Design Patterns SS 2014
Hausaufgabe 3

UNIKASSEL VERSITÄT

Die Aufgaben müssen von jedem Teilnehmer einzeln bearbeitet und abgegeben werden. Die Abgabe muss **bis spätestens Montag 12.05.2014 um 23:59 Uhr** über unser Hausaufgabenverwaltungssystem **https://se.cs.uni-kassel.de/hms/** 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 ist in Form von **drei** exportierten Eclipse Projekten abzugeben. Mit Hilfe der Eclipse Export Funktion ist es möglich drei Projekte in eine zip-Datei zu exportieren. Ist ein 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 ihre Projekte für diese Abgabe nach folgendem Schema:

  DPSS14_HA<a>_<b>_<Matrikelnummer>,

  wobei <a> für die aktuelle Hausaufgabe und <b> für die Aufgabennummer steht.
  Beispiel für Aufgabe 1:
  DPSS14_HA3_1_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-patterns2014/

Die Benotung ergibt sich aus den Hausaufgaben, wobei eine Abgabe ausgelassen werden darf. Für die Note wird die nicht abgegebene Abgabe, beziehungsweise die Abgabe mit der geringsten Prozentzahl, nicht beachtet. Die Endnote ergibt sich aus dem Mittelwert der erreichten Prozentpunkte der übrigen Abgaben.

Die Veranstaltung gilt als nicht bestanden, wenn mehr als **eine Hausaufgabe nicht abgegeben** wurde oder der Mittelwert der zur Benotung herangezogenen Abgaben **weniger als 50%** beträgt.

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

Aufgabe 1 Strategy Pattern

Implementieren Sie ein Datenmodell für ein Heizkreislauf-System unter Verwendung des Strategy Pattern

- Bilden Sie dabei folgende Teile des Systems ab:
  - HeatingBoiler
  - HeatingStrategy
- Integrieren Sie zwei Attribute vom Typ **`int`** in die Strategy, die über get-Methoden abgefragt werden können:
  - boilerTemperature
  - bufferTemperature
- Implementieren Sie folgende Strategien inklusive Attribut-Belegungen:
  - HeatUpBoiler (boilerTemp = 30, bufferTemp = 30)
  - HeatUpBuffer (boilerTemp = 60, bufferTemp = 30)
  - RadiatorCirculationOn (boilerTemp = 60, bufferTemp = 60)
  - RadiatorCirculationOff (boilerTemp = 90, bufferTemp = 90)
- Schreiben Sie einen Junit-Test in welchem
  - eine sinnvolle Objektstruktur erzeugt wird
  - die Strategien ausgetauscht und abgefragt werden
  - sinnvolle Assertions verwendet werden

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

Aufgabe 2 Hook Pattern

Implementieren Sie ein Datenmodell für einen Geldautomaten-Systems unter Verwendung des Hook Pattern

- Bilden Sie dabei folgende Teile des Systems ab:
  - CashTerminal, BankAccount, Person, HookStrategy
- Integrieren Sie die Methode *payOut(Person person) : void* in die Strategy.
- Implementieren Sie folgende Strategien:
  - RequestPIN
    - fragt über eine Methode die PIN der Person ab und gleicht sie mit der des zugeordneten Kontos ab
  - RequestAmount
    - fragt über eine Methode den Betrag ab, welchen die Person wünscht
  - CheckBalance
    - überprüft, ob auf dem der Person zugeordneten Konto, ein ausreichender Betrag vorhanden ist
  - Dispense
    - verringert den Kontostand der Person, reduziert das im Automat vorhandene Bargeld und übergibt der Person das Geld über einen Methodenaufruf
- Schreiben Sie einen Junit-Test in welchem
  - eine sinnvolle Objektstruktur erzeugt wird (Reihenfolge der Strategien beachten)
  - die Methode payOut*(..)* aufgerufen wird
  - sinnvolle Assertions verwendet werden

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

Aufgabe 3 Chain of Responsibility Pattern

Stellen Sie das Strategy Pattern aus Aufgabe 1 auf eine Chain of Responsibility um.

- Dazu zählen auch Anpassungen an sämtlichen Strategie-Klassen.
- Ob eine Strategie verwendet werden kann, hängt von den aktuellen Belegungen der Attribute boilerTemperature und bufferTemperature ab.
- Testen Sie ihre Chain of Responsibilty nach dem gleichen Vorgehen wie im Test für Aufgabe 1.

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