HTWK Leipzig, FIMN
Dipl.-Math. Dörte König

Anwendungsorientierte Programmierung II
Hinweise zur Übung 9

Aufgabe 1

1. Model-View-Controller-Architektur

Architekturmuster zur Unterteilung einer Software in die drei Komponenten **Datenmodell** (`model`), **Präsentation** (`view`) und **Programmsteuerung** (`controller`).

Model	→	Klasse `NoteBookData`
View	→	Klasse `NoteBookGUI`
Controller	→	Klasse `NoteBook`

2. Klasse NoteBookData

Die Klasse dient der Verwaltung der Daten des Notizbuches als Einträge einer HashMap.

Attribute: HashMap für Wochentag und Termine

```
private Map<String,String> notes = new HashMap<String,String>();
```

Methoden: Speicherung der Daten als (`key`,`value`)-Paare in der HashMap

```
public void speichernTermin(String key, String value) {}
```

Rückgabe eines Wertes (`Termine`) zu übergebenem Schlüssel (`Wochentag`)

```
public String holeTermin(String key) {}
```

3. Klasse NoteBookGUI

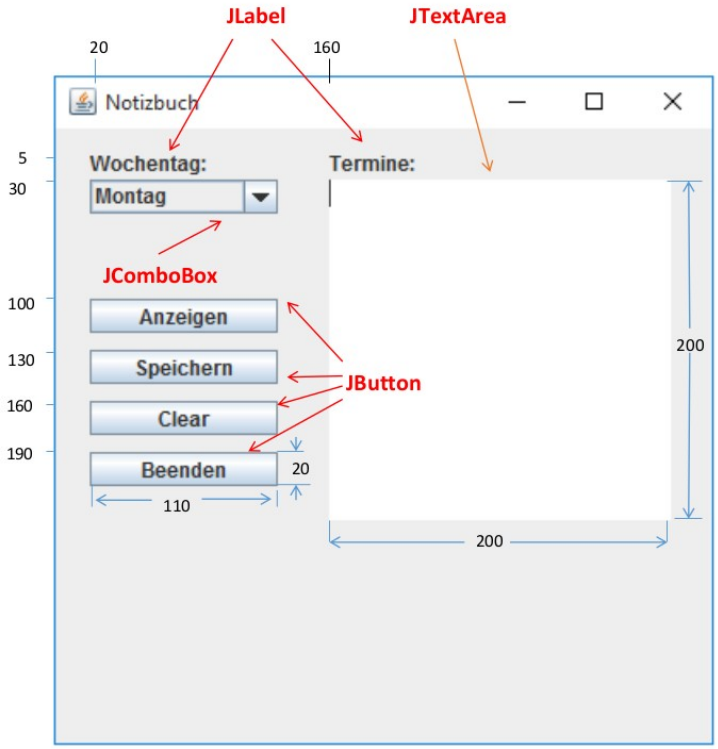
Die Klasse `JFrame` ist die wichtigste Hauptfensterklasse im Framework `Swing`.
Erzeugung der grafischen Oberfläche als `JFrame` :

```
public class NoteBookGUI extends JFrame {
	//Verknuepfung mit der Daten-Klasse
	private NoteBookData daten;

	//Attribute fuer alle Komponenten der GUI
	private JLabel termineLabel;
	private JTextArea termine;
	//...

	public NoteBookGUI (NoteBookData d) {
		this.daten = d;

		//Layout festlegen
		this.setLayout(null);

		//ausserdem:
		//Titel+Groesse+Location fuer das Fenster
		//Farbe der ContentPane aendern
		this.getContentPane().setBackground(Color.orange);

		//alle Komponenten erzeugen+konfigurieren+hinzufuegen
	}
}
```

Komponenten
Für die abgebildete Notizbuch-Anwendung werden folgende Komponenten aus dem Framework `Swing` als Attribute der Klasse NoteBookGUI benötigt:

Komponente	**Funktion(en)**
JButton	Anzeigen, Speichern, Clear, Beenden
JLabel	Bezeichner Wochentag, Bezeichner Termine
JComboBox	Wochentag-Auswahl
JTextArea	Eingabefläche für Termine

