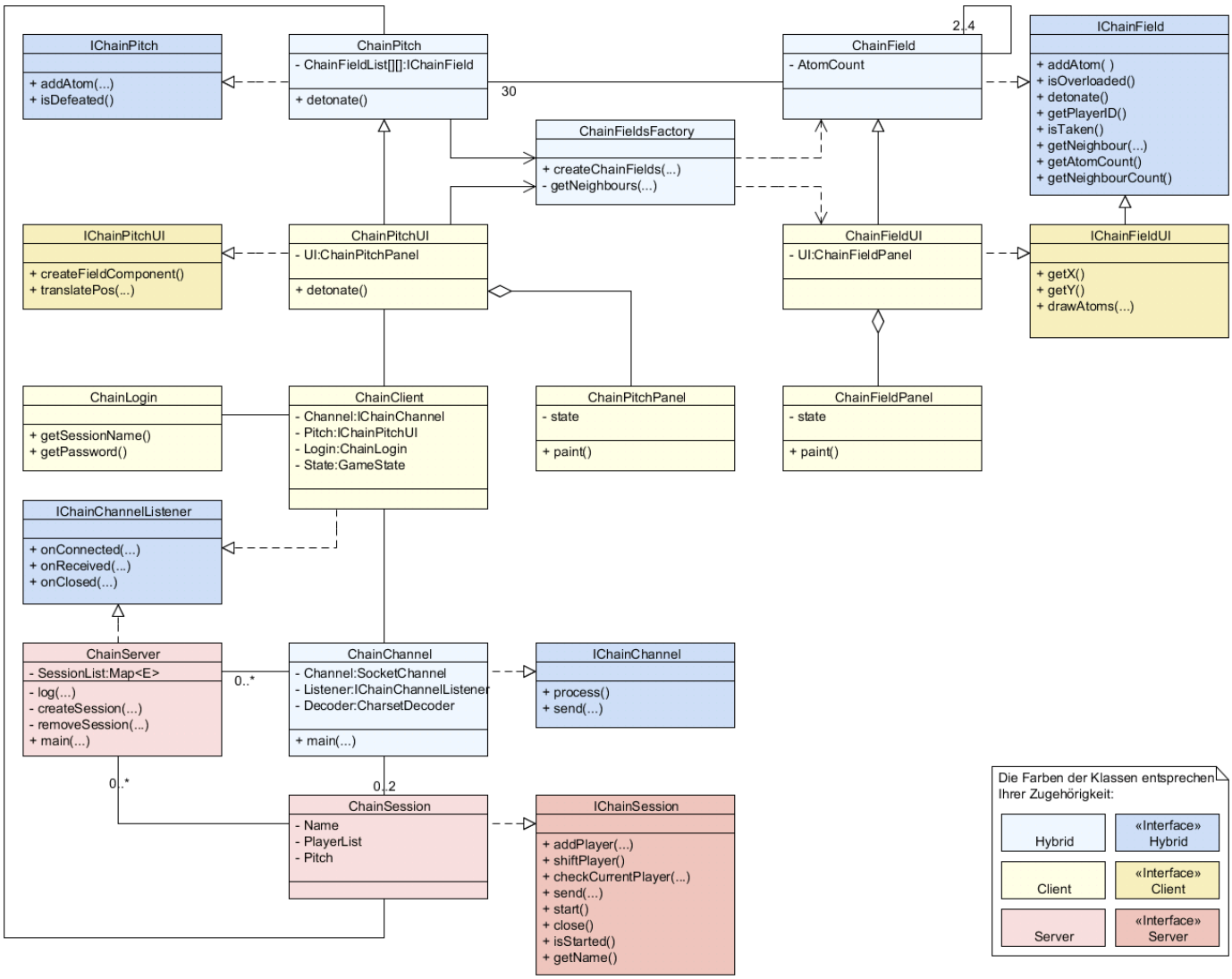
Java NIO

Autor: Manuel Siekmann

Titel: Chain Reaction



© Copyright Manuel Siekmann. All Rights Reserved

Table Of Content

Class ChainChannel

```
java.lang.Object
    |
    +--ChainChannel
```

All Implemented Interfaces:
IChainChannel

< Constructors > < Methods >

---

public class **ChainChannel**
extends java.lang.Object
implements IChainChannel

Constructors

ChainChannel

```
public  ChainChannel(java.nio.channels.Selector selector,
                     java.nio.channels.SocketChannel channel,
                     IChainChannelListener listener)
```

