**1. startGame()**

**Pre-conditions**

* 애플리케이션이 기동된 상태이다.

**Post-conditions**

* “게임 시작” 메뉴가 UI에 표시된다.

**Main Flow**

1. 사용자가 메인 메뉴에서 “게임 시작”을 선택한다.
2. 시스템이 내부 상태(Game, Player 리스트, Board 등)를 초기화한다.
3. UI를 “참가자 설정” 화면으로 전환한다.

**2. selectUI(mode)**

**Pre-conditions**

* startGame()이 완료되어 UI 선택 화면에 진입한 상태이다.

**Post-conditions**

* 선택한 UI 모듈이 메모리에 로딩되고, 다음 화면으로 전환 준비가 완료된다.

**Main Flow**

1. 시스템이 지원하는 UI 목록(SWING, JAVAFX)을 표시한다.
2. 사용자가 원하는 mode를 선택한다.
3. 선택된 UI 모듈(YutGameUI 구현체)을 로딩하고 초기화한다.
4. UI가 “보드 레이아웃 선택” 화면으로 전환된다.

**Exception Flow**

* 사용자가 유효하지 않은 mode를 선택하면(예: null 또는 지원하지 않는 값)
  + 에러 메시지를 표시하고 2번으로 돌아간다.

**3. selectBoardLayout(layoutType)**

**Pre-conditions**

* UI가 정상 로딩된 상태이다.

**Post-conditions**

* GameController.selectedBoardShape에 layoutType이 저장되고,
* 내부 Board가 해당 궤도(TRADITIONAL/PENTAGON/HEXAGON)로 초기화된다.

**Main Flow**

1. 시스템이 사용 가능한 보드 모양 목록을 표시한다.
2. 사용자가 layoutType을 선택한다.
3. GameController가 선택값을 저장하고, Board를 해당 모양으로 생성한다.
4. UI가 “참가자 설정” 화면으로 전환된다.

**Exception Flow**

* 사용자가 null 또는 지원하지 않는 layoutType을 선택하면
  + 에러 메시지를 표시하고 1번으로 돌아간다.

**4. setupParticipants(playerNum, pieceNum)**

**Pre-conditions**

* 보드 레이아웃이 설정된 상태이다.

**Post-conditions**

* Game.players 리스트에 playerNum명의 Player 객체가 생성되어 저장된다.
* 각 Player.pieces에 pieceNum개의 Piece가 초기 위치(OFFBOARD)에 생성·연결된다.

**Main Flow**

1. 시스템이 “참가자 수(2-4)”, “이름”, “말 개수(2-5)” 입력 폼을 표시한다.
2. 사용자가 playerNum과 각 Player의 name, pieceNum을 입력한다.
3. 유효성 검사를 통과하면 각 Player와 Piece를 초기화한다.
4. UI가 “게임 진행” 화면으로 전환된다.

**Exception Flow**

* playerNum 또는 pieceNum이 허용 범위를 벗어나면
  + 경고 메시지를 표시하고 1번으로 돌아간다.
* 이름이 빈 문자열이거나 중복되면
  + 경고 메시지를 표시하고 해당 입력을 다시 받는다.

**5. throwYut(player)**

**Signature**

**Pre-conditions**

* 현재 턴(Game.currentPlayer)인 player가 존재하며,
* checkGameEnd()가 false인 상태이다.

**Post-conditions**

* 무작위 또는 지정된(YutThrower.throwSpecified) 윷 결과가 반환되고,
* GameController.currentTurnThrows에 결과가 저장된다.

**Main Flow**

1. 사용자가 “랜덤 던지기” 또는 “지정 던지기”를 선택한다.
2. 시스템이 YutThrower.throwRandom() 또는 throwSpecified(...) 호출해 YutThrowResult 생성.
3. 결과를 UI에 표시하고 반환한다.

**6. movePiece(yutResult, piece)**

**Pre-conditions**

1. currentTurnThrows에 yutResult가 마지막으로 저장되어 있다.
2. piece.owner == Game.getCurrentPlayer()
3. piece.isFinished() == false

**Post-conditions**

* piece.position이 계산된 newPos로 변경되고,
* 잡기(CAPTURE)·묶기(GROUP)·도착(FINISH) 등 상태가 반영된다.
* 반환값(MoveOutcome)이 다음 중 하나이다:
  + SUCCESS, CAPTURE, GROUP, FINISH

**Main Flow**

1. steps = yutResult.getMove()
2. path = PathManager.getNextPositions(piece, steps, selectedBoardShape) 계산
3. newPos = path.last()
4. Board.placePiece(piece, newPos) 호출
   * 내부에서 잡기/묶기/종료 로직 처리
5. 이동 결과에 따라 MoveOutcome 반환

**Exception Flow**

* yutResult == BACKDO(빽도)이고 piece.position == OFFBOARD인 경우
  + 이동 없이 MoveOutcome.FAIL 반환

**7. checkExtraTurn(result)**

**Pre-conditions**

* result가 movePiece 전에 currentTurnThrows에 저장된 상태이다.

**Post-conditions**

* result가 YUT 또는 MO이면 pendingExtraTurns가 1 증가,
* 그 외엔 0으로 초기화된다.

**Main Flow**

1. if result == YUT || result == MO
2. pendingExtraTurns += 1  
   else  
    pendingExtraTurns = 0

**8. checkGameEnd()**

**Pre-conditions**

* 매 턴 이동(movePiece) 이후 호출된다.

**Post-conditions**

* 모든 Player.pieces가 isFinished() == true이면 true, 아니면 false 반환

**Main Flow**

1. for each Player in Game.players
2. if player.hasFinishedAllPieces() == false return false
3. return true

**9. postGameChoice()**

**Pre-conditions**

* checkGameEnd()가 true를 반환하여 승패가 결정된 상태이다.

**Post-conditions**

* 사용자가 선택한 동작이 반환된다:
  + RESTART → 내부 상태(Game, Board, Player 등) 초기화
  + EXIT → 프로그램 종료

**Main Flow**

1. UI에 “다시 시작” / “종료” 버튼을 표시한다.
2. 사용자가 버튼을 클릭한다.
3. 선택된 PostGameAction을 반환한다.
4. 호출 쪽에서 반환값에 따라
   * RESTART → startGame() 재호출
   * EXIT → 애플리케이션 종료