Software-Architektur Edgy

Stephanie Brogli, Rüthemann Peter

19.05.2019

Part I
Einleitung

In diesem Dokument finden Sie einen Überblick über die Architektur und Struktur vom Spiel Edgy.

Hier noch eine kurze Einführung ins Spiel und seine Regeln (aus dem Handbuch), für den Fall, dass Sie es noch nicht kennen sollten:

Spielfläche

Spielbrett: Links oben hast du das Spielbrett mit einer Fläche von 20x20 Quadraten. Dort kannst du deine Spielsteine ablegen, wenn du am Zug bist.

Spielsteine: Links unten sind all die Spielsteine, die du noch übrig hast. Du kannst jeden Spielstein nur einmal legen. Entweder verschiebst du den Stein per Drag and Drop auf das Board oder du kannst ihn per Mausklick einfügen. Dafür musst du zuerst den Stein deiner Wahl anwählen, indem du ihn mit der linken Maustaste anklickst. Danach kannst du ihn mit den enstprechenden Buttons flippen oder drehen. Nun musst du kurz schauen, welches Quadrat die Mitte des Spielsteines ist (Siehe Anhang) und auf dem Spielbrett das Quadrat, auf dem die Mitte des Spielsteines liegen sollte, wieder mit der linken Maustaste anklicken.

Tabelle: Rechts oben ist die Liste mit allen Spielern, die das gleiche Match spielen wie du. Also all die Leute, die es zu besiegen gilt. In der ersten Spalte dieser Tabelle stehen die Ränge. Allerdings handelt es sich hierbei nicht um den aktuellen Rang im Spiel, sondern um den Gesamtrang im Vergleich zu allen Spielern, die sich jemals eingeloggt haben! In der zweiten Spalte stehen die Spielernamen und in der dritten siehst du deinen aktuellen Punktestand aus dem laufenden Spiel. Dabei gilt: ein Quadrat gibt einen Punkt.

Buttons: Rechts in der Mitte sind die Buttons, mit denen du einen Stein beliebig oft flippen und drehen kannst. Falls du dich im Spiel noch nicht zurecht findest (wofür du dich nicht schämen musst), findest du einen

Button mit einem Tutorial, das dir alles Wichtige in Kürze näherbringt. Ausserdem findest du dort noch den absoluten Highlight Button, den "hint", der dir das Leben retten kann: Wenn du nicht mehr weiterweisst, kannst du ihn um Hilfe bitten. Er wird dir einen möglichen Spielzug vorschlagen. Nebenbei siehst du mitten drin auch das schwarze Schaf: Den Button "surrender". Diesen kannst du benutzen, wenn du keine Lust hast, weiterzuspielen (unmögliche Situation) oder wenn du keinen Stein mehr legen kannst. Du scheidest dann vom Spiel aus und musst dich gedulden, bis die anderen Spieler ebenfalls ausgeschieden sind.

Chat: Rechts unten ist der Chat. Er bietet dir die Möglichkeit, mit deinen Mitspielern zu kommunizieren.

Spielverlauf

Die Spieler sind der Farbe nach (gelb-rot-blau-grün) einmal pro Runde an der Reihe mit Legen. Jeder darf pro Runde maximal einen Stein ablegen. Der rote Pfeil zeigt dir an, wer gerade an der Reihe ist. Du kannst aus allen deinen verbleibenden Spielsteinen einen wählen. Auch wenn du nicht an der Reihe bist, kannst du deine Steine schon drehen und flippen. Nur legen kannst du dann nicht. Das Spiel ist beendet, wenn keiner mehr legen kann oder alle aufgegeben haben.

Punkte

Sieger wird der- diejenige mit den meisten Punkten. Ein Quadrat eines abgelegten Steines gibt einen Punkt.

Regeln

- Du musst den ersten Spielstein in eine noch freie Ecke des Spielbretts legen.
- Danach dürfen sich nur noch Ecken deiner Spielsteine berühren nicht aber die Kanten (Es muss mindestens eine Ecke, egal welche, berührt werden).
- Du darfst jedoch sowohl Kanten als auch Ecken der Steine anderer berühren.
- Der ganze Spielstein muss auf dem Spielbrett liegen.
- Die Spielsteine dürfen sich nicht überlappen.
- Du darfst pro Runde nur einen Stein ablegen.
- Du darfst jedoch vor dem Legen jeden Stein beliebig oft drehen und flippen.
- Du kannst jederzeit "surrender" drücken um nicht mehr weiterzuspielen, danach kannst du jedoch nicht mehr ins gleiche Spiel zurückkehren und musst warten, bis alle zu Ende gespielt haben.

Ziel Ziel ist es, so viele Quadrate (aus denen die Spielsteine bestehen) wie möglich auf dem Spielbrett abzulegen und die anderen gleichzeitig so gut wie möglich zu blockieren

Part II
Architektur

Edgy ist ein online Multiplayer Puzzlespiel, das folgende Anforderungen stellt:

- Mindestens Java 8.0. Java 11 empfohlen
- Windows, MacOS, Linux
- Mindestens 4 Spieler
- Netzwerk
- Persistenter Highscore
- Rundenbasiert

Um diese Vorgaben zu erfüllen war es nötig Anforderungen im Bereich Usermanagment, Netzwerkkommunikation und Qualitymanagment zu erfüllen. Die Software-Architektur wird somit im folgenden im Detail erklärt.

