sollte ein neuer Solver in die Methode implementiert werden. Im Package „Model“ wurde dazu die Java-Klasse „Cplex.java“ erzeugt. Die Klasse implementiert das Interface „SolverInterface.java“, die die benötigten Methoden für die Solver vorgibt. In der Klasse werden die benötigte IN-Datei, sowie die benötigte OUT-Datei für den Solver erzeugt, sowie der Solver aufgerufen und die IN-Datei mit dem LP-Ansatz übergeben. Der Solver schreibt die Lösung in die OUT-Datei, welche auf dem Temp-Verzeichnis liegt. Diese Datei wird in der Klasse „Cplex.java“ ausgelesen und das Ergebnis an die View-Komponente gesendet, um angezeigt zu werden.

### 4.4 Lebensmittel dauerhaft löschen und hinzufügen

Wie nach einigen Tests der Methode herausgefunden wurde, war es nicht möglich, Lebensmittel dauerhaft zu löschen oder hinzuzufügen. Wurden Lebensmittel in der Methode verändert, so wurden sämtliche Änderungen nach Neustart der Methode rückgängig gemacht. Durch weitere Überprüfung der Ursache wurde festgestellt, dass die XML-Datei, in der sämtliche Lebensmittel gespeichert sind nie verändert wurde und daher sämtliche Änderungen nach Neustart verloren waren. Zur Lösung dieses Problems musste im Programmcode an den jeweiligen Stellen nur die Methode „saveXmlList“ ausgeführt werden, die bereits in der Klasse „XmlListHandler.java“ implementiert war.

```
try {
    saveXmlList(xmlFilename);
} catch (Exception e) {
```

Abbildung 9: Aufruf der Save-Methode zum Speichern der Änderungen in der XML-Datei

#### 4.5 Anpassung der Hilfe-Datei

Die Hilfe-Datei wird beim „Ernährungsplaner 2.0“ als externe JAR eingebunden und in der Windows-Hilfe Oberfläche angezeigt. Zum Bearbeiten der Hilfedatei muss zuerst die JAR entpackt werden (z.B. mit WinRAR). Sämtliche HTML-Dateien können dann ohne Probleme mit einem Text-Editor bearbeitet werden. Die Hilfe-Datei ist im Programmverzeichnis unter „Ernaehrungsplaner.jar“ zu finden.

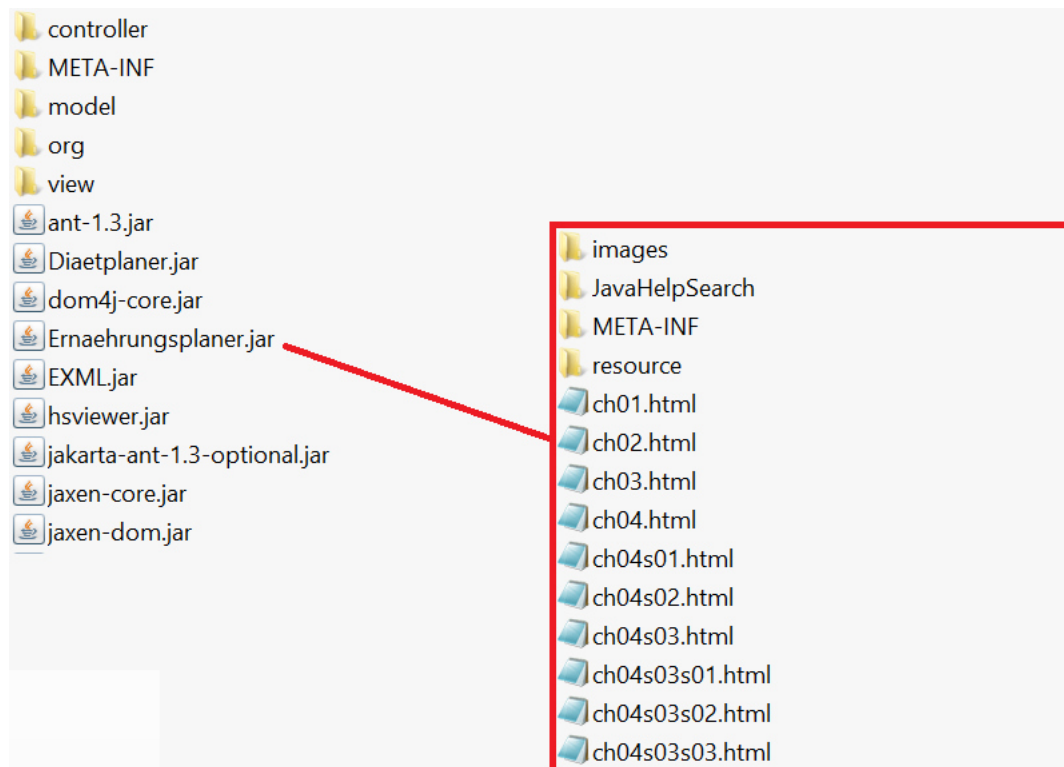


Abbildung 10: Programmverzeichnis mit Hilfe-Datei (rechts: Einzelne Kapitel der Hilfe-Datei)

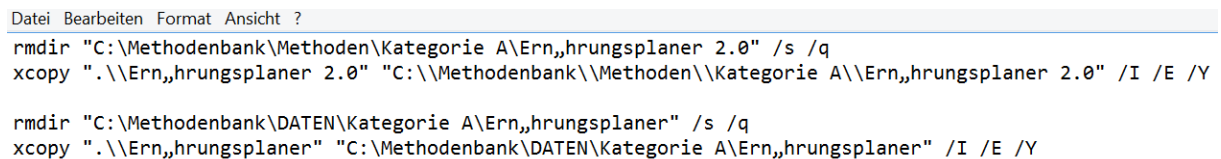
Wir haben aufgrund der von uns durchgeführten Änderungen an der Methode die Hilfe-Datei aktualisiert. Es wurden sämtliche Screenshots erneuert, die Architekturabbildungen und UML-Diagramme angepasst, sowie sämtliche neue Funktionen in die Hilfe-Datei eingefügt.

