Design Pattern SS 2013
Hausaufgabe 8

UNIKASSEL VERSITÄT

Die Aufgaben müssen einzeln bearbeitet und abgegeben werden. Die Abgabe muss **bis spätestens Montag, den 01.07.2013 um 23:59 Uhr** über unser Hausaufgabenabgabesystem **http://seblog.cs.uni-kassel.de/dpss13/** erfolgen. Die Abgabe ist nur als einzelne *.zip oder *.jar-Datei möglich. Daher müssen alle für eine Abgabe relevanten Daten zu einem solchen Archiv kombiniert werden.

Hinweise zur Abgabe:

- Die Hausaufgabe als exportiertes Eclipse Projekt (*.zip, nicht den gesamten Workspace) abgeben. Das kann mit Hilfe der Eclipse Export Funktion durchgeführt werden. Ist das Projekt nicht korrekt exportiert, kann es bei der Korrektur nicht berücksichtigt werden (es bietet sich also an, den Import des exportierten Projektes auszuprobieren).

  WICHTIG Benennen Sie ihr Projekt für diese Abgabe nach folgendem Schema:

  DPSS13_HA<a>_<Matrikelnummer>,

  wobei <a> für die aktuelle Hausaufgabe steht.
  Beispiel:
  DPSS13_HA8_12345678.

Allgemeines
Orientieren Sie sich für die Lösung der Aufgaben an den zugehörigen Übungen und Vorlesungen:
http://seblog.cs.uni-kassel.de/category/currentterm/design-patterns/

Die Benotung beruht auf den Hausaufgaben.

Hierfür werden die gesamten Hausaufgaben minus zwei vom Studenten abgegebenen Hausaufgaben addiert, die mit mehr als 50 % bewertet wurden. Nicht abgegebene Hausaufgaben oder Betrugsversuche bekommen 0 Punkte.

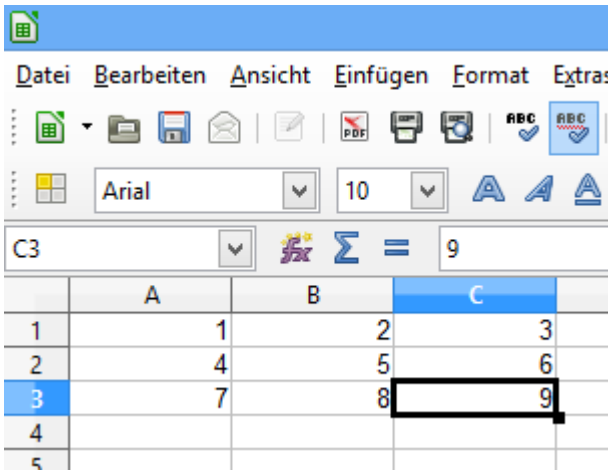
Die Prüfung gilt als nicht bestanden, wenn alle Hausaufgabenpunkte weniger als **50 %** ergeben oder der Student mehr als **zwei Hausaufgaben nicht abgibt**.

Fragen bitte an Stefan.Lindel@cs.uni-kassel.de oder george@uni-kassel.de

Aufgabe 1 (Facade Pattern) (3P)

- Hierfür soll ein Excel-Export entworfen werden.
- Die Erstellung von Excel-Dateien soll mit dem Facade Pattern programmiert werden.
- Hierfür werden folgende Elemente benötigt: ExcelFacade und die für das Pattern erforderlichen Klassen
- Der ExcelFacade soll zwei Methoden enthalten
  - boolean setCellValue(workbook::String, sheet::String, row::Int, cell::Int, String value)
    - Die Methode schaut nach ob es die entsprechende Zelle gibt oder erstellt diese ggf. danach setzt die den Zellenwert
  - String getCellValue(workbook::String, sheet::String, row::Int, cell::Int)
    - Die Methode schaut nach ob es die entsprechende Zelle exitiert und gibt es den aktuellen Zellenwert zurück oder gibt null zurück.
- Bei dem ExcelFacade Pattern soll sich auf die Apache POI Bibliothek abstützen
- http://poi.apache.org/apidocs/org/apache/poi/hssf/usermodel/HSSFWorkbook.html

- Das System soll in einem Test, der eine einfache Excel Datei erzeugt getestet werden.
  - Der Test sollen drei Zeilen mit den Werten raus schreiben.



Aufgabe 2 (Singleton-Pattern) (3P)

- In dieser Aufgabe soll ein Taschenrechner mit Client-Server gebaut werden.
- Hierfür werden folgende Elemente benötigt: Calculator, Server, Client, SocketConnection
- Der Server läuft ein einem Thread und erzeugt für jeden ankommenden Connection einen neuen SocketConnection wo er die Befehle mittels readline gelesen und abarbeitet werden.
  - Die SocketConnection soll die Befehlezeilen abarbeiten und diese ggf. an den Calculator Singleton weiterleiten. Die Befehle heißen:
    - add <Number>
    - minus <Number>
    - result
  - Der gemeinsame Calculator soll die Methode getInstance() besitzen.
  - Weiterhin soll der Calculator sicher gegen Fremdinstanzieren erstellt werden. Siehe Alberts Screencast.
  - Der gemeinsame Calculator fügt bei den Befehlen add und minus die Zahl zu ihrer Liste von Integern den aktuellen Wert hinzu <Calculator>:addNumber(<Number>).
  - Bei dem Befehl result rechnet der Calculator das Ergebnis aus <Calculator>:getResult() und gibt es dann auf der Konsole aus. Weiterhin gibt es das Ergebnis an den Client zurück.
  - **Zusätzlich: Kann eine Oberfläche programmiert werden.**
    - **Weiterhin soll eine zusätzliche Methode addNumber(<Operator>, <Number>) zu dem Calculator erstellt werden**
    - **Es soll nun auch zusätzliche Befehlezeilen geben wie**
      - **open bracket**
      - **close bracket**
      - **multiply <Number>**
      - **div <Number>**
      - **resultCalculator Ermittelt den richtigen Wert an hand mathematischer Vorgaben Puntk vor Stich und Klammersetzung**
    - Das System soll in einem Test, mittels einer einfachen Rechnung validiert werden
  - add 5
  - add 40
  - minus 3
  - result
- getestet werden.

Fragen bitte an Stefan.Lindel@cs.uni-kassel.de oder george@uni-kassel.de