1 Usermanagment

Damit keine Daten verloren gehen, wird ein Player, das heisst ein User mit einem unique name, password und score in einer SQLite Datenbank abgespeichert, sobald er sich registriert hat. Diese Daten können mit einem Login jederzeit wieder abgerufen werden. Ein Feature, dass die Spieler anspornen soll, da sie ihre Punkte und somit den Fortschritt im Spiel sehen können. Da das Usermanagement eine externe Ausführung, ausserhalb des richtigen Spieles ist, wurde es in ein eigenes Package gelegt. Sie finden es unter dem Namen UserManagement. Der Code in diesem Packet wurde wiederum in drei Klassen aufgeteilt, hauptsächlich der Übersicht halber und um einen zeitgleichen Zugriff von zwei verschiedenen Player zu gewährleisten.

Metrix: Eine Klasse, die die ganzen Informationen eines Spielers als Variablen speichert. Es handelt sich dabei um einen username und eine userid, die beide unique sind und so sicherstellen, dass es zu keiner Verwechslung kommen kann. Des weiteren speichert die Klasse den score, also Punktestand, das Passwort, den Online Status und den Rang auf der Highscoreliste. Es ermöglicht eine schnelle Identifizierung des Spielers.

SQLServer: Eine Klasse, die alle internen SQLite Befehle ausführt. Sie speichert und verändert die Daten bei Gebrauch. Zum Beispiel speichert sie neue Spieler in der Datenbank. Diese Klasse hat direkten Zugriff auf die Daten.

UserManagement: Eine Klasse, die Befehle an den SQLServer weiterleitet und ihn auffordert, die gewünschten Daten zu bearbeiten. Die Klasse ist nur zur Weiterleitung von Serverbefehlen gedacht und hat keinen direkten Zugriff auf die Daten. Von dieser Klasse aus kann die Datenbank nicht verändert werden. Eine solche Struktur verhindert unerwünschte Datenmanipulierung oder einen Absturz der Datenbank.

2 Netzwerkkommunikation

Um ein rundenbasiertes Multiplayer-Spiel zu ermöglichen, ist ein Netzwerkprotokoll nötig. Edgy verwendet ein String basiertes Protokoll. Da es sich um ziemlich viele Klassen handelt, ist eine grobe Sturktur in Form von Packages angebracht:

Server: Dieses Package bildet das erste "Antriebsrad" der ganzen Software. Der Server ist einer der beiden Hauptpfeiler der Kommunikation. Er aktualisiert fast permanent den Userstatus, also den score und online Status. Folglich sendet er wichtige Nachrichten an den Client und die GUI. Seine zweite Hauptfunktion besteht darin, alles was ein User macht, zu überprüfen. Er testet jeden Zug, jedes Login, jede Registrierung auf seine Richtigkeit. Ausserdem sendet er einen Ping an den Client.

Client: Kurz zusammengefasst könnte man sagen, das der Client die Anfragen macht und diejenigen Befehle ausführt, die vom Server als korrekt bestätigt wurden. Wie zum Beispiel ein Spielstein zu legen oder einen Benutzer auszuloggen. Nebenbei werden die Befehle in der Klasse Alias geparst, also in lesbare Commands umgewandelt. Dieses Package ist somit der zweite Hauptpfeiler der Netzwerkkommunikation.

Zwischen Server und Client werden somit permanent Daten gesendet und empfangen. Allerdings nicht auf direktem Weg, sondern über geparste Befehle, was die Sicherheit und Stabilität der Verbindung verbessert.

3 JUnit

JUnit wurde zur Überprüfung auf anfällige Fehler implementiert und ausgeführt. Für dieses Projekt wurde lediglich die Spiellogik getestet. Es handelt sich um einen Zweig, der sich nicht in der Struktur des Spiels befindet und daher als extern bezeichnet werden kann. Ein weiteres Packet "Edgy" wurde erstellt. In ihm befinden sich die beiden Testklassen, die es ermöglichen, jegliche gewünschte Methode auf einen reibungs und fehlerlosen Durchlauf zu testen.

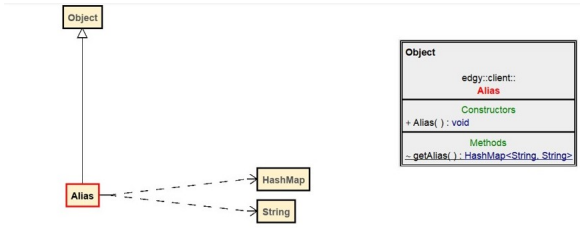
BoardTest: Es handelt sich um eine Klasse, in der die einzelnen Methoden getestet werden, die zum Ablegegen eines Spielsteines nötig sind. Besonders wurden die check-Methoden manuell durchgespielt und mit einem Beispiel abgeglichen. Diese Methoden bilden den Kern der Spielmechanik, da sie die Spielregeln bezüglich eines Zuges überprüfen.

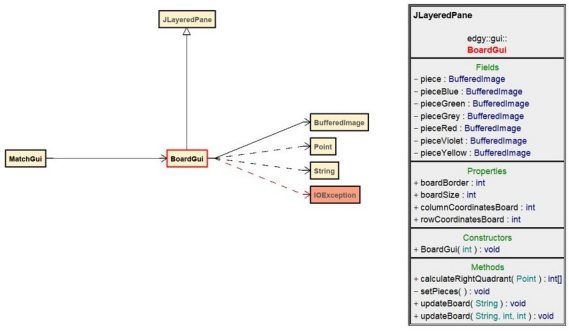
GameplayTest: Hierbei handelt es sich um eine Klasse, die ein ganzes Spiel, aus Sicht der Spiellogik, prüft. Die abgelegten Steine aller vier Spieler, zuerst einzeln, dann zusammen, wurden getestet. Zu guter Letzt, wurde noch ein Test bezüglich der Spielerpunkte nach einem Spiel, durchgeführt.

