Projektdokumentation zur Anwendung der linearen Optimierung

Werbebudgetoptimierung 1.3

Datum:
15.05.2016

Teilnehmer:
Bernhard Kaiser
289015
Eugen Kostilew
289073

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Im Rahmen der ALO Veranstaltung wird die Methode Werbebudgetoptimierung weiter verbessert. Dabei gilt es die Spezifikationen aus dem Pflichtenheft zu erfüllen.

2 Spezifizierungen aus dem Pflichtenheft

2.1 Solver Pfad

Die Spezifikation war „Der Benutzer muss den Solver Pfad manuell im Programm ändern können.“ Damit dies möglich ist, wird zunächst nachvollzogen, wie die Einbindung des Solvers mit der Methode erfolgt.
Der Werbebudgetoptimierer benutzt dazu 3 Dateien.

- Solver.in
- Solver.out
- solver.bat

Dabei enthält die „.in“ den LP Ansatz, die „.out“ enthält die Informationen die vom Solver zurückgegeben werden und die „.bat“ welche eine Windowsbatchdatei ist.

```
@echo off
cd C:\Mull\Solver\
LP_SOLVE.EXE <C:\Temp\Solver.in> C:\Temp\Solver.out
@echo on
```

Abbildung 1: Inhalt der solver.bat

Die „.bat“ enthält den eigentlichen Pfad des Solvers. Damit das ganze flexibler wird, ist der Code so angepasst, dass der Werbebudgetoptimierer 1.3 die Windowsbatchdatei erzeugt oder anpasst.

Dadurch lässt sich das System einerseits stabilisieren, da es nicht mehr notwendig ist jedes Mal auch die Windowsbatchdatei mit zu kopieren, wenn die Methode verschoben wird.

Folgend kommen nun die Code Änderungen im genauen:

```
// statische Variablen um die Zuordnung zu vereinfachen
public final static int NEU = 1;
public final static int OEFFNEN = 2;
public final static int SCHLIESSEN = 3;
public final static int SICHERN = 4;
public final static int VERLASSEN = 5;
public final static int KOPIEREN = 6;
public final static int AUSSCHNEIDEN = 7;
public final static int EINFUEGEN = 8;
public final static int HILFE = 9;
public final static int PFAD = 10;
```

Abbildung 2: Menue.class; Variablen Anpassung

Erweitern der Statischen Variablen um die Pfad Variable. Dient lediglich der Steuerung innerhalb des Code.

```
datei.add(item = new JMenuItem("Solver Pfad"));
item.setActionCommand(String.valueOf(PFAD));
item.addActionListener(this);
/*
```

Abbildung 3: Menue.class; Erzeugen eines weiteren Menü Punktes

Erzeugen des Menü Punktes „Solver Pfad“ damit der Anwender bequem und jederzeit den Solver Pfad unter einem Menü Punkt erreichen kann.

```
case 10: {
    try {
        wb.solverPfad();
    } catch (IOException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
}
```

Abbildung 4: Menue.class; erweitern der Switch-Case

Die Steuerung des „ActionListener“ wurde im Code durch eine Switch-Case realisiert. Damit das Programm mit der Neuen Solver Pfad Funktion umgehen kann, ist an dieser Stelle der Case 10 angelegt. Innerhalb dieses Case wird eine Methode „solverPfad()“ aufgerufen.

```java
@SuppressWarnings("deprecation")
public void solverPfad() throws IOException {
    FileDialog f = new FileDialog(this, "Solver Pfad", FileDialog.LOAD);

    f.show();
    String filename = f.getFile();
    String directory = f.getDirectory();

    FileWriter writer = new FileWriter("solver.bat");
    writer.write("@echo off\ncd "+directory+ "\n"+filename+" <C:\\Temp\\Solver.in> C:\\Temp\\Solver.out\n@echo on");
    writer.close();
}
```

Abbildung 5: WerbebudgetApplication.class; Methode solverPfad()

Damit der neue Menü Punkt seine Aufgabe erfüllt, ist in dieser Methode die Funktionalität Programmiert. Da der „FileDialog“ eine sehr bequeme Art ist nach dem Richtigen Pfad zu suchen, wird der „FileDialog“ hier auch benutzt.

Der „FileDialog“ macht es auch einfach den Pfad und den Dateinamen zu generieren.

Mit dem Dateinamen und dem Pfad wird eine neue Windowsbatchdatei geschrieben oder eine alte überschrieben. Die Windowsbatchdatei landet dabei immer in dem Ordner wo auch der Werbebudgetoptimierer liegt.

