Technische Universität Berlin
Institut für Softwaretechnik und Theoretische Informatik
FG Softwaretechnik
Franklinstr. 28/29
10587 Berlin

Helke
Mertgen
Dobrev

MPGI 3

Muster-Klausur B

Wintersemester 2008/2009
19. Februar 2009

Prüfen Sie zunächst, ob dieses Exemplar vollständig ist (7 beidseitig bedruckte Blätter).

Tragen Sie auf diesem Titelblatt und darüber hinaus auf allen Blättern, die Sie für Ihre Niederschrift verwenden, Ihren Namen und Ihre Matrikelnummer ein. Zum Bestehen der Klausur sind mindestens **25 Punkte** notwendig. Insgesamt sind 50 Punkte möglich. Die Bearbeitungszeit beträgt 75 Minuten. In der Klausur sind außer einem beschriebenen DIN-A4-Blatt **keine Hilfsmittel** zugelassen. Verwenden Sie ausschließlich das ausgeteilte Klausur-Papier. Es dürfen nur **permanent-schwarze oder -blaue Stifte** zum Lösen der Aufgaben verwendet werden.

Viel Erfolg!

Name, Vorname: _____

Matrikelnummer: _____

Studienrichtung: _____

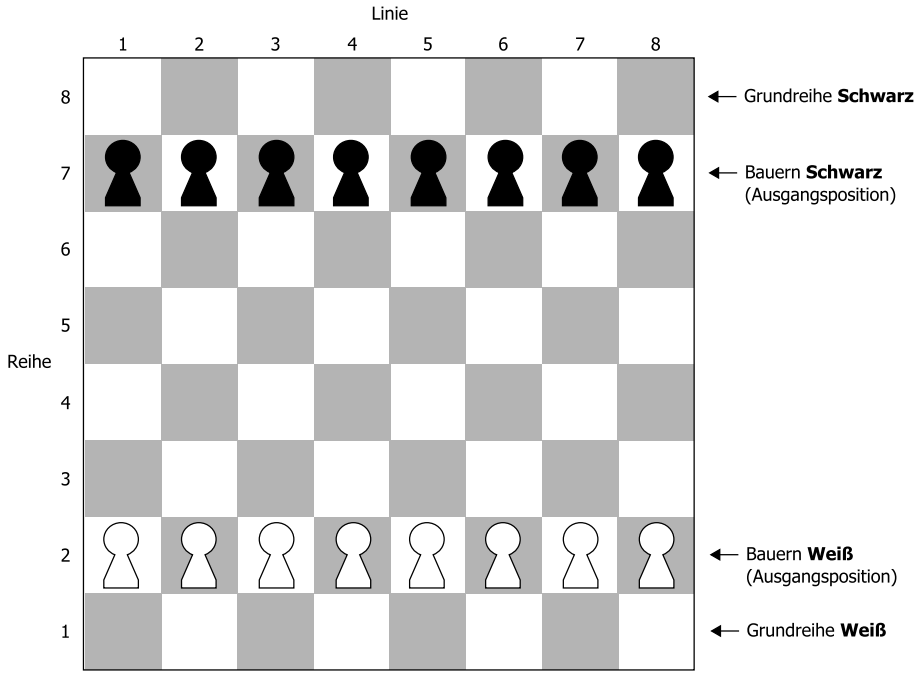
MPGI3-Übungen habe ich im _____ besucht.
(z.B. WS 2008/09)

Aufgabe	1a	1b	1c	2	Gesamt
Maximal:	10	6	14	20	50
Erreicht:					

Aufgabe 1 (Object-Z) - Bauernschach

Bauernschach ist eine vereinfachte Variante des Schachs, bei dem ausschließlich die Bauern auf dem Spielbrett stehen.

Beim Bauernschach spielen zwei Spieler der Farben *Weiß* und *Schwarz* gegeneinander. Das Spielbrett besteht aus 64 Feldern, angeordnet in 8 waagerechten Reihen und 8 vertikalen Linien. Jeder Spieler hat jeweils 8 Spielfiguren - die *Bauern* -, die zu Beginn in ihrer Ausgangsposition (wie in der Grafik veranschaulicht) aufgestellt werden. Die Spieler machen abwechselnd Züge, bei denen jeweils ein eigener Bauer nach bestimmten Regeln bewegt werden kann. In einigen Fällen dürfen dabei gegnerische Bauern geschlagen werden. Es gewinnt der Spieler, dem es als ersten gelingt, alle gegnerischen Bauern zu schlagen oder mit einem seiner Bauern ein Feld der gegnerischen Grundreihe zu erreichen; Reihe 1 ist die Grundreihe von Spieler *Weiß*, Reihe 8 die Grundreihe von Spieler *Schwarz*.