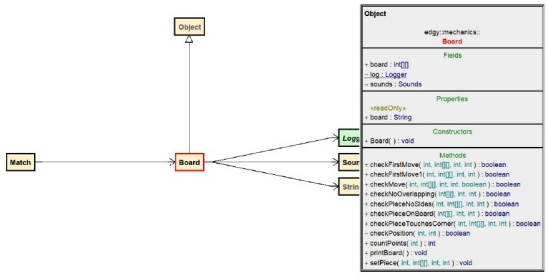
Part III
UML-Klassendefinitionen

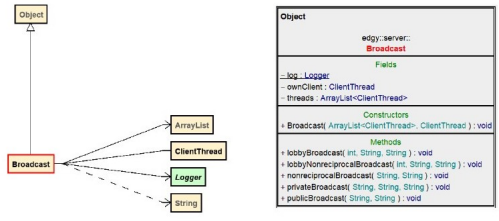
Einen detaillierten Einblick in die Software-Architektur liefern UML-Diagramme und die dazugehörenden Vektorgraphiken, die Abhängigkeiten mit anderen Klassen ausdrücken. Die Software ist in 8 Pakete aufgeteilt:

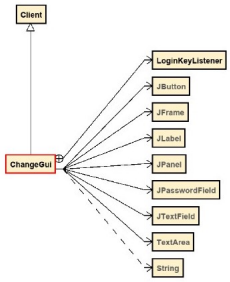
- Client: Hier startet der Client und empfängt Informatinen vom Server. Der Client startet ebenfalls abhängig vom Input die richtigen GUI-Fenster
- Gui: Edgy nutzt die Bibliothek Swing zur Darstellung der graphischen Benutzeroberfläche. Jedes Fenster bestitzt eine eigene Klasse
- Server: Der Server von Edgy regelt den Datenfluss zwischen Server und Client. Der Client alleine entscheidet über Spielentscheide und die Richtigkeit von Spielzügen.
- Mechanics: Das Herzstück von Edgy berechnet Spielzüge voraus und bestimmt ob die vom User gewünschten Züge überhaupt erlaubt sind.
- Ping: Diese Klassen sind verantwortlich, dass der Server als auch der Client registriert ob die Verbindung noch vorhanden ist.
- Sound: Der SoundThread steuert die gezielte Soundausgabe.
- Usermanagement: Alle Spielerdaten werden peristent in einer SQL-Datenbank gespeichert.
- Start: Edgy wird über eine Start-Datei gestartet.


Object
Alias
HashMap
String
Object
edgy::client::
Alias
Constructors
+ Alias( ) : void
Methods
~ getAlias( ) : HashMap<String, String>


JLayeredPane
MatchGui
BoardGui
BufferedImage
Point
String
IOException
JLayeredPane
edgy::gui::
BoardGui
Fields
- piece : BufferedImage
- pieceBlue : BufferedImage
- pieceGreen : BufferedImage
- pieceGrey : BufferedImage
- pieceRed : BufferedImage
- pieceViolet : BufferedImage
- pieceYellow : BufferedImage
Properties
+ boardBorder : int
+ boardSize : int
+ columnCoordinatesBoard : int
+ rowCoordinatesBoard : int
Constructors
+ BoardGui(int) : void
Methods
+ calculateRightQuadrant(Point) : int[]
- setPieces( ) : void
+ updateBoard(String) : void
+ updateBoard(String, int, int) : void


Object
Match
Board
Object
edgy::mechanics::
Board
Fields
+ board : int[][]
- log : Logger
- sounds : Sounds
Properties
«readOnly»
+ board : String
Constructors
+ Board( ) : void
Methods
+ checkFirstMove(int, int[][], int, int) : boolean
+ checkFirstMove1(int, int[][], int, int) : boolean
+ checkMove(int, int[][], int, int, boolean) : boolean
+ checkNoOverlapping(int[][], int, int) : boolean
+ checkPlacesNSides(int, int[][], int, int) : boolean
+ checkPieceOnBoard(int[][], int, int) : boolean
+ checkPieceTouchesCorner(int, int[][], int, int) : boolean
- checkPosition(int, int) : boolean
+ countPoints(int) : int
+ printBoard( ) : void
+ setPiece(int, int[][], int, int) : void


Object
Broadcast
ArrayList
ClientThread
Logger
String
Object
edgy::server::
Broadcast
Fields
- log : Logger
- ownClient : ClientThread
- threads : ArrayList<ClientThread>
Constructors
+ Broadcast(ArrayList<ClientThread>, ClientThread) : void
Methods
+ lobbyBroadcast(int, String, String) : void
+ lobbyNonreciprocalBroadcast(int, String, String) : void
+ nonreciprocalBroadcast(String, String) : void
+ privateBroadcast(String, String, String) : void
+ publicBroadcast(String, String) : void



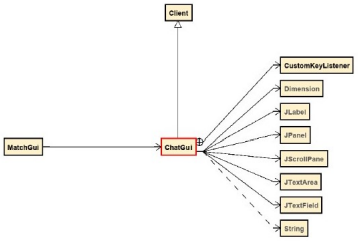
Client

edgy::gui::
ChangeGui

Fields
- buttonPanel : JPanel
- change : boolean
~ changeButton : JButton
- controlPanel : JPanel
- headerLabel : JLabel
+ mainFrame : JFrame
~ passwordText : JPasswordField
- textArea : TextArea
- userText : JTextField

Constructors
+ ChangeGui(String) : void

Methods
+ buttonListener( ) : void
- send(String) : void
+ setText(String) : void
+ show( ) : void