Erstellt eine neue Instanz vom Kommunikationskanal

Methods

getAddress

```
public java.lang.String getAddress()
```

Ermittelt die Adresse der Gegenstelle

Returns:

Name der Gegenstelle

---

process

```
public boolean process()
```

Startet die Verarbeitung der empfangenen Daten.

Returns:

true, wenn die Verarbeitung erfolgreich war. Anderenfalls false

---

send

`public boolean` **send**(ChainMessage msg)

Sendet eine Nachricht an die Gegenstelle

Parameters:

msg - definiert die zu sendende Nachricht

Returns:

true, wenn die Nachricht erfolgreich gesendet wurde. Anderenfalls false

---

Class ChainClient

```
java.lang.Object
    |
    +--java.awt.Component
        |
        +--java.awt.Container
            |
            +--java.awt.Window
                |
                +--java.awt.Frame
                    |
                    +--javax.swing.JFrame
                        |
                        +--ChainClient
```

All Implemented Interfaces:

IChainChannelListener, java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, java.lang.Runnable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.TransferHandler.HasGetTransferHandler, javax.swing.WindowConstants

---

< Constructors > < Methods >

---

public class **ChainClient**
extends javax.swing.JFrame
implements IChainChannelListener, java.lang.Runnable

Constructors

ChainClient

`public` **ChainClient**(java.lang.String titel)

Erstellt eine neue Instanz der Client-Software

ChainClient

```
public  ChainClient(java.lang.String titel,
                    java.lang.String hostname,
                    int port)
```

Erstellt eine neue Instanz der Client-Software mit definierten Verbindungsdaten

Methods

createPitchComponent

```
public ChainPitchPanel createPitchComponent()
```

Erstellt die visuelle Komponente für das Spielfeld

Returns:

Visuelle Komponente

---

main

```
public static void main(java.lang.String[] args)
```

Einsprungspunkt für die Client-Software

---

onClosed

```
public void onClosed(ChainMessage e)
```

Der Server hat die Verbindung getrennt

Parameters:

e - definiert die empfangene Nachricht

---

onConnected

```
public void onConnected(ChainMessage e)
```

Wird aufgerufen, wenn eine neue Verbindung aufgebaut wurde

---

onReceived

```
public void onReceived(ChainMessage e)
```

Verarbeitung der empfangenen Daten aus Sicht der Client-Software.

Parameters:

e - definiert die empfangene Nachricht

---

run

```
public void run()
```

---

simulate

```
public static void simulate(java.lang.String hostname,
                            int port)
```

Simuliert zwei Instanzen von ChainClient

Parameters:

hostname - definiert den Servernamen, zu dem die Verbindung aufgebaut werden soll
port - definiert den Port, zu dem die Verbindung aufgebaut werden soll

---

Class ChainField

```
java.lang.Object
    |
    +--ChainField
```

All Implemented Interfaces:
IChainField

Direct Known Subclasses:
ChainFieldUI

< Constructors > < Methods >

---

public class **ChainField**
extends java.lang.Object
implements IChainField

Constructors

ChainField

```
public  ChainField()
```

Methods

addAtom

```
public boolean addAtom(int playerID)
```

Erhöht den Zähler der Atome für dieses Element und validiert bzw. definiert den Spieler

Parameters:

playerID - Die Identifikations-ID für den Spieler

Returns:

true, wenn der Zähler erfolgreich geändert wurde. Anderenfalls false

---

addAtom

```
public boolean addAtom(int playerID,
                       boolean force)
```