Globale Definitionen:

$Nachricht ::= Weiß\_gewinnt \mid Schwarz\_gewinnt \mid Weiterspielen$
$Farbe ::= Weiß \mid Schwarz$
$Linie ::= 1..8$
$Reihe ::= 1..8$
$Feld == Linie \times Reihe$
$linie == first$
$reihe == second$

Aufgabenstellung:
Spezifizieren Sie Bauernschach in Object-Z, indem Sie die vorgegebenen Klassendefinitionen ergänzen.

a) Bauer

10 Punkte

Jeder Bauer hat eine konstante Farbe und gehört zu einem Spiel. Der Bauer ist *imSpiel*, solange er noch eine Position auf dem Spielbrett einnimmt; *feld* ist in diesem Fall seine aktuelle Position. Diese Zustandsinformationen werden aus dem Zustand der Klasse *Spiel* von Seite 7 abgeleitet.

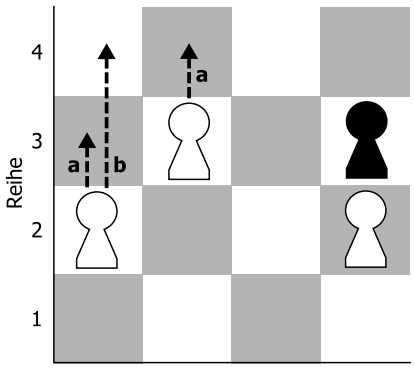
Initial befinden sich Bauern in ihrer Ausgangsposition auf dem Spielbrett.

Beachten Sie im Folgenden, dass die Operationen des Bauern keine Zustandsveränderung bewirken, sondern zunächst nur das Ziel einer Bewegung bzw. eines Schlags identifizieren.

Bauern bewegen sich stets vorwärts, auf die gegnerische Grundlinie zu. Es dürfen sich nur Bauern bewegen, die sich noch im Spiel befinden. Ein Bauer kann sich mit einem *Einfachschritt* oder einem *Doppelschritt* bewegen. Beide Operationen sollen das Zielfeld liefern, auf das der Bauer sich bewegen möchte. Sie sollen zudem sicherstellen, dass der Bauer gemäß den folgenden Regeln die Bewegung ausführen darf:

Einfachschritt: Ein Bauer kann ein Feld vorrücken, sofern das Zielfeld frei ist (d.h. nicht von einem anderen Bauern besetzt ist).

Doppelschritt: Befindet sich ein Bauer noch in seiner Ausgangsposition, dann darf er alternativ auch zwei Felder vorrücken, sofern das Zielfeld und das Feld auf dem Weg dorthin frei sind.



Beispiel: (a) Einfachschritt; (b) Doppelschritt.

Die kombinierte Operation ***Ziehen*** beschreibt, dass der Bauer sich mit einer der beiden Varianten bewegen möchte.

Bauer

$farbe : Farbe$
$spiel : Spiel$

Δ
$imSpiel : \mathbb{B}$
$feld : Feld$

$imSpiel \Leftrightarrow Self \in spiel.bauern$
$imSpiel \Rightarrow (spiel.positionen(Self) = feld)$

INIT

$imSpiel$
$farbe = Weiß \Rightarrow reihe(feld) = 2$
$farbe = Schwarz \Rightarrow reihe(feld) = 7$

EinfachSchritt

$ziel! : Feld$

$imSpiel$
$farbe = Weiß \Rightarrow reihe(ziel!) = reihe(feld) + 1$
$farbe = Schwarz \Rightarrow reihe(ziel!) = reihe(feld) - 1$
$linie(ziel!) = linie(feld)$
$ziel! \notin \mathsf{ran}\, spiel.positionen$