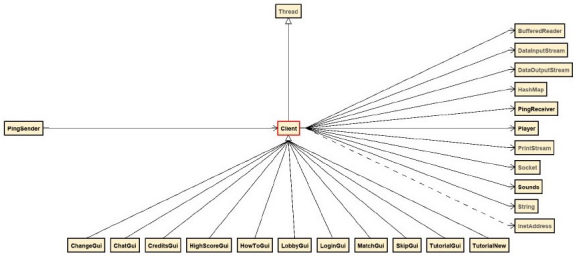
Client

edgy::gui::
ChatGui

Fields
- controlPanel : JPanel
- dim : Dimension
- mainPanel : JPanel
- message : JTextField
- scroll : JScrollPane
- statusLabel : JLabel
- textArea : JTextArea

Constructors
+ ChatGui(Dimension) : void

Methods
+ displayMessage(String) : void
- isNumber(String) : boolean
- prepareGui( ) : void
- send(String) : void
+ showChat( ) : JPanel



Thread

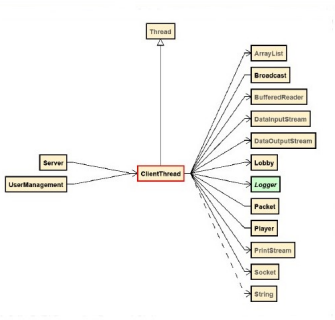
edgy::client::
Client

Constants
+ loginGUI : LoginGui
+ player : Player

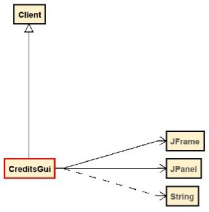
Fields
- alias : HashMap<String, String>
- clientSocket : Socket
- closed : boolean
- dataInputStream : DataInputStream
- dataOutputStream : DataOutputStream
- highscoreGUI : HighScoreGui
- inputLine : BufferedReader
- is : BufferedReader
- lobbyGUI : LobbyGui
- match : MatchGui
- os : PrintStream
- receiver : PingReceiver
sounds : Sounds

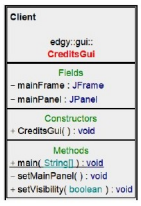
Constructors
+ Client( ) : void

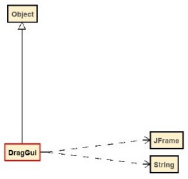
Methods
- connectToServer(String[]) : void
- disconnect( ) : void
- getClientAddress( ) : InetAddress
+ init(String[]) : void
- initStreams( ) : void
- login(String, String, String, int) : void
+ main(String[]) : void
- movePiece(String, String) : void
- mutatePiece(String, int) : void
+ parseClientInput(String) : void
- parseServerInput(String, String[]) : void
receiveFromServer( ) : String
- register(String, String, String) : void
+ run( ) : void
- sendClientAddress( ) : void
- sendPong( ) : void
sendToServer(String) : boolean
- troll( ) : void
- updateGameStatus(String) : void
- updateLobby(String) : void

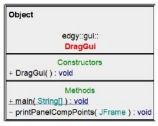

Thread
ArrayList
Broadcast
BufferedReader
DataInputStream
DataOutputStream
Server
ClientThread
UserManagement
Lobby
Logger
Packet
Player
PrintStream
Socket
String

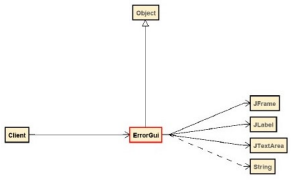

Thread
edgy::server::
ClientThread
Fields
+ broadcast : Broadcast
- clientSocket : Socket
- closed : boolean
- dataInputStream : DataInputStream
- dataOutputStream : DataOutputStream
- is : BufferedReader
- lobby : Lobby
- log : Logger
- os : PrintStream
+ packetCommunication : Packet
+ player : Player
- threads : ArrayList<ClientThread>
Constructors
+ ClientThread(Socket, ArrayList<ClientThread>) : void
Methods
«synchronized»
- closeStreams( ) : void
- disconnect( ) : void
«synchronized»
- initStreams( ) : void
+ printToClient(String) : void
+ receiveClientAddress( ) : void
+ receiveFromClient( ) : String
+ run( ) : void
+ sendToClient(String) : boolean

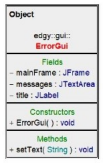

Client
CreditsGui
JFrame
JPanel
String

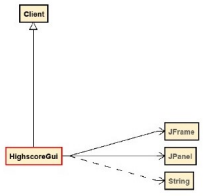

Client
edgy::gui::
CreditsGui
Fields
- mainFrame : JFrame
- mainPanel : JPanel
Constructors
+ CreditsGui( ) : void
Methods
+ main(String[]) : void
- setMainPanel( ) : void
+ setVisibility(boolean) : void

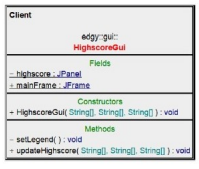

Object
DragGui
JFrame
String

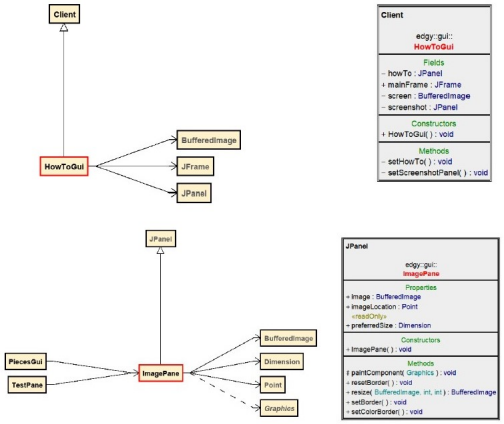

Object
edgy::gui::
DragGui
Constructors
+ DragGui( ) : void
Methods
+ main(String[]) : void
- printPanelCompPoints(JFrame) : void