Erhöht den Zähler der Atome für dieses Element und validiert bzw. definiert den Spieler

Parameters:

playerID - Die Identifikations-ID für den Spieler
force - true überspringt die Validierung und definiert den Spieler neu

Returns:

true, wenn der Zähler erfolgreich geändert wurde. Anderenfalls false

---

detonate

```
public boolean detonate()
```

Prüft, ob dieses Element überladen ist und verteilt die Atome auf die Nachbarfelder

Returns:

true, wenn das Element überladen war. Anderenfalls false

---

getAtomCount

`public int` **`getAtomCount`**`()`

Ermittelt die aktuelle Anzahl der Atome von diesem Feld

Returns:

Anzahl der Atome

---

getNeighbour

`public` `IChainField` **`getNeighbour`**`(int index)`

Ermittelt ein Nachbarfeld

Parameters:

index - definiert den Index für ein Nachbarfeld im Bereich 0 - getNeighbourCount()

Returns:

Ein direkt an Kante liegendes Nachbarfeld

---

getNeighbourCount

`public int` **`getNeighbourCount`**`()`

Ermittelt die Anzahl der direkt an Kante liegenden Nachbarfelder

Returns:

Anzahl der Nachbarfelder

---

getPlayerID

`public int` **`getPlayerID`**`()`

Ermittelt die Identifikations-ID für den Eigentümer des Elements

Returns:

Identifikations-ID des Eigentümers

---

init

```
public void init(int x,
                 int y,
                 IChainField[] neighbours)
```

Initialisiert das Spielfeld

Parameters:

x - Position X im Raster
y - Position Y im Raster
neighbours - Direkt an Kante liegende Elemente

---

isOverloaded

```
public boolean isOverloaded()
```

Ermittelt, ob dieses Element überladen ist

Returns:

true, wenn das Element überladen ist. Anderenfalls false

---

isTaken

```
public boolean isTaken()
```

Ermittelt, ob dieses Element leer ist oder einem Spieler zugeordnet wurde

Returns:

true, wenn das Element einem Spieler gehört. Anderenfalls false

---

Class ChainFieldPanel

```
java.lang.Object
    |
    +--java.awt.Component
        |
        +--java.awt.Container
            |
            +--javax.swing.JComponent
                |
                +--javax.swing.JPanel
                    |
                    +--ChainFieldPanel
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, javax.swing.TransferHandler.HasGetTransferHandler

---

< Constructors > < Methods >

public class **ChainFieldPanel**
extends javax.swing.JPanel

Constructors

ChainFieldPanel

`public` **`ChainFieldPanel`**`()`

Methods

drawExplosion

`public void` **`drawExplosion`**`(java.awt.Graphics g)`

Zeichnet eine Explosion auf den gegebenen Context

Parameters:

g - definiert den Context, auf dem gezeichnet werden soll

---

init

`public void` **`init`**`(IChainFieldUI owner)`

Initialisiert das User Interface

Parameters:

owner - Besitzer des Panel

---

paint

`public void` **`paint`**`(java.awt.Graphics g)`

Overrides:

paint in class javax.swing.JComponent

---

setExplosion

`public void` **`setExplosion`**`(boolean explosionMode)`

Definiert den Modus für das Spielfeld.

Parameters:

explosionMode - true zeigt anstelle der Atome eine Explosion. false zeichnet die Atome entsprechend der Anzahl von IChainField.getAtomCount()

Class ChainFieldUI

```
java.lang.Object
    |
    +--ChainField
        |
        +--ChainFieldUI
```

All Implemented Interfaces:
IChainField, IChainFieldUI

< Constructors > < Methods >

---

public class **ChainFieldUI**
extends ChainField
implements IChainFieldUI

Constructors

ChainFieldUI

`public` **`ChainFieldUI`**`(ChainFieldPanel ui)`

Erstellt ein neues Element für das Spielfeld und verwendet die Oberfläche ui

Methods

addAtom