DoppelSchritt

$ziel! : Feld$

$INIT$
$farbe = Weiß \Rightarrow reihe(ziel!) = 4$
$farbe = Schwarz \Rightarrow reihe(ziel!) = 5$
$linie(ziel!) = linie(feld)$
$ziel! \notin \mathsf{ran}\, spiel.positionen$
$\nexists\, besetzt : \mathsf{ran}\, spiel.positionen \bullet$
$\quad (linie(besetzt) = linie(feld) \land$
$\quad farbe = Weiß \Rightarrow reihe(besetzt) = 3 \land$
$\quad farbe = Schwarz \Rightarrow reihe(besetzt) = 6)$

$Ziehen \mathrel{\widehat=} EinfachSchritt \,[]\, DoppelSchritt$

b) Bauer (Fortsetzung)

6 Punkte

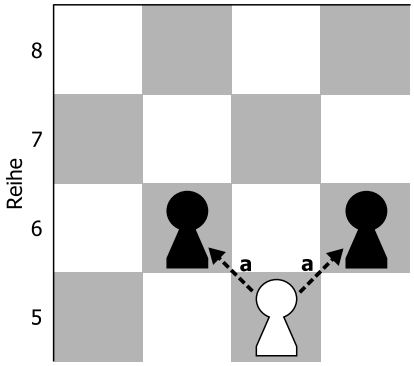
Ein Bauer kann gegnerische Bauern schlagen.

Beim Schlagen geht der Bauer mit einem *Diagonalschritt* vor; dies dürfen nur Bauern, die noch im Spiel sind. Die Operation liefert das Zielfeld, zu dem sich der Bauer bewegen möchte.

Auf dem Zielfeld der Bewegung muss ein gegnerischer Bauer stehen - dies soll mit der Operation *SchlagMöglich* geprüft werden *(andere Schlagvarianten werden in dieser Aufgabe nicht berücksichtigt)*. Zusätzlich zum Zielfeld soll die Operation auch den Bauern liefern, der geschlagen werden soll.

Diagonalschritt: Der Bauer rückt ein Feld diagonal vor (nach links oder rechts).

SchlagMöglich: Auf dem Zielfeld der Bewegung muss ein gegnerischer Bauer stehen, dies ist der zu schlagende Bauer.



Beispiel: (a) Schlagen nach links oder rechts

Die kombinierte Operation ***Schlagen*** beschreibt, dass der Bauer einen gegnerischen Bauern schlagen möchte.

Bauer(Fortsetzung)

DiagonalSchritt

$ziel! : Feld$

$imSpiel$
$(linie(ziel!) = linie(feld) + 1) \lor (linie(ziel!) = linie(feld) - 1)$
$farbe = Weiß \Rightarrow reihe(ziel!) = reihe(feld) + 1$
$farbe = Schwarz \Rightarrow reihe(ziel!) = reihe(feld) - 1$

SchlagMöglich

$ziel! : Feld$
$gegner! : Bauer$

$gegner!.farbe \neq farbe$
$(gegner! \mapsto ziel!) \in spiel.positionen$

$Schlagen \mathrel{\widehat{=}} DiagonalSchritt \land SchlagMöglich$

c) Spiel

14 Punkte

Zu einem Spiel gehört das *spielbrett* mit allen Feldern des Typs *Feld*. Die *positionen* beschreiben Felder, auf denen die Bauern stehen. *Weiß* und *Schwarz* sind jeweils abwechselnd *amZug*; am Anfang beginnt Farbe *Weiß*. *(An der Vorgabe sind keine Ergänzungen gefordert.)*

Die Operation ***BauerBewegen*** erwartet einen Bauern und ein Zielfeld als Eingabe. Die *positionen* werden gemäß der Bewegung angepasst. Nach der Operation soll der andere Spieler am Zug sein.

Die Operation ***BauerVomBrettNehmen*** erwartet einen zu schlagenden Bauern als Eingabe. Dieser Bauer soll aus den *positionen* entfernt werden.

Mit der Operation ***Auswerten*** wird der aktuelle Spielstand ausgewertet und dazu eine Nachricht ausgegeben. Hat eine Farbe die gegnerische Grundreihe erreicht oder alle gegnerischen Bauern geschlagen, so wird die Farbe zum Gewinner erklärt. Andernfalls wird weitergespielt.

