

Einführung in die Programmierung für Kunst und Multimedia

Übungsblatt 11

Ende der Abgabefrist: 23.01.2015 10:00 Uhr

Hinweise zur Abgabe:

Übungsblätter dürfen NICHT in Teams abgegeben werden, da Sie sich durch eine erfolgreiche Bearbeitung einen Bonus für die Klausur verdienen können. Es ist zwar sinnvoll in kleinen Teams die Aufgaben zu diskutieren, die Lösungen müssen aber von jedem Studenten EINZELN bearbeitet werden. Bitte beachten Sie, dass abgeschriebene Lösungen mit 0 Punkten bewertet werden!

Sammeln Sie die Lösungen zu diesem Übungsblatt in einem zip-Archiv loesung11.zip. Dieses zip-Archiv können Sie schließlich in UniWorX abgeben.

Wichtig: Achten Sie bitte darauf, dass Ihre Lösungsdateien den korrekten Namen und das korrekte Format haben! Beides wird in der Angabe explizit angegeben. Dateien im falschen Format oder mit falschem Namen werden im Allgemeinen nicht korrigiert.

Aufgabe 11-1 Teilnehmerliste

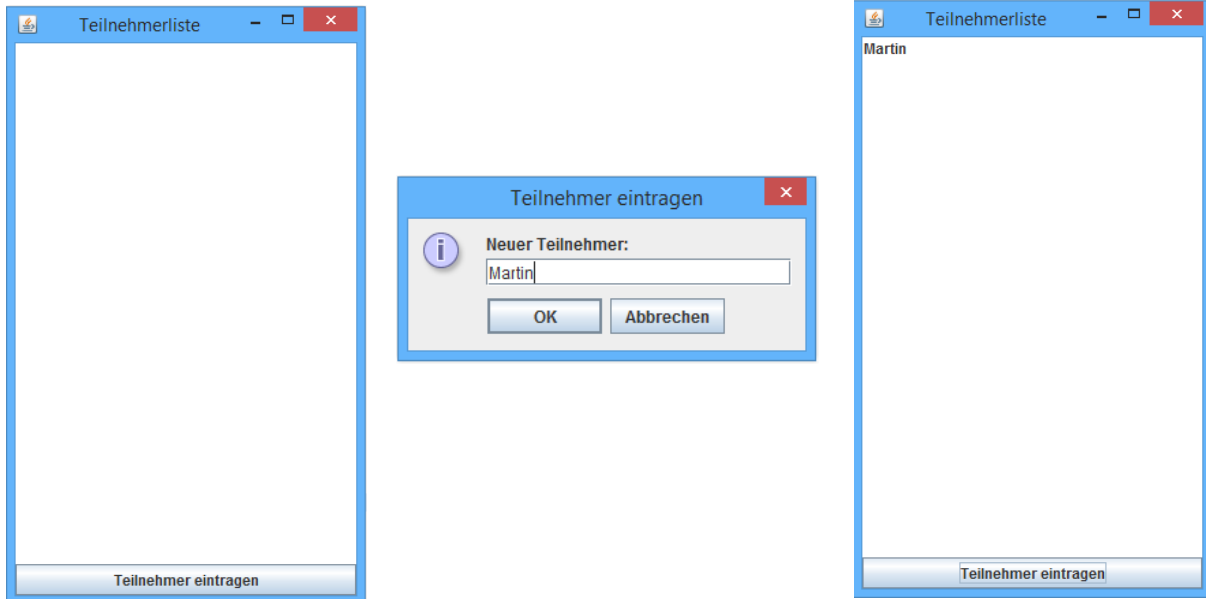
14 Punkte

In dieser Aufgabe sollen Sie ein Programm erstellen, mit dessen Hilfe Namen auf eine Liste eingetragen werden können. Das Programm soll das MVC- sowie das Observerpattern verwenden. Die eingetragenen Namen werden in einer Liste im Model gespeichert; wenn diese Liste verändert wird, soll sich die View automatisch aktualisieren und den Inhalt dieser Liste anzeigen.

