SpielStein
− farbe : int − img : Image − blackStone : static Image − whiteStone : static Image
+ getColor() : final int + getImage() : final Image

GewinnerController
− gewinnerPane : AnchorPane − abbrechenButton : Button − neuButton : Button − gewinneriView :ImageView − gewinnerText : Text − spielController : SpielController − gewinnerStage : Stage
+ handleAbbrechenButton(event : ActionEvent) : void + handleNeuButton(event : ActionEvent) : void + setGewinnerImage(image Image) : void + setDialogStage(gewinnerStage : Stage) : void + setGewinnerText(s : String) : void + setDialogSpielController(spielController : SpielController):void

Brett
− _dim : int − _spieler : int − _brett : SpielStein[ ][ ] − _gitterVert : List<Line> − _gitterHorz : List<Line> − _gitter : List<Line> − _SpielZuege : List<SpielZug> − _gitterWeite : double − _randX : double − _randY : double − _CheckAdjacent : boolean
+ Brett(dim : int, x : double, y : double) + redrawGitter(x : double, y : double) : void + roundCoord(x : double, y : double) : double[ ] + steinAt(int x, int y) : SpielStein + steinAt(double x, double y) : SpielStein − steinSet(int x, int y, SpielStein s) : boolean + makeMove(SpielZug zug) : boolean + printMoves() : void + getNextMoveColour() : int + List<SpielZug> getSpielZuege() : final + getDim() : int + getGitter() : List<Line> + getGitterWeite() : double + getRandX() : double + getRandY() : double + getSpieler() : final int + getBrett() : final SpielStein[ ][ ]

+ static SpielZug
+ x, y : int + stein:SpielStein + iView:ImageView
+ SpielZug(x:int, y:int, stein:SpielStein, iView:ImageView) + toString():String

Options

− _menge : HashSet<Tupel>

+ Options()
+ setOption(name : String , objekt : Object) : void
+ getOption(name : String) : Object
+ printOption(name : String) : void
+ toString() : String

-Tupel

+ name : String
+ objekt : Object

+ Tupel(name : String, objekt : Object)
+ hashCode() : int
+ equals(Object obj) : boolean
− getOuterType() : Options
+ toString() : String

Main

+ optionen : static Options
+ primaryStage : static Stage

+ start(primaryStage : Stage) : void
+ main(args : String[]) : static void

SpielAI

− _brett : Brett
− _possibleMoves : ArrayList<LinkedHashSet<Savegame>>

+ SpielAI(brett : Brett)
+ generateNextMoves() : void
+ updateMoves() : void
+ getBestMoves() : Integer[ ][ ]
+ addDoubleArray(a : Double[ ][ ], b : Double[ ][ ]) : static Double[ ][ ]
+ multDoubleArray(a : Double[ ][ ], f : double) : static void
+ twoDeepCloneDouble(a : Double[ ][ ]) : static Double[ ][ ]
+ printDoubleArray(a : Double[ ][ ]) : static void

Savegame

− moveNr : int
− steine : int[ ][ ]
− spielerAnz : int
− dim : int
− nextMove : int[ ]

+ Savegame(brett : Brett)
+ generateNextMoves() : LinkedHashSet<Savegame>
+ generateHeuristic() : Double[ ][ ]
+ hashCode() : int
+ toString() : String
+ equals(obj : Object) : boolean

SpielController

- mitteBeginnCheckBox : CheckBox
- backgroundImage : ImageView
- tabPaneSwitch : TabPane
- ueberTab : Tab
- anlegenCheckBox : CheckBox
- brettGroesseLabel : Label
- einstellungenAnchorPane : AnchorPane
- gameAnchorPane : AnchorPane
- aiCheckBox : CheckBox
- helpTab : Tab
- gameTab : Tab
- brettGroesseTextField : TextField
- zweiSpielerButton : RadioButton
- stoneImage : ImageView
- brettGroesseBox : ComboBox<String>
- bild2Button : ToggleButton
- einstellungenTab : Tab
- anzahlReiheTextField : TextField
- bild1Button : ToggleButton
- einSpielerButton : RadioButton
- aiButton : RadioButton
- hilfeText : TextArea
- uberText : TextArea
- neuButton : Button
- spielStartenButton : Button
- startButton : Button
- newGameButton : Button
- pauseGameButton : ToggleButton
- zuruecksetzenButton : Button
- wrapAnchorPane : AnchorPane
- aiSpeedSlider : Slider
- radioButtonGroup : final ToggleGroup
- bildGroup : final ToggleGroup
- choiceBoxOptions : ObservableList<String>
- spielbrett : Brett
- gameDone : boolean
- s : SpielStein
- currWidth, currHeight : double
- lastPlayed : ImageView
- winningStone : List<ImageView>
- gegner : SpielAI
- lastTime : static long
- aiPaused : boolean
- zweiAiTimer : AnimationTimer

- initialize() : void
- standardEinstellungen() : void
+ bildeBrett() : void
- handleSpielerAnzahlButton(ActionEvent event) : void
- handleBrettGroesseBox(ActionEvent event) : void
- handleBrettGroesseFeld(ActionEvent event) : void
- handleSpielregeln(ActionEvent event) : void
- handleBackground(ActionEvent event) : void
+ neustart() : void
- handleSpielStartenButton() : void
- handleZuruecksetzenButton(ActionEvent event) : void
- handleStartButton() : void
- disable() : void
- enable() : void
- handleNewGameButton() : void
- handlePauseGameButton(ActionEvent event) : void
- handleMouseMoved(MouseEvent event) : void
- handleSizeChanged() : void
- handleSizeChanged(boolean forceIt) : void
- updatePlayMarkers() : void
- handleDragDetected(MouseEvent event) : void
- handleMouseClicked(int x, int y) : void
- handleMouseClicked(MouseEvent event) : void
- letAImakeMove() : void
- handleGewinner() : boolean
- handleGewinner(boolean unentschieden) : boolean
- checkIfGewinner() : boolean
- handleKeyPressed(KeyEvent event) : void
- handleKeyReleased(KeyEvent event) : void

getBestMoves():

```
 1: h ← generateHeuristic()
 2: max ← −∞
 3: n ← 0
 4: for i ∈ {0, dim(h)} do
 5:     for j ∈ {0, dim(h)} do
 6:         if max < h_{i,j} then
 7:             max ← h_{i,j}
 8:             n ← 1
 9:         else if max = h_{i,j} then
10:             n + +
11:         end if
12:     end for
13: end for
14: erg :∈ ℕ^{n×2}
15: for i ∈ {0, dim(h)} do
16:     for j ∈ {0, dim(h)} do
17:         if  max = h_{i,j}  then
18:             n ← n − 1
19:             erg_{n,0} ← i
20:             erg_{n,1} ← j
21:         end if
22:     end for
23: end for
24: return erg
```