```
public boolean addAtom(int playerID,
                       boolean force)
```

Erhöht den Zähler der Atome für dieses Element und validiert bzw. definiert den Spieler

Parameters:

playerID - definiert die Identifikations-ID für den Spieler
force - true überspringt die Validierung und definiert den Spieler neu

Returns:

true, wenn der Zähler erfolgreich geändert wurde. Anderenfalls false

Overrides:

addAtom in class ChainField

---

drawAtoms

```
public void drawAtoms(java.awt.Graphics g)
```

Zeichnet die Atome, entsprechend der Anzahl von IChainField.getAtomCount()

Parameters:

g - definiert den Context, auf dem gezeichnet werden soll

---

drawAtoms

```
public void drawAtoms(java.awt.Graphics g,
                      int offsetX,
                      int offsetY,
                      int offset)
```

Zeichnet die Atome an den Standard-Positionen.

Parameters:

g - definiert den Context, auf dem gezeichnet werden soll
offsetX - Offset der X-Koordinate im gegebenen Context
offsetY - Offset der Y-Koordinate im gegebenen Context
offset - definiert den Abstand der Atome in Abhängigkeit von Ihrer Position im Element in Richtung der Nachbarelemente

---

getNeighbour

`public IChainFieldUI getNeighbour(int index)`

Ermittelt ein direkt an Kante liegendes Nachbarfeld

Parameters:

index - vom Nachbarfeld muss in einem Bereich zwischen 0 - IChainField.getNeighbourCount() liegen

Returns:

Ein direkt an Kante liegendes Nachbarfeld

Overrides:

getNeighbour in class ChainField

---

getPlayerColor

`public java.awt.Color getPlayerColor()`

Ermittelt die Farbe des Spielers, der dieses Feld besetzt

Returns:

Spielerfarbe

---

getX

`public int getX()`

Ermittelt die X-Koordinate vom ersten Pixel des User Interface

Returns:

X-Koordinate

---

getY

`public int getY()`

Ermittelt die Y-Koordinate vom ersten Pixel des User Interface

Returns:

Y-Koordinate

---

setExplosion

```
public void setExplosion(boolean value)
```

Definiert den Modus für das Spielfeld.

Parameters:

value - true zeigt anstelle der Atome eine Explosion. false zeichnet die Atome entsprechend der Anzahl von IChainField.getAtomCount()

Class ChainLogin

```
java.lang.Object
    |
    +--java.awt.Component
        |
        +--java.awt.Container
            |
            +--javax.swing.JComponent
                |
                +--javax.swing.JPanel
                    |
                    +--ChainLogin
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, javax.swing.TransferHandler.HasGetTransferHandler

---

< Constructors > < Methods >

---

public class **ChainLogin**
extends javax.swing.JPanel

Constructors

ChainLogin

```
public  ChainLogin(java.awt.event.ActionListener listener)
```

Methods

getPassword

```
public java.lang.String getPassword()
```

---

getSessionName

```
public java.lang.String getSessionName()
```

---

setText

```
public void setText(java.lang.String value)
```

Class ChainMessage

```
java.lang.Object
    |
    +--ChainMessage
```

< Fields > < Constructors > < Methods >

---

public class **ChainMessage**
extends java.lang.Object

Fields

CHAIN_SOCKET_CLOSE

```
public static final java.lang.String CHAIN_SOCKET_CLOSE
```

CHAIN_SOCKET_CREATE

```
public static final java.lang.String CHAIN_SOCKET_CREATE
```

CHAIN_SOCKET_ERR

```
public static final java.lang.String CHAIN_SOCKET_ERR
```

CHAIN_SOCKET_SET

```
public static final java.lang.String CHAIN_SOCKET_SET
```

CHAIN_SOCKET_START

```
public static final java.lang.String CHAIN_SOCKET_START
```

CHAIN_SOCKET_VERSION

```
public static final java.lang.String CHAIN_SOCKET_VERSION
```