2.2 Navigation im Programm

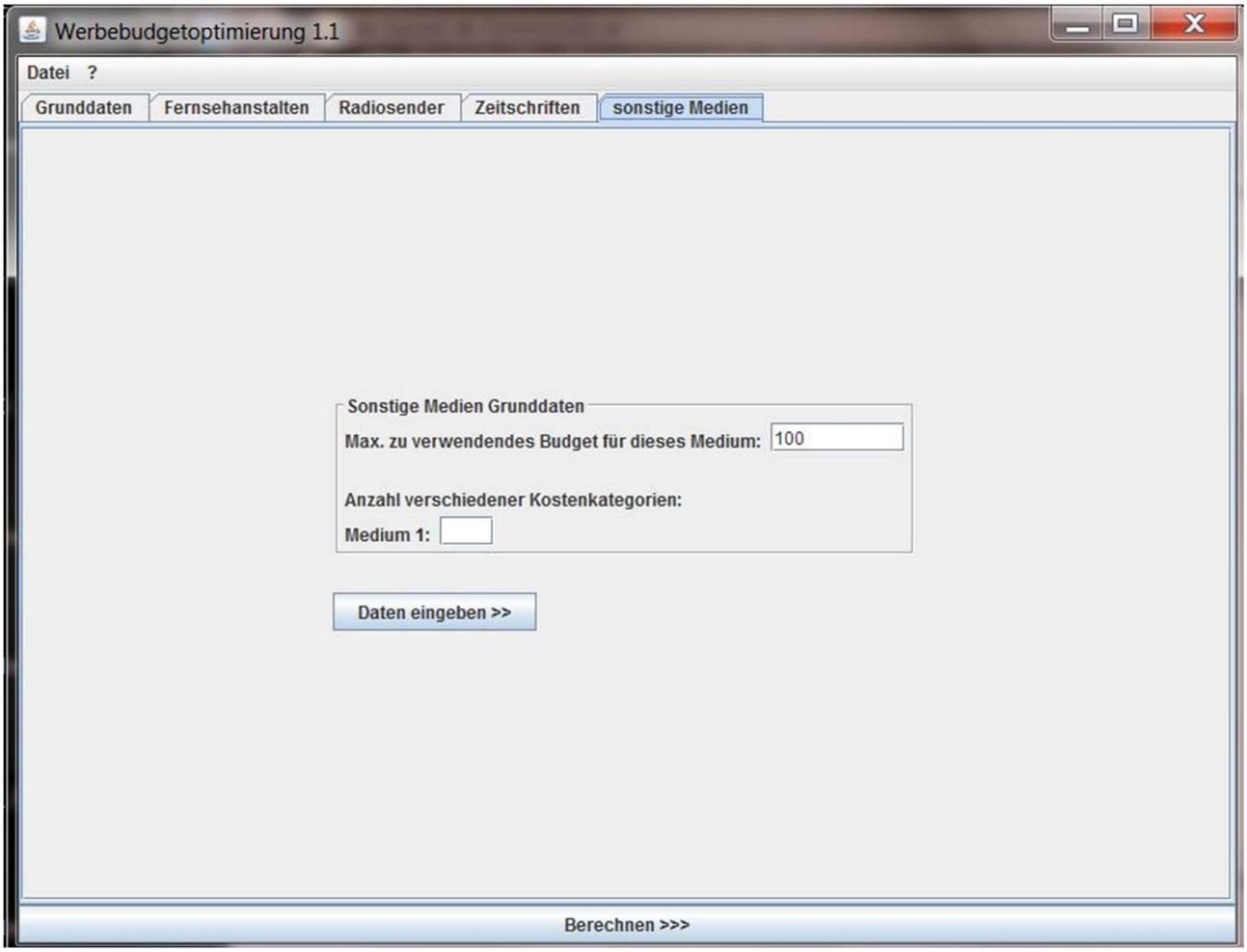

Abbildung 6: Medium Grunddaten

Sobald im Werbebudgetoptimierer bei den Medien Grunddaten eine Tabelle Erzeugt wurde, ließ sich die Tabelle nicht mehr anpassen ohne alle Medien zurück zu setzten.

Damit das möglich ist, wurde ein Zurück Button Programmiert.