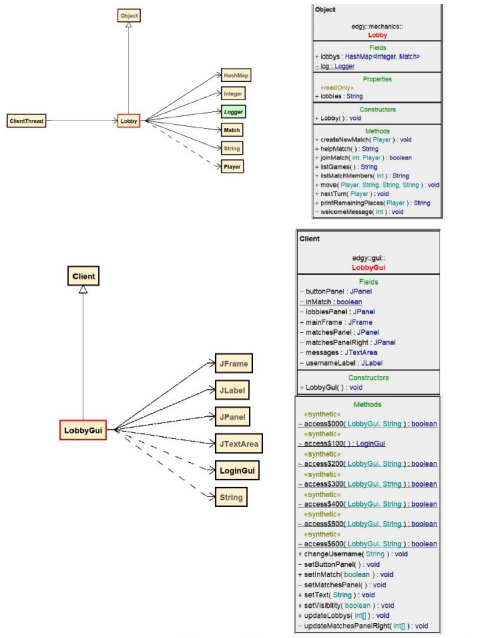

Object
Client
ErrorGui
JFrame
JLabel
JTextArea
String

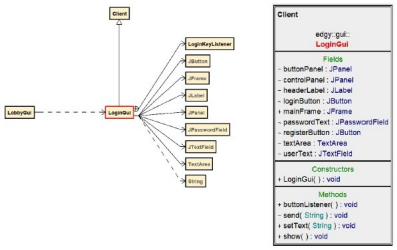

Object
edgy::gui::
ErrorGui
Fields
- mainFrame : JFrame
- messages : JTextArea
- title : JLabel
Constructors
+ ErrorGui( ) : void
Methods
+ setText(String) : void


Client
HighscoreGui
JFrame
JPanel
String

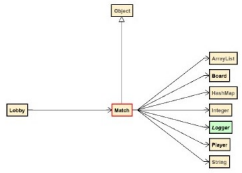

Client
edgy::gui::
HighscoreGui
Fields
- highscore : JPanel
+ mainFrame : JFrame
Constructors
+ HighscoreGui(String[], String[], String[]) : void
Methods
- setLegend( ) : void
+ updateHighscore(String[], String[], String[]) : void


Client
HowToGui
BufferedImage
JFrame
JPanel
Client
edgy::gui::
HowToGui
Fields
- howTo : JPanel
+ mainFrame : JFrame
- screen : BufferedImage
- screenshot : JPanel
Constructors
+ HowToGui( ) : void
Methods
- setHowTo( ) : void
- setScreenshotPanel( ) : void
JPanel
PiecesGui
TextPane
ImagePane
BufferedImage
Dimension
Point
Graphics
JPanel
edgy::gui::
ImagePane
Properties
+ image : BufferedImage
+ imageLocation : Point
«readOnly»
+ preferredSize : Dimension
Constructors
+ ImagePane( ) : void
Methods
paintComponent(Graphics) : void
+ resetBorder( ) : void
+ resize(BufferedImage, int, int) : BufferedImage
+ setBorder( ) : void
+ setColorBorder( ) : void

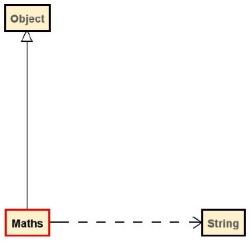

Object
ClientThread
Lobby
HashMap
Integer
Logger
Match
String
Player
Object
edgy::mechanics::
Lobby
Fields
+ lobbys : HashMap<Integer, Match>
- log : Logger
Properties
«readOnly»
+ lobbies : String
Constructors
+ Lobby( ) : void
Methods
+ createNewMatch(Player) : void
+ helpMatch( ) : String
+ joinMatch(int, Player) : boolean
+ listGames( ) : String
+ listMatchMembers(int) : String
+ move(Player, String, String, String) : void
+ nextTurn(Player) : void
+ printRemainingPieces(Player) : String
- welcomeMessage(int) : void
Client
LobbyGui
JFrame
JLabel
JPanel
JTextArea
LoginGui
String
Client
edgy::gui::
LobbyGui
Fields
- buttonPanel : JPanel
- inMatch : boolean
- lobbiesPanel : JPanel
- mainFrame : JFrame
- matchesPanel : JPanel
- matchesPanelRight : JPanel
- messages : JTextArea
- usernameLabel : JLabel
Constructors
+ LobbyGui( ) : void
Methods
«synthetic»
- access$300(LobbyGui, String) : boolean
«synthetic»
- access$100( ) : LoginGui
«synthetic»
- access$200(LobbyGui, String) : boolean
«synthetic»
- access$300(LobbyGui, String) : boolean
«synthetic»
- access$400(LobbyGui, String) : boolean
«synthetic»
- access$500(LobbyGui, String) : boolean
«synthetic»
- access$600(LobbyGui, String) : boolean
+ changeUsername(String) : void
- setButtonPanel( ) : void
+ setInMatch(boolean) : void
- setMatchesPanel( ) : void
+ setText(String) : void
+ setVisibility(boolean) : void
+ updateLobbys(int[]) : void
- updateMatchesPanelRight(int[]) : void

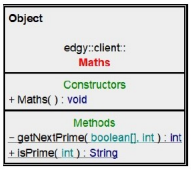

Client
LoginKeyListener
JButton
JFrame
JLabel
JPanel
JPasswordField
JTextField
TextArea
String
LobbyGui
LoginGui