CHAIN_SOCKET_WIN

```
public static final java.lang.String CHAIN_SOCKET_WIN
```

Constructors

ChainMessage

```
public  ChainMessage(IChainChannel channel)
```

Erstellt eine neue Instanz einer Nachricht aus dem Übertragungsprotokoll

---

ChainMessage

```
public  ChainMessage(IChainChannel channel,
                     java.lang.String message)
```

Erstellt eine neue Instanz einer Nachricht aus dem Übertragungsprotokoll

---

ChainMessage

```
public  ChainMessage(java.lang.String msgType,
                     java.lang.String[] params)
```

Erstellt eine neue Instanz einer Nachricht aus dem Übertragungsprotokoll

Methods

createCloseMsg

```
public static ChainMessage createCloseMsg(java.lang.String name,
                                           java.lang.String message)
```

Erstellt eine neue Nachricht, um über das Ende eine Spielsitzung zu informieren (Nachricht vom Server)

Parameters:

name - definiert den Namen der Sitzung
message - enthält eine Begründung für das Ende einer Sitzung

Returns:

Nachricht entsprechend der Definition aus dem Übertraungsprotokoll

---

createCreateMsg

```
public static ChainMessage createCreateMsg(java.lang.String name,
                                            java.lang.String password,
                                            int sizeX,
                                            int sizeY)
```

Erstellt eine neue Nachricht, um ein Spielsitzung zu erstellen (Nachricht vom Client und vom Server)

Parameters:

name - definiert den Namen der Sitzung
password - definiert das Passwort der Sitzung
sizeX - definiert die horizontale Größe des Spielfeldes
sizeY - definiert die vertikale Größe des Spielfeldes

Returns:

Nachricht entsprechend der Definition aus dem Übertraungsprotokoll

---

createErrorMsg

```
public static ChainMessage createErrorMsg(java.lang.String error)
```

Erstellt eine neue Nachricht, um über einen Fehler zu informieren (Nachricht vom Server)

Parameters:

error - enthält den Fehlertext

Returns:

Nachricht entsprechend der Definition aus dem Übertraungsprotokoll

---

createSetMsg

```
public static ChainMessage createSetMsg(java.lang.String name,
                                         int x,
                                         int y)
```

Erstellt eine neue Nachricht, um ein Atom auf dem Spielfeld zu platzieren (Nachricht vom Client)

Parameters:

name - definiert den Namen der Sitzung
x - definiert die Position der X-Koordinate im Raster des Spielfeldes
y - definiert die Position der Y-Koordinate im Raster des Spielfeldes

Returns:

Nachricht entsprechend der Definition aus dem Übertraungsprotokoll

---

createSetMsg

```
public static ChainMessage createSetMsg(java.lang.String name,
                                         int x,
                                         int y,
                                         int playerID)
```

Erstellt eine neue Nachricht, um über ein neues Atom auf dem Spielfeld zu informieren (Nachricht vom Server)

Parameters:

name - definiert den Namen der Sitzung
x - definiert die Position der X-Koordinate im Raster des Spielfeldes
y - definiert die Position der Y-Koordinate im Raster des Spielfeldes
playerID - definiert die Identifikations-ID des Spielers, der das neue Atom platziert hat

Returns:

Nachricht entsprechend der Definition aus dem Übertraungsprotokoll

---

createStartMsg

```
public static ChainMessage createStartMsg(java.lang.String name,
                                           int playerID)
```

Erstellt eine neue Nachricht, um eine Spiel zu starten (Nachricht vom Server)

Parameters:

name - definiert den Namen der Sitzung
playerID - definiert die Identifikations-ID des Spielers

Returns:

Nachricht entsprechend der Definition aus dem Übertraungsprotokoll

---

createVersionMsg

`public static ChainMessage createVersionMsg(java.lang.String version)`

Erstellt eine neue Nachricht mit Informationen zur Version des Protokolls (Nachricht vom Server)

Parameters:

version - des Protokolls

Returns:

Nachricht entsprechend der Definition aus dem Übertraungsprotokoll

---

createWinMsg

`public static ChainMessage createWinMsg(java.lang.String name, int playerID)`

Erstellt eine neue Nachricht, um über den Spielsieg zu informieren (Nachricht vom Server)