Erstellen Sie dazu die vier Klassen *Teilnehmerliste.java*, *Model.java*, *Controller.java* und *View.java*. *Teilnehmerliste.java* enthält die main-Methode, die eine neue Instanz von Controller erzeugt. Verknüpfen Sie im Controller Model und View im Sinne des Observer-Patterns. Denken Sie daran, dass die View die Instanz des Controllers kennen muss (*this*), diesen also bei ihrer Erstellung mit übergeben bekommt, um später über den Controller die Daten des Models auslesen zu können.

Der Controller enthält neben dem Konstruktor Methoden, die Befehle von der View an das Model übergeben. Die View kennt das Model nicht und ruft daher nur Methoden im Controller auf, die an das Model weitergegeben werden. Das Model enthält eine leere Liste für Strings, eine Methode, um dieser Liste Strings hinzuzufügen sowie eine Methode, die die Liste zurückgibt. Beim Hinzufügen eines Eintrags zur Liste werden die Observer benachrichtigt.

Die View enthält eine JList zur Anzeige der Namen sowie einen Button in der gezeigten Anordnung, der einen InputDialog öffnet, mithilfe dessen Namen eingetragen werden können.



Wenn ein Name eingegeben wurde, wird die entsprechende Funktion im Controller mit diesem Namen aufgerufen, die wiederum die Methode im Model aufruft, die den Namen an die Liste im Model anhängt. Da dort auch die Observer benachrichtigt werden, wird automatisch die update()-Methode der View ausgeführt.

In der update()-Methode der View muss die angezeigte JList geleert werden, die Namensliste mithilfe des Controllers aus dem Model ausgelesen werden und dann die Einträge der Modelliste auf die JList übernommen werden.

Bitte geben Sie Ihr vollständiges Eclipse-Projekt in einem Ordner *aufgabe1* ab. Verwenden Sie dazu einfach den gesamten Ordner Ihres Projektes im workspace. Bitte achten Sie unbedingt darauf, dass Ihre Lösung fehlerfrei und kompilierbar ist. **Es werden ausschließlich lauffähige Lösungen im korrekten Format bewertet.**

Aufgabe 11-2 Interfaces

6 Punkte

In der Vorlesung haben Sie Interfaces als Möglichkeit kennengelernt, einer Klasse, die dieses Interface implementiert, Methodennamen und –Parameter vorzugeben.

Erstellen Sie ein Interface *IKonto.java*, das für die Implementierung eines Bankkontos, wie es z.B. in der Banksimulation verwendet wurde, entsprechende Methoden vorgibt. Folgende Methoden soll jedes Konto, das dieses Interface implementiert, beinhalten:

- Methode für Einzahlungen
- Methode für Auszahlungen
- Methode zur Kontostandsabfrage

Denken Sie dabei auch an sinnvolle Parameter.

Erstellen Sie zudem eine Klasse *Konto.java*, die dieses Interface implementiert und die Funktionen sinnvoll umsetzt. Geben Sie die beiden Klassen *IKonto.java* und *Konto.java* im Ordner *aufgabe2* ab. **Es werden ausschließlich lauffähige Lösungen im korrekten Format bewertet.**

Achten Sie auf eine korrekte Dokumentation, also Kommentierung Ihres Codes! Vor jeder Klasse und jede Methode muss dabei ein korrekt definierter Javadoc-Kommentar stehen!

Fehlende Javadoc-Kommentare führen Aufgaben-übergreifend zu Punktabzug!

Abgabe

Zulässige Dateiformate für die Lösungen dieses Übungsblattes ist JAVA. Bitte geben Sie Ihre Lösung als ZIP-Datei bis zum 23.01.15 10:00 Uhr in UniWorX (<https://uniworx.ifi.lmu.de>) ab.

Hinweis: Verspätete Abgaben, Abgaben im falschen Dateiformat und nicht lauffähige Java-Dateien werden nicht bewertet.