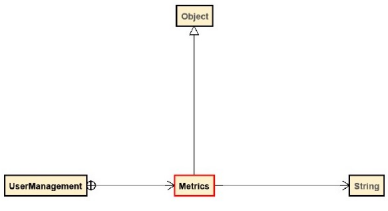
Client
edgy::gui::
LoginGui
Fields
- buttonPanel : JPanel
- controlPanel : JPanel
- headerLabel : JLabel
- loginButton : JButton
+ mainFrame : JFrame
- passwordText : JPasswordField
- registerButton : JButton
- textArea : TextArea
- userText : JTextField
Constructors
+ LoginGui( ) : void
Methods
+ buttonListener( ) : void
- send(String) : void
+ setText(String) : void
+ show( ) : void

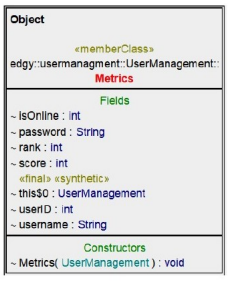

Object
ArrayList
Board
HashMap
Integer
Logger
Player
String
Lobby
Match

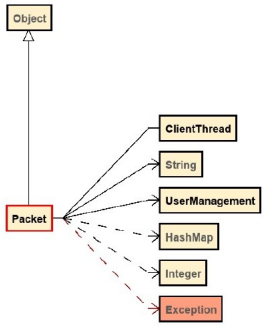

Object
edgy::mechanics::
Match
Fields
- board : Board
- log : Logger
- matchID : int
Properties
+readOnly+
+ board : String
+readOnly+
+ full : boolean
+readOnly+
+ matchNumber : int
+readOnly+
+ matchPlayers : ArrayList<String>
+readOnly+
+ playersAmount : int
+readOnly+
+ players : HashMap<Integer, Player>
Constructors
+ Match(int, Player) : void
Methods
+ addPlayer(int, Player) : void
+ calculateGameEnd( ) : boolean
+ calculateRanking( ) : int[]
+ calculateScores( ) : HashMap<String, Integer>
+ checkDraGame( ) : boolean
- checkMove(Player, int, int, int) : boolean
+ endGame(Player) : void
+ listPieces(int) : String
+ move(int, int, int, int) : boolean
+ movePiece(Player, int, int, int) : void
+ predictHint(int) : String
+ printBoard( ) : void
+ printOnePiece(int[][], int) : String
+ printPieces(int, HashMap<Integer, Match>) : String
- printScores( ) : void
- removePiece(int, int) : void
+ setHmPlayers(HashMap<Integer, Player>) : void
+ setNextTurnTurn(Player) : void
+ updateMatch(Player) : void
+ welcomeMessage( ) : void
+ whoseTurn( ) : String


Object
Maths
String

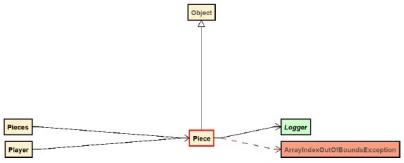

Object
edgy::client::
Maths
Constructors
+ Maths( ) : void
Methods
- getNextPrime(boolean[], int) : int
+ isPrime(int) : String

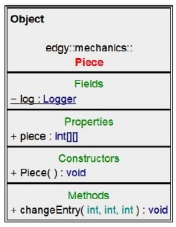

Object
UserManagement
Metrics
String

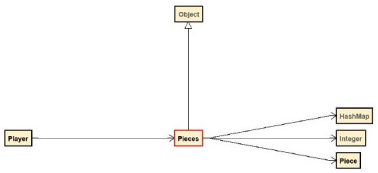

Object
«memberClass»
edgy::usermanagment::UserManagement::
Metrics
Fields
~ isOnline : int
~ password : String
~ rank : int
~ score : int
«final» «synthetic»
~ this$0 : UserManagement
~ userID : int
~ username : String
Constructors
~ Metrics(UserManagement) : void

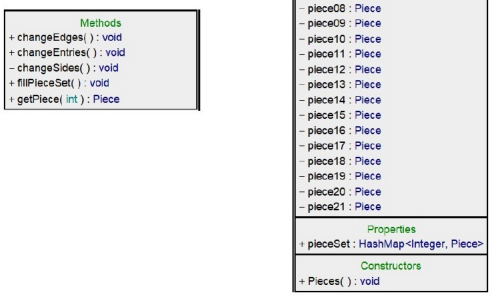

Object
Packet
ClientThread
String
UserManagement
HashMap
Integer
Exception


Object
edgy::server::
Packet
Fields
- ct : ClientThread
- userManagement : UserManagement
Properties
«readOnly»
+ ranks : String
Constructors
+ Packet(ClientThread) : void
Methods
- getMatchID( ) : int
- help( ) : void
+ parseClientInput(String, String[]) : void
- pleaseLogin( ) : boolean
- pleaseLogout( ) : boolean
+ sendBoard( ) : void
+ sendPieces( ) : void
- sendPong( ) : void
+ sendRanks( ) : void
- sendTroll( ) : void
- skip( ) : void
+ updateScore(HashMap<String, Integer>) : void