Parameters:

name - definiert den Namen der Sitzung
playerID - definiert die Identifikations-ID des Spielers, der das Spiel gewonnen hat

Returns:

Nachricht entsprechend der Definition aus dem Übertraungsprotokoll

---

getChannel

`public IChainChannel getChannel()`

Ermittelt den Kommunikationskanal einer Nachricht, wenn vorhanden

Returns:

Kommunikationskanal

---

getError

`public java.lang.String getError()`

Ermittelt der Text einer Fehlermeldung, wenn vorhanden

Returns:

Fehlertext

---

getName

`public java.lang.String` **`getName`**`()`

Ermittelt den Namen einer Sitzung, wenn vorhanden

Returns:

Sitzungsname

---

getPassword

`public java.lang.String` **`getPassword`**`()`

Ermittelt das Passwort einer Sitzung, wenn vorhanden

Returns:

Sitzungspasswort

---

getPlayerID

`public int` **`getPlayerID`**`()`

Ermittelt die Identifikations-ID des Spielers, wenn vorhanden

Returns:

Identifikations-ID

---

getSizeX

`public int` **`getSizeX`**`()`

Ermittelt die horizontale Größe des Spielfeldes, wenn vorhanden

Returns:

Horizontale Größe

---

getSizeY

`public int` **`getSizeY`**`()`

Ermittelt die vertikale Größe des Spielfeldes, wenn vorhanden

Returns:

Vertikale Größe

---

getType

`public java.lang.String` **`getType`**`()`

Ermittelt die Nachrichtenart, wenn vorhanden

Returns:

Nachrichtenart

---

getVersion

`public int` **`getVersion`**`()`

Ermittelt die Protokollversion aus, wenn vorhanden

Returns:

Protokollversion

---

getX

`public int` **`getX`**`()`

Ermittelt die X-Koordinate im Raster des Spielfeldes, wenn vorhanden

Returns:

X-Koordinate

---

getY

`public int` **`getY`**`()`

Ermittelt die Y-Koordinate im Raster des Spielfeldes, wenn vorhanden

Returns:

Y-Koordinate

---

toString

`public java.lang.String` **`toString`**`()`

Serialisiert die Nachricht als Text

Returns:

Serialisierte Nachricht

Overrides:

toString in class java.lang.Object

Class ChainPitch

```
java.lang.Object
    |
    +--ChainPitch
```

All Implemented Interfaces:
IChainPitch

Direct Known Subclasses:
ChainPitchUI, ModulTest

< Constructors > < Methods >

---

public class **ChainPitch**
extends java.lang.Object
implements IChainPitch

Constructors

ChainPitch

```
public  ChainPitch()
```

Erstellt ein neues Spielfeld

ChainPitch

```
public  ChainPitch(int maxX,
                   int maxY)
```

Erstellt ein neues Spielfeld und mit (maxX, maxY)-Elementen

Methods

addAtom

```
public boolean addAtom(int playerID,
                       int x,
                       int y)
```

Erhöht den Zähler der Atome für das Element an Position (X, Y) und validiert bzw. definiert den Spieler

Parameters:

playerID - Die Identifikations-ID für den Spieler
x - definiert die X-Koordinate vom Zielelement im Raster des Spielfeldes
y - definiert die Y-Koordinate vom Zielelement im Raster des Spielfeldes

Returns:

true, wenn der Zähler erfolgreich geändert wurde. Anderenfalls false

---

detonate

```
protected boolean detonate()
```

Ermittelt alle überladenen Felder und ruft deren detonate-Methode auf

Returns:

true, wenn überladenen Felder detoniert sind

---

detonate

```
protected boolean detonate(java.util.Vector fields)
```

Ruft die detonate-Methode für jedes Element einer Liste auf

Parameters:

fields - definiert eine Liste an Elementen

Returns:

true, wenn mindestens eine detonate-Methode aufgerufen wurde

---

getField

```
protected IChainField getField(int x,
                               int y)
```

