GameConfigSelector

«extends JFrame»
-themeGroup: ButtonGroup
-difficultyGroup: ButtonGroup
-startButton: JButton

+main(String[]): void

uses

MemoryGameGUI

«extends JFrame»
-game: Game
-boardPanel: BoardPanel
-resetButton: JButton

+updateTitle(): void
+keyPressed(KeyEvent): void

has-a

BoardPanel

«extends JPanel»
-game: Game
-rows: int
-cols: int

+paintComponent(Graphics): void
+mouseClicked(MouseEvent): void

has-a

has-a

Game

-deck: Deck
-firstSelected: Card
-secondSelected: Card
-busy: boolean
-isHard: boolean
-moveCount: int

+selectCard(Card): void
+flipBackSelections(): void
+isGameOver(): boolean
+restart(theme, isHard): void

has-a

has-a

Deck

-cards: List<Card>

+shuffle(): void
+getCards(): List<Card>

has-a

Card

-value: int
-faceUp: boolean
-matched: boolean
-frontImage: Image
-coverImage: static Image

+getValue(): int
+flip(): void
+isMatched(): boolean
+draw(Graphics2D, int, int, int, int): void

UML Diagram Description

This UML diagram shows the main structure of the memory matching game.

1. Card stores the value, image, and status (flipped or matched) of a card. It can flip itself and be drawn on the screen.

2. Deck creates and holds a list of 20 cards (10 pairs). It can shuffle and return the cards.

3. Game controls the game logic, such as selecting cards, checking matches, counting moves (in hard mode), and restarting the game.

4. BoardPanel is where the cards are displayed. It handles mouse clicks to flip cards and shows the game board.

5. MemoryGameGUI is the main game window. It includes the board, a reset button, and listens for keyboard input like pressing "R" to restart.

6. GameConfigSelector is the first window that lets the player choose a theme and difficulty before the game starts.

Relationships:

1. GameConfigSelector creates MemoryGameGUI

2. MemoryGameGUI contains Game and BoardPanel

3. Game has a Deck, which has many Card objects

4. This setup keeps the game logic, user interface, and card drawing separated, making the code easier to understand and manage.