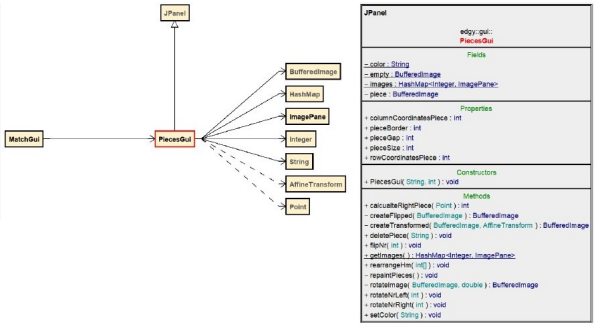

Object
Pieces
Player
Piece
Logger
ArrayIndexOutOfBoundsException

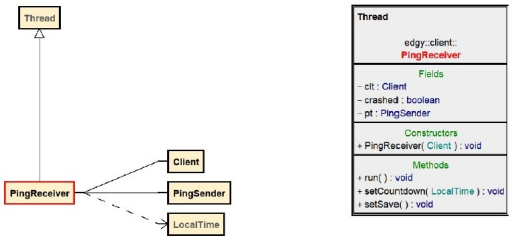

Object
edgy::mechanics::
Piece
Fields
- log : Logger
Properties
+ piece : int[][]
Constructors
+ Piece( ) : void
Methods
+ changeEntry(int, int, int) : void

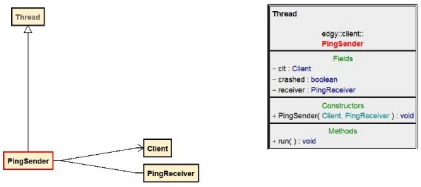

Object
Player
Pieces
HashMap
Integer
Piece

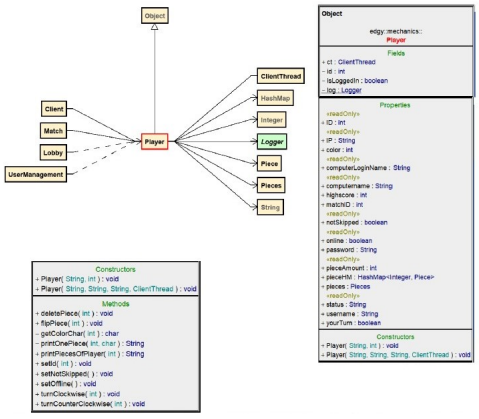

Methods
+ changeEdges( ) : void
+ changeEntries( ) : void
- changeSides( ) : void
+ fillPieceSet( ) : void
+ getPiece(int) : Piece

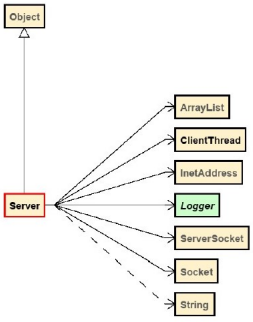
Object
edgy::mechanics::
Pieces
Fields
- piece01 : Piece
- piece02 : Piece
- piece03 : Piece
- piece04 : Piece
- piece05 : Piece
- piece06 : Piece
- piece07 : Piece
- piece08 : Piece
- piece09 : Piece
- piece10 : Piece
- piece11 : Piece
- piece12 : Piece
- piece13 : Piece
- piece14 : Piece
- piece15 : Piece
- piece16 : Piece
- piece17 : Piece
- piece18 : Piece
- piece19 : Piece
- piece20 : Piece
- piece21 : Piece
Properties
+ pieceSet : HashMap<Integer, Piece>
Constructors
+ Pieces( ) : void

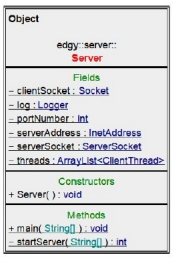

JPanel
MatchGui
PiecesGui
BufferedImage
HashMap
ImagePane
Integer
String
AffineTransform
Point
JPanel
edgy::gui::
PiecesGui
Fields
– color : String
– empty : BufferedImage
– images : HashMap<Integer, ImagePane>
– piece : BufferedImage
Properties
+ columnCoordinatesPiece : int
+ pieceBorder : int
+ pieceGap : int
+ pieceSize : int
+ rowCoordinatesPiece : int
Constructors
+ PiecesGui(String, int) : void
Methods
+ calculateRightPiece(Point) : int
– createFlipped(BufferedImage) : BufferedImage
– createTransformed(BufferedImage, AffineTransform) : BufferedImage
+ deletePiece(String) : void
+ flipNr(int) : void
+ getImages( ) : HashMap<Integer, ImagePane>
+ rearrange1Int(int[]) : void
– repaintPieces( ) : void
– rotateImage(BufferedImage, double) : BufferedImage
+ rotateNrLeft(int) : void
+ rotateNrRight(int) : void
+ setColor(String) : void

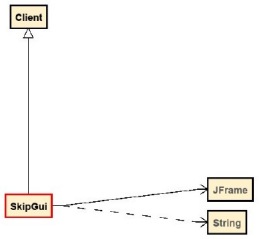

Thread
PingReceiver
Client
PingSender
LocalTime
Thread
edgy::client::
PingReceiver
Fields
– clt : Client
– crashed : boolean
– pt : PingSender
Constructors
+ PingReceiver(Client) : void
Methods
+ run( ) : void
+ setCountdown(LocalTime) : void
+ setSave( ) : void

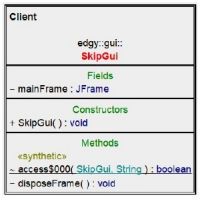

Thread
PingSender
Client
PingReceiver
Thread
edgy::client::
PingSender
Fields
– clt : Client
– crashed : boolean
– receiver : PingReceiver
Constructors
+ PingSender(Client, PingReceiver) : void
Methods
+ run( ) : void