Ermittelt das Feld an der Position x, y

Parameters:

x - definiert die X-Koordinate im Raster
y - definiert die Y-Koordinate im Raster

Returns:

Elemente an Position x, y

---

isDefeated

```
public boolean isDefeated()
```

Ermittelt den Status des Spielfeldes

Returns:

true, wenn das Spiel durch einen Sieg entschieden wurde. Anderenfalls false

---

setChainFields

```
public void setChainFields(IChainField[][] chainFields)
```

Definiert eine neue Liste an Elementen für das Raster

Parameters:

chainFields - Neue Liste an Elementen

---

toString

```
public java.lang.String toString()
```

Ermittelt den Zustand des Spielfeldes als Text (wird nur für Debug-Ausgaben verwendet)

Returns:

Zustand des Spielfeldes

Overrides:

toString in class java.lang.Object

Class ChainPitchPanel

```
java.lang.Object
    |
    +--java.awt.Component
        |
        +--java.awt.Container
            |
            +--javax.swing.JComponent
                |
                +--javax.swing.JPanel
                    |
                    +--ChainPitchPanel
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, javax.swing.TransferHandler.HasGetTransferHandler

---

< Constructors > < Methods >

---

public class **ChainPitchPanel**
extends javax.swing.JPanel

Constructors

ChainPitchPanel

```
public  ChainPitchPanel()
```

Methods

init

```
public void init(IChainPitchUI owner)
```

Initialisiert das User Interface

Parameters:

owner - Besitzer des Panel

---

paint

```
public void paint(java.awt.Graphics g)
```

Overrides:

paint in class javax.swing.JComponent

setFields

```
public void setFields(java.util.Vector fields,
                      int offset)
```

Initialisiert das User Interface

Parameters:

fields - definiert eine Liste an UI-Elementen, die neu gezeichnet werden sollen
offset - definiert den Abstand der Atome in Abhängigkeit von Ihrer Position im Element in Richtung der Nachbarelemente

---

Class ChainPitchUI

```
java.lang.Object
    |
    +--ChainPitch
        |
        +--ChainPitchUI
```

All Implemented Interfaces:
IChainPitch, IChainPitchUI

< Constructors > < Methods >

---

public class **ChainPitchUI**
extends ChainPitch
implements IChainPitchUI

Constructors

ChainPitchUI

```
public  ChainPitchUI(int maxX,
                     int maxY,
                     ChainPitchPanel ui)
```

Erstellt ein neues Spielfeld und initialisiert die (maxX, maxY)-Elemente mit der Oberfläche ui

Methods

createFieldComponent

`public ChainFieldPanel createFieldComponent()`

Erzeugt das User Interface für ein Element aus dem Raster des Spielfeldes

Returns:

Neues Element

---

detonate

`protected boolean detonate(java.util.Vector fields)`

Ruft die detonate-Methode für jedes Element einer Liste auf und zeichnet den Zustand den Elemente neu

Parameters:

fields - definiert eine Liste an Elementen

Returns:

true, wenn mindestens eine detonate-Methode aufgerufen wurde

Overrides:

detonate in class ChainPitch

---

getField

```
protected IChainFieldUI getField(int x,
                                 int y)
```

Ermittelt das Feld an der Position x, y

Parameters:

x - X-Koordinate im Raster
y - Y-Koordinate im Raster

Returns:

Elemente an Position x, y

Overrides:

getField in class ChainPitch

---

translatePos

```
public java.awt.Point translatePos(java.awt.Point pos)
```

Transformiert die Koordinaten vom Raster des User Interface in Raster-Koordinaten des Spielfeldes

Parameters:

pos - definiert die Pixel-Position im Spielfeld

Returns:

Koordinaten im Raster des Spielfeldes

Class ChainServer

```
java.lang.Object
    |
    +--ChainServer
```

All Implemented Interfaces:
IChainChannelListener, java.lang.Runnable

< Constructors > < Methods >

---

public class **ChainServer**
extends java.lang.Object
implements IChainChannelListener, java.lang.Runnable

Constructors

ChainServer

