

Checkers Design Document

me 3

Contents

1	Introduction	1
2	Module Guide	1
2.1	Hardware Hiding Module	2
2.2	Behaviour Hiding Module	2
2.3	Software Decision Hiding Module	2
2.3.1	Piece Module	2
2.3.2	Board Module	2
3	Module Interface Specification	3
3.1	Piece Module	3
3.1.1	Interface	3
3.1.2	Implementation	3
3.2	Board Module	4
3.2.1	Interface	4
3.2.2	Implementation	5

1 Introduction

This document contains decomposition (MG and MIS), uses relationship, and traceability.

2 Module Guide

Modules are stuff.

2.1 Hardware Hiding Module

2.2 Behaviour Hiding Module

2.3 Software Decision Hiding Module

2.3.1 Piece Module

Type	Software Module
Secret	This module hides and separates specific piece information.
Responsibilities	This will hold the necessary components to describe what a game piece will contain, which will be separate from the game board.
Uses	None
Design	3.1

2.3.2 Board Module

Type	Software Module
Secret	This module serves to hide the secret of how the board is defined internally.
Responsibilities	This module is responsible for holding the necessary components and attributes to setup the board and describe piece locations.
Uses	2.3.1
Design	3.2

3 Module Interface Specification

3.1 Piece Module

3.1.1 Interface

Types	
typeState	enumerate if the piece is normal or king
player	enumerate if piece owned by Black or White
Constants	
None	
Access Programs	
getType()	Retrieves the piece's current type.
setType(newType : typeState)	Changes the piece's type.
getOwner()	Says who owns the piece.

3.1.2 Implementation

Variables	
pieceType : typeState	holds current piece type
owner : player	holds information of the piece's owner
Access Programs	
getType() : typeState	
Inputs	None
Outputs	pieceType
Updates	None
setType(newType : typeState)	
Inputs	newType
Outputs	None
Updates	pieceType
getOwner() : player	
Inputs	None
Outputs	owner
Updates	None

3.2 Board Module

3.2.1 Interface

Types	None
Constants	None
Access Programs	
clear()	Removes all pieces from the board.
placePiece(col : int, row : int, piece : Piece)	Places the piece on the board while checking if the placement is legal (in terms of checkers).
movePiece(fromCol : int, fromRow : int, toCol : int, toRow : int)	Moves the piece from starting to end positions while checking if the movement is valid (in terms of checkers).
getPiece(col : int, row : int)	This method is used to determine if a piece exists on a square of the board. If the piece does exist, we pass it along to the caller.

3.2.2 Implementation

Types	None	
Constants	None	
Variables	pieceArray[]	The board will be implemented as an array.
Access Programs	clear()	
	Inputs	None
	Outputs	pieceArray[]
	Updates	None
	placePiece(col : int, row : int, piece : Piece)	
	Inputs	col, row, piece
	Outputs	pieceArray[]
	Updates	None
	movePiece(fromCol : int, from-Row : int, toCol : int, toRow : int)	
	Inputs	None
	Outputs	None
	Updates	None
	getPiece(col : int, row : int) : Piece	
	Inputs	col, row
	Outputs	piece
	Updates	None