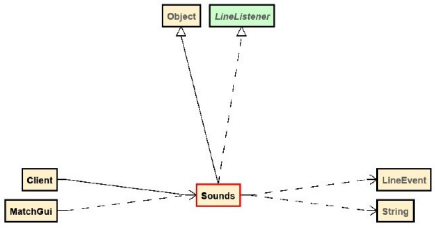

Object
Client
Match
Lobby
UserManagament
Player
ClientThread
HashMap
Integer
Logger
Piece
Pieces
String
Constructors
+ Player(String, int) : void
+ Player(String, String, String, ClientThread) : void
Methods
+ deletePiece(int) : void
+ flipPiece(int) : void
– getColorChar(int) : char
– printOnePiece(int, char) : String
+ printPiecesOfPlayer(int) : String
+ setId(int) : void
+ setNotSkipped( ) : void
+ setOffline( ) : void
+ turnClockwise(int) : void
+ turnCounterClockwise(int) : void
Object
edgy::mechanics::
Player
Fields
+ ct : ClientThread
– id : int
– isLoggedIn : boolean
– log : Logger
Properties
«readOnly»
+ ID : int
«readOnly»
+ IP : String
+ color : int
«readOnly»
+ computerLoginName : String
«readOnly»
+ computerName : String
+ highscore : int
+ matchID : int
«readOnly»
+ netSkipped : boolean
«readOnly»
+ online : boolean
+ password : String
«readOnly»
+ pieceAmount : int
+ pieceHM : HashMap<Integer, Piece>
+ pieces : Pieces
«readOnly»
+ status : String
+ username : String
+ yourTurn : boolean
Constructors
+ Player(String, int) : void
+ Player(String, String, String, ClientThread) : void


Object
ArrayList
ClientThread
InetAddress
Server
Logger
ServerSocket
Socket
String

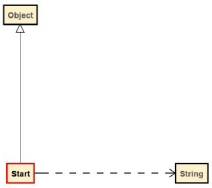

Object
edgy::server::
Server
Fields
– clientSocket : Socket
– log : Logger
– portNumber : int
– serverAddress : InetAddress
– serverSocket : ServerSocket
– threads : ArrayList<ClientThread>
Constructors
+ Server( ) : void
Methods
+ main(String[]) : void
– startServer(String[]) : int

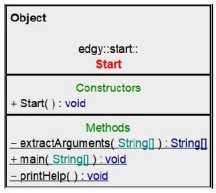

Client
SkipGui
JFrame
String

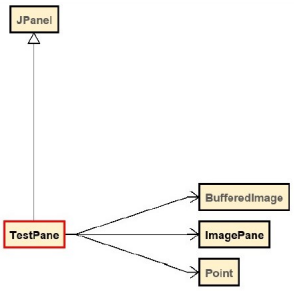

Client
edgy::gui::
SkipGui
Fields
– mainFrame : JFrame
Constructors
+ SkipGui( ) : void
Methods
«synthetic»
– access$000(SkipGui, String) : boolean
– disposeFrame( ) : void


Object
LineListener
Client
MatchGui
Sounds
LineEvent
String


Object
edgy::sound::
Sounds
Constructors
+ Sounds( ) : void
Methods
+ main(String[]) : void
+ playLost( ) : void
+ playPlace( ) : void
+ playSound(String, int) : void
+ playTrololo( ) : void
+ playTurn( ) : void
+ playWin( ) : void
+ update(LineEvent) : void


Object
Start
String


Object
edgy::start::
Start
Constructors
+ Start( ) : void
Methods
– extractArguments(String[]) : String[]
+ main(String[]) : void
– printHelp( ) : void



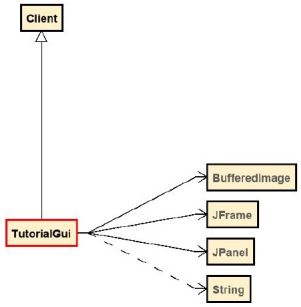
JPanel

edgy::gui::
TestPane

Fields
- imagePane : ImagePane
- imagePoint : Point
- picture : BufferedImage

Constructors
+ TestPane( ) : void

Methods
- createImagepane( ) : ImagePane



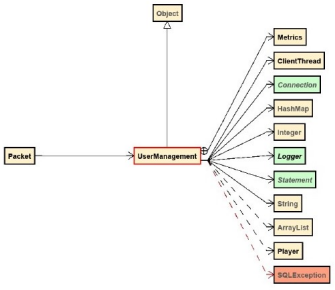
Client

edgy::gui::
TutorialGui

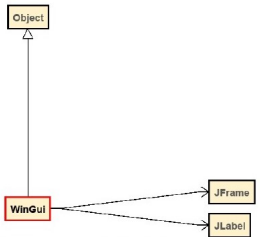
Fields
+ mainFrame : JFrame
- rules : JPanel
- screen : BufferedImage
- screenshot : JPanel

Constructors
+ TutorialGui( ) : void

Methods
+ main(String[]) : void
- setRulesPanel( ) : void
- setScreenshotPanel( ) : void



Methods
+ changePassword(String, Player) : boolean
+ changeUsername(String, Player) : boolean
- checkPassword(Metrics, String) : boolean
+ close( ) : void
+ connect( ) : void
+ createUser(String, String, Player) : boolean
- findLargestUserID( ) : int
+ getRank(String) : int
+ getUserID(String) : int
+ insertUserSql(String, String) : void
+ list(ArrayList<ClientThread>) : void
+ login(String, String, Player) : boolean
+ logout(Player) : void
+ printHighscore( ) : void
+ removeUser(String) : void
+ resetStates( ) : void
+ suggestUsername( ) : void
- updateOnlineStatusSql(int, boolean) : boolean
+ updateScore(String, int) : boolean



Object

edgy::gui::
WinGui

Fields
+ mainFrame : JFrame
- rank : JLabel

Constructors
+ WinGui(int) : void