```
public  ChainServer(int port)
```

Ein neuer ChainServer wurde erstellt

Methods

main

```
public static void main(java.lang.String[] args)
```

Einsprungspunkt für die Server-Software

Parameters:

args - definiert die Parameter der Kommandozeile (-p port)

onClosed

`public void onClosed(ChainMessage e)`

Wird aufgerufen, wenn eine Verbindung geschlossen wurde

Parameters:

e - enthält Informationen über die geschlossene Verbindung

---

onConnected

`public synchronized void onConnected(ChainMessage e)`

Wird aufgerufen, wenn eine neue Verbindung aufgebaut wurde

Parameters:

e - enthält Informationen über den neuen Spieler

---

onReceived

`public synchronized void onReceived(ChainMessage e)`

Wird aufgerufen, wenn eine neue Nachricht von einem Spieler empfangen wurde

Parameters:

e - enthält die Nachricht und Informationen über den Sender

---

removeSession

`public synchronized boolean removeSession(IChainChannel channel)`

Entfernt eine Sitzung aus der Liste verwalteter Sitzungen auf Basis von einem Spieler

Parameters:

channel - definiert den Spieler, dessen Sitzungen entfernt werden sollen

Returns:

true, wenn die Sitzung entfernt wurde

---

removeSession

```
public synchronized boolean removeSession(IChainSession session,
                                          java.lang.String message)
```

Entfernt eine Sitzung aus der Liste verwalteter Sitzungen

Parameters:

session - definiert die Sitzung, die entfernt werden soll
message - definiert eine Nachricht, die als Grund für das Ende der Sitzung an alle Teilnehmer dieser Sitzung gesendet werden soll

Returns:

true, wenn die Sitzung entfernt wurde

---

run

```
public void run()
```

---

simulate

```
public static void simulate(int port)
```

Erstellt eine neue Instanz vom Server

Parameters:

port - auf dem der Server eingehende Verbindungen erwartet

---

start

```
public void start()
```

Class ChainSession

```
java.lang.Object
    |
    +--ChainSession
```

All Implemented Interfaces:
IChainSession

< Constructors > < Methods >

---

public class **ChainSession**

extends java.lang.Object
implements IChainSession

Constructors

ChainSession

```
public  ChainSession(java.lang.String name,
                     java.lang.String password,
                     int sizeX,
                     int sizeY)
```

Methods

addPlayer

```
public boolean addPlayer(IChainChannel playerChannel)
```

Fügt einen neuen Spieler zur Teilnehmerliste einer Sitzung hinzu

Parameters:

playerChannel - definiert den neuen Spieler

Returns:

true wenn der Spieler erfolgreich der Teilnehmerliste hinzugefügt werden konnte. Anderenfalls false

---

checkCurrentPlayer

```
public boolean checkCurrentPlayer(IChainChannel playerChannel)
```

Ermittelt, ob der übergebene Spieler der gerade aktive Spieler ist

Parameters:

playerChannel - definiert den zu suchenden Spieler

Returns:

true wenn der übergebene Spieler der aktive Spieler ist. Anderenfalls false

---

close

```
public void close(java.lang.String message)
```

Beendet ein Spiel. Alle Teilnehmer erhalten eine Ende-Nachricht.

Parameters:

message - Text der Ende-Nachricht

getName

`public java.lang.String` **`getName`**`()`

Ermittelt den Namen einer Sitzung

Returns:

Name des Spiels

---

getPitch

`public` [`IChainPitch`] **`getPitch`**`()`

Ermittelt das Spielfeld dieser Sitzung

Returns:

Spielfeld

---

getPlayerCount

`public int` **`getPlayerCount`**`()`

Ermittelt die Anzahl der Spieler in der Teilnehmerliste dieser Sitzung

Returns:

Anzahl der Teilnehmer

---

getPlayerID

`public int` **`getPlayerID`**`(`[`