Anordnung der Komponenten



Hinzufügen einer Komponente

```
//Erzeugung der Komponente
termineLabel = new JLabel("Termine");
//Konfiguration der Komponente
termineLabel.setBounds(160,5,110,30);
//Platzieren auf der ContentPane
this.add(termineLabel);
```

ComboBox für die Wochentage
Eine ComboBox braucht für die Initialisierung zusätzlich ein Array mit den **Auswahloptionen**:

```
private JComboBox<String>wochentag;
private String[] tage={"Montag","Dienstag","Mittwoch",
                       "Donnerstag","Freitag","Samstag","Sonntag"};

//Erzeugung der Komponente
wochentag= new JComboBox<String>(tage);
```

Ereignisbehandlung

Verwendung von Lambda-Ausdrücken ab Java 8 möglich
Lambda-Ausdruck: Liste formaler Parameter → Funktionsrumpf / Methodenaufruf

Beispiel:

```
private JButton showButton;
//...
//im Konstruktor:
showButton.addActionListener(e-> anzeigen());
```

Die Auswertung des Events erfolgt in der Methode `anzeigen()` außerhalb des Konstruktors:

```
public void anzeigen()
{
	String w= wochentag.getSelectedItem().toString();
	String t = daten.holeTermin(w);
	if(t!=null)
		termine.setText(t);  //termine – Attribut fuer JTextArea
	else
		termine.setText("Keine Termine!");
}
```

Beispiel: Bei wenig Quellcode kann die Auswertung des Ereignisses auch direkt in den Lambda-Ausdruck integriert werden:

```
exitButton.addActionListener(e->{System.exit(0);});
```

4. **Klasse NoteBook**
   Die Klasse erzeugt eine Notizbuch-Instanz mit Daten und GUI und schaltet die Anwendungsoberfläche sichtbar.

```
public class NoteBook
{
	private NoteBookData data;
	private NoteBookGUI view;

	//Konstruktor
	public NoteBook()
	{
		data = new NoteBookData();
		view = new NoteBookGUI(data);
	}

	public static void main(String[] args)
	{
		NoteBook notebook = new NoteBook();
		//Sichtbarkeit des Notebooks
		notebook.view.setVisible(true);
	}
}
```

Erster Test des Fensters:

a. erzeuge leere Klasse `NoteBookData`

b. erzeuge Klasse `NoteBookGUI` mit Instanz `daten` der Klasse NoteBookData als Attribut und mit Konstruktor

```
public NoteBookGUI (NoteBookData d)
{
   this.daten = d;
}
```

c. erzeuge Klasse `NoteBook` wie in Punkt 4 beschrieben

Das so erstellte Projekt lässt sich schon starten.

Da die GUI noch nicht spezifiziert wurde, erscheint sie als sehr kleines Fenster rechts (oder links) oben in der Ecke des Bildschirms. Es kann vergrößert werden und besitzt die üblichen Schaltflächen einer Applikation. Beenden Sie anschließend die Anwendung in der Ausgabekonsole.

Aufgabe 2

Permanente Speicherung des Notizbuchs als Object in einer Datei.
Erweiterung der Klasse NoteBookData um ein Attribut für die Datei

```
private File file = new File("./Daten/daten.txt");
```

sowie Methoden zum Lesen bzw. Speichern der HashMap in der Datei.

Speicherung der HashMap als Objekt

```
FileOutputStream fos = new FileOutputStream(file);
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(notes);
System.out.println("Daten wurden in Datei geschrieben!");
oos.close();
fos.close();
```

Lesen der Daten aus einem gespeicherten Objekt in die HashMap

```
FileInputStream fis = new FileInputStream(file);
ObjectInputStream ois = new ObjectInputStream(fis);
notes= (HashMap<String, String>) ois.readObject();
System.out.println("Daten wurden aus Datei gelesen!");
ois.close();
fis.close();
```

In beiden Fällen ist eine Exceptionbehandlung erforderlich!