Die Aufgaben müssen einzeln bearbeitet und abgegeben werden. Die Abgabe muss **bis spätestens Montag, den 03.06.2013 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_HA6_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 (Flyweight) (3P)

- Hierfür soll ein kleines Textverarbeitungsprogramm entworfen werden.
- Die Formatierungen der einzelnen Buchstaben sollen nach dem Flyweight-Pattern implementiert werden.
- Hierfür werden folgende Elemente benötigt: CharItem, Style
- In dem Test sollen 10 CharItems mit dem Inhalt: {H,A,L,L,O, ,W,E,L,T} und einer Standardformatierung intialisiert werden. Danach soll für die Buchstaben {W,E,L,T} die Formatierung so angepasst werden, dass das Wort Fett ausgedruckt wird.

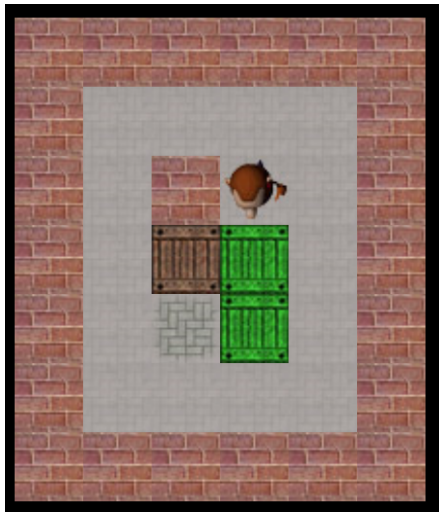
Fragen bitte an Stefan.Lindel@cs.uni-kassel.de oder george@uni-kassel.de

Aufgabe 2 (Command-Pattern) (6P)

- In dieser Aufgabe soll das Spiel Sokoban entwickelt werden
- Hierfür werden folgende Elemente benötigt: Field, Player, Map,
  Enum StandOn (Wall (#), Blank(„ „), Box(x), Player(P), Target(+), Target mit Box(*), Player auf einem Target (B))
  und die für das DesignPattern erforderlichen Klassen
- Die Felder besitzen ein Attribut, welches Element auf dem jeweiligen Feld zu finden ist
- Die Map kann so implementiert werden, dass Sie eine Assoziation zu allen Feldern besitzt
- Das Pattern soll so implementiert werden, dass anhand der Pfeiltasten auf der Tastatur der Spieler bewegt werden kann, um somit die Boxen auf die Zielfelder zu schieben. Es weder erlaubt 2 Boxen zu schieben, noch eine Box zu ziehen. (Beispiel: http://www.pottersworld.de/sokuban/)
- Das Spiel kann komplett ohne Oberfläche programmiert werden und gibt, wie unten zu sehen, den momentanen Zustand aus
- **Zusätzlich kann eine Oberfläche ähnlich wie das Bild rechts erstellt werden und eine Überprüfung implementiert werden, ob alle Boxen auf einem Zielfeld stehen**
- Es soll weiterhin möglich sein mittels Undo- (Z) und Redo- (Y) Funktionalität die Bewegungen des Spielers wieder rückgängig zu machen bzw. erneut auszuführen
- Das System soll in einem Test, anhand von 3 Bewegungen und zwei Undo-Bewegungen verifiziert werden

0 Move(s)

```
######
#    #
# #P #
# x* #
# +* #
#    #
######
```



1 Move(s)

```
######
#   P #
# #   #
# x* #
# +* #
#    #
######
```

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