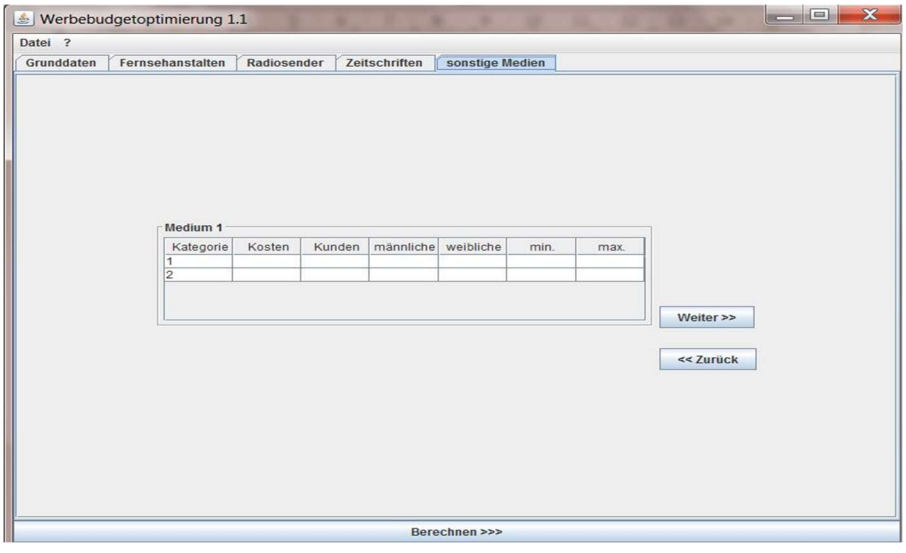

Abbildung 7: Medium Zurück Button

Die genauen Code Veränderungen sind wie folgt beschrieben:

```
public abstract class Medium implements ActionListener {
    // statische Werte, um die Zuordnung zu vereinfachen
    public final static int DATEN_EINGEBEN = 1;
    public final static int WEITER = 2;
    public final static int ZURÜCK = 3;
```

Abbildung 8: Medium.class; Statische Variable

Die Statische Variable „ZURÜCK“ mit dem Wert 3 wurde programmiert.

```
// Swing-Komponenten, welche angezeigt werden
protected JLabel[] anzahlLabel;
protected JTextField[] anzahlField;
protected JButton datenEingebenButton;
protected JButton weiterButton;
protected JLabel maxBudgetLabel;
protected JTextField maxBudgetField;
protected JButton zurückButton;
```

Abbildung 9: Medium.class; Swing-Komponente

Weiter musste eine „zurückButton“ angelegt werden, damit ein Steuerelement

```
zurückButton = new JButton("<< Zurück");
zurückButton.setActionCommand(String.valueOf(ZURÜCK));
zurückButton.addActionListener(this);
```

vorhanden ist.

Abbildung 10: Medium.class; Erzeugen des Buttons

Der Button wird an der Stelle hier erzeugt und benannt. Außerdem wird der Zurück Button noch an den „ActionListener“ angebunden.

```
c.gridx = 1;
c.gridy = (index - 2) * 2;
c.gridwidth = 1;
c.gridheight = 1;
c.anchor = GridBagConstraints.SOUTHWEST;
medienPanel.add(zurückButton, c);
```

Abbildung 11: Medium.class; Positionieren des Zurück Button

Im nächsten Schritt muss der Zurück Button positioniert werden und an das Panel gebunden werden.

```
if (Integer.parseInt(e.getActionCommand()) == ZURÜCK){

    initMedium();
    medienPanel.repaint();
}
```

Abbildung 12: Medium.class; Ausführung von Zurück Button

Abschließend muss dem Zurück Button noch Funktionalität gegeben werden, hierzu wird dem „ActionListener“ in Form einer If-Abfrage gesagt, dass sobald der ZURÜCK Wert übergeben wird, die Methoden „initMedium()“ und „medienPanel.repaint()“ aufgerufen werden.

Die Methode „initMedium()“ initialisiert nochmal die einzelnen Medien Reiter und war schon vorhanden. Die Methode „medienPanel.repaint()“ dient dazu die Medium Reiter im Programm zu aktualisieren.

2.3 Anpassen der Versionsnummer

Eine weitere Aufgabe bestand darin, die vorhandene Versionsnummer von 1.1 auf 1.3 abzuändern, hierzu wurden im Code die Bezeichner angepasst.

2.4 Handbuch

Zusätzlich zu dem Programmier-Anteil musste ein Handbuch geschrieben werden.

3 Fazit

Die Anforderungen wurden alle Umgesetzt und sind Funktionstüchtig. Die Arbeit an dem Werbebudgetoptimierer 1.3 war anspruchsvoll, aber im Rahmen der ALO Veranstaltung realisierbar. Auch die Tatsache das Arbeiten an Fremden Code für gewöhnlich sehr anspruchsvoll ist, war dies durch eine gute Kommentierung des Codes Möglich.