Die kombinierte Operation ***Ziehen*** beschreibt, dass ein Spieler eine vollständige Bewegung mit einem seiner Bauern ausführt (ohne zu Schlagen). Der Bauer soll aus allen Bauern der Farbe, die am Zug ist, ausgewählt werden.

Die kombinierte Operation ***Schlagen*** beschreibt, dass ein Spieler einen vollständigen Schlag mit einem seiner Bauern ausführt und den geschlagenen Bauern vom Brett entfernt. Der zu ziehende Bauer soll aus allen Bauern der Farbe, die am Zug ist, ausgewählt werden.

Die kombinierte Operation ***ZugMachen*** beschreibt nun einen vollständigen Zug, wobei gezogen oder geschlagen wird, und danach der Spielstand ausgewertet wird. Den Fall, dass weder Ziehen noch Schlagen möglich sind, lassen wir komplett außer Acht.

Spiel(*Vorgabe*)

$$spielbrett : \mathbb{P}\, Feld$$
$$\#spielbrett = 64$$

$$positionen : Bauer \rightarrowtail Feld$$
$$amZug : Farbe$$
$$\Delta$$
$$bauern : \mathbb{P}\, Bauer$$
$$bauern = \mathsf{dom}\, positionen$$
$$\forall\, b : bauern \bullet b.spiel = Self$$

INIT

$$\#\{b : bauern \mid b.farbe = Weiß\} = 8$$
$$\#\{b : bauern \mid b.farbe = Schwarz\} = 8$$
$$\forall\, b : bauern \bullet b.INIT$$
$$amZug = Weiß$$

Spiel(Fortsetzung)

BauerBewegen

$\Delta(positionen, amZug)$
$bauer? : Bauer$
$ziel? : Feld$

$positionen' = (positionen \setminus \{(bauer? \mapsto bauer?.feld)\}) \cup \{(bauer? \mapsto ziel?)\}$
$amZug' \neq amZug$

BauerVomBrettNehmen

$\Delta(positionen)$
$gegner? : Bauer$

$positionen' = positionen \setminus \{(gegner? \mapsto gegner?.feld)\})$

Auswerten

$n! : Nachricht$

---

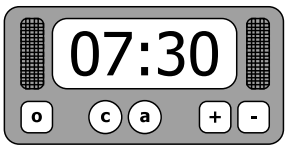
let
$GewinnerWeiß ==$
$\quad (\forall b : bauern \bullet b.farbe = Weiß) \lor$
$\quad (\exists b : bauern \bullet (b.farbe = Weiß \land reihe(b.feld) = 8));$
$GewinnerSchwarz ==$
$\quad (\forall b : bauern \bullet b.farbe = Schwarz) \lor$
$\quad (\exists b : bauern \bullet (b.farbe = Schwarz \land reihe(b.feld) = 1));$
•
$\quad GewinnerWeiß \Rightarrow n! = Weiß\_gewinnt$
$\quad GewinnerSchwarz \Rightarrow n! = Schwarz\_gewinnt$
$\quad \neg(GewinnerWeiß \lor GewinnerSchwarz) \Rightarrow n! = Weiterspielen$

$Ziehen \mathrel{\widehat=} \quad [] \; b : bauern \mid b.Farbe = amZug \bullet$
$\quad b.Ziehen \parallel BauerBewegen[b/bauer?]$

$Schlagen \mathrel{\widehat=} \quad [] \; b : bauern \mid b.Farbe = amZug \bullet$
$\quad (b.Schlagen \parallel BauerVomBrettNehmen) \fatsemi BauerBewegen[b/bauer?]$

$ZugMachen \mathrel{\widehat=} (Ziehen \; [] \; Schlagen) \fatsemi Auswerten$

Aufgabe 2 (Statecharts) - Radiowecker



Radiowecker

Ein Radiowecker besteht aus den zu steuernden Hardware-Komponenten *Radio* und *Display*. Ferner besitzt er eine interne *Uhr* zur Zeitmessung und einen *Wecker*. Der Radiowecker kann über 5 Tasten (*o*, *c*, *a*, +, -) bedient werden. Er besitzt einen normalen Betriebsmodus sowie Modi zum Einstellen von Uhrzeit und Wecker.