## 4.6 Implementierung Fehlerausgabe

Zur besseren Erkennung von Fehlern in der Methode wurde von uns eine Klasse „ErrorMessage.java“ erzeugt, welche sämtliche Fehlermeldungen im Programm abfängt und diese als Pop-Up-Fenster ausgibt. Dadurch kann der Benutzer einen Fehler erkennen, ohne sich diesen auf der Konsole ausgeben lassen zu müssen.

## 4.7 Erstellung Installationsdatei

Zur einfacheren Installation des überarbeiteten „Ernährungsplaners 2.0“ wurde eine Batch-Datei erzeugt, welche alle notwendigen Dateien auf „C:\Methodenbank\“ überschreibt. Es genügt ein einfacher Doppelklick auf den Installer (die Batch-Datei) und der Ernährungsplaner wird upgedatet. Die Batch-Datei löscht zuerst den alten Ernährungsplaner und kopiert danach die neuen Dateien an dieselbe Stelle. Die Ordnerstruktur wird dabei beibehalten.



```
Datei Bearbeiten Format Ansicht ?
rmdir "C:\Methodenbank\Methoden\Kategorie A\Ern,,hrungsplaner 2.0" /s /q
xcopy ".\Ern,,hrungsplaner 2.0" "C:\Methodenbank\Methoden\Kategorie A\Ern,,hrungsplaner 2.0" /I /E /Y

rmdir "C:\Methodenbank\DATEN\Kategorie A\Ern,,hrungsplaner" /s /q
xcopy ".\Ern,,hrungsplaner" "C:\Methodenbank\DATEN\Kategorie A\Ern,,hrungsplaner" /I /E /Y
```

Abbildung 11: Batch-Datei zur Installation des Ernährungsplaners

## 5 Quick-Installation Guide

Der Quick-Installation Guide beschreibt die Installation des Ernährungsplaners mit der im SS15 entwickelten Version 2.1.0.0 von OR-Alpha. Für ältere Versionen von OR-Alpha genügt das Ausführen des mitgelieferten Installers, um die Methode lokal zu installieren.

Zum Installieren der Methode auf allen HTWG-Rechnern mit OR-Alpha Version 2.1.0.0 muss erst eine lokale Installation erfolgen. Diese wird durch starten des Installers automatisch durchgeführt und sollte innerhalb weniger Sekunden abgeschlossen sein.

Für die weiteren Schritte ist es notwendig, am Computer mit einem Rechenzentrumsaccount angemeldet zu sein, der Schreibrechte auf den Merkur-Server besitzt. Diese Rechte werden von den zuständigen Serveradministratoren der Hochschule vergeben.

Sind diese Rechte vorhanden, so kann nun OR-Alpha geöffnet werden. Es ist zuerst sicherzustellen, dass die lokale Installation erfolgreich war. Dazu sollte der „Ernährungsplaner 2.0“ einmal ausgeführt und getestet werden. War die Installation erfolgreich, so muss nun die Admin-Oberfläche von OR-Alpha geöffnet werden. Es erscheint ein Login-Fenster, in dem der Benutzer sich mit dem Administratoraccount von OR-Alpha einloggen muss. Nach dem Login wird durch Klick auf den Button „Änderungen übertragen“ die neue Version des Ernährungsplaners auf den Merkur-Server geschrieben (ACHTUNG: Auch alle anderen Änderungen an Methoden werden dadurch an den Server gesendet!!!).

Nach erfolgreichem Deployment können sämtliche Änderungen von den zuständigen Serveradministratoren der Hochschule an alle Hochschulrechner verteilt werden.

1. Anmelden am Computer mit einem Account, der Schreibrechte auf Merkur besitzt
2. Lokale Installation durch Doppelklick auf den Ernährungsplaner Installer
3. Starten von OR-Alpha Version 2.1.0.0
4. Anmelden am Adminbereich von OR-Alpha
5. Deployment auf Merkur durch Klick auf „Änderungen übertragen“
6. Verteilen der neuen Version durch den Serveradministrator

## 6 Abbildungsverzeichnis

Abbildung 1: Änderungen an den Java-Klassen während des Refactorings .....	4
Abbildung 2: Package-Struktur mit Java-Klassen vor und nach dem Refactoring .....	5
Abbildung 3: Header in der Klasse ApplicationDietplaner zum besseren Verständnis .....	5
Abbildung 4: UML-Klassendiagramm der Methode vor dem Refactoring .....	6
Abbildung 5: UML-Klassendiagramm der Methode nach dem Refactoring .....	6
Abbildung 6: Screenshot aus der Datei paths.ini .....	7
Abbildung 7: Getter und Setter für den LP-Solve Pfad in der Klasse "IniPaths" .....	7
Abbildung 8: Fenster zum Setzen der Pfade in der Methode .....	8
Abbildung 9: Aufruf der Save-Methode zum Speichern der Änderungen in der XML-Datei....	9
Abbildung 10: Programmverzeichnis mit Hilfe-Datei (rechts: Einzelne Kapitel der Hilfe-Datei) .....	9
Abbildung 11: Batch-Datei zur Installation des Ernährungsplaners.....	10