Radio: Im Normalbetrieb kann mit der Taste *o* das Radio angeschaltet (wenn es aus ist) und ausgeschaltet (wenn es an ist) werden. Die Radiohardware wird über die externen Ausgabeereignisse *r_an* und *r_aus* gesteuert.

Uhr: Die Uhrzeit wird in der internen Variable *t* als Anzahl der Minuten eines Tages gespeichert. Ein Tag hat 1440 Minuten. Um 0:00 Uhr ist t=0, um 23:59 Uhr ist t=1439. Eine interne Uhr soll die Zeit im Minutentakt aktualisieren.

Uhrzeit-Einstellung: Drückt man im Normalbetrieb auf die Taste *c* und hält sie gedrückt, so kann man währenddessen mit den Tasten + und - die Uhrzeit einstellen. Bei + wird *t* um eine Minute erhöht, bei - um eine Minute verringert. *t* sollte dabei immer im Bereich von 0 bis 1439 Minuten liegen (nutzen Sie dazu den modulo-Operator *mod*). Wird *c* wieder losgelassen, so kehrt der Radiowecker in den Normalbetrieb zurück. Parallel zur Zeiteinstellung darf die interne Uhr ganz normal weiterlaufen.

Wecker: Der Wecker kann aktiviert oder deaktiviert sein. Die Weckzeit ist in einer Variable *w* gespeichert (analog zu Variable *t*). Tritt der Fall ein, dass die Weckzeit bei aktiviertem Wecker gleich der aktuellen Uhrzeit ist, dann soll der Wecker das Radio anschalten, sofern es noch nicht spielt. Wenn das Radio vom Wecker eingeschaltet wurde, dann soll es nach einer Stunde automatisch wieder ausgeschaltet werden. Ein deaktivierter Wecker startet das Radio nicht. Der Wecker startet das Radio auch nicht während der Uhrzeit- oder Weckzeit-Einstellung.

Wecker-Einstellung: Drückt man im Normalbetrieb auf die Taste *a* und hält sie gedrückt, so kann man währenddessen mit den Tasten + und - die Weckzeit einstellen (analog zur Uhrzeit-Einstellung). Weiterhin kann währenddessen mit der Taste *o* der Wecker aktiviert bzw. deaktiviert werden. Wird *a* wieder losgelassen, so kehrt der Radiowecker in den Normalbetrieb zurück.

Display: Das Display soll im Normalbetrieb und während der Uhrzeit-Einstellung stets die mit der Variablen *t* beschriebene Uhrzeit anzeigen. Mit dem externen Ereignis *dt* wird die Hardware zur Anzeige der Uhrzeit von *t* im Display veranlasst. Während der Einstellung der Weckzeit soll im Display mit dem externen Ereignis *dw* stets die aktuelle Zeit von *w* angezeigt werden. Bei jeder Änderung der Variablen soll mit *dt* bzw. *dw* eine Aktualisierung der Anzeige veranlasst werden. Dies soll aber nicht häufiger als nötig geschehen.

Initial sind *t* und *w* gleich 0, das Radio ist aus und der Wecker deaktiviert.

Aufgabenstellung

20 Punkte

Modellieren Sie den beschriebenen Radiowecker mit seinen Komponenten und dem Ablauf mit Hilfe von Statecharts. Es sollen nur die beschriebenen Eigenschaften modelliert werden.
Verwenden Sie die in der folgenden Tabelle aufgeführten Ereignisse, die durch den Benutzer ausgelöst werden können. Wenn Sie neue Ereignisse für die Modellierung definieren wollen, ergänzen Sie entsprechend die Tabelle.

NAME	BESCHREIBUNG
EXTERNE EINGABEEREIGNISSE	
o **c** **ĉ** **a** **â** **+** **-**	Taste **o** wird gedrückt. Taste **c** wird gedrückt. Taste **c** wird losgelassen. Taste **a** wird gedrückt. Taste **a** wird losgelassen. Taste **+** wird gedrückt. Taste **-** wird gedrückt.
EXTERNE AUSGABEEREIGNISSE	
r_an **r_aus** **dt** **dw**	Signal zum Einschalten des Radios. Signal zum Ausschalten des Radios. Signal, um im Display die Zeit der Variable t anzuzeigen. Signal, um im Display die Zeit der Variable w anzuzeigen.
INTERNE EREIGNISSE	

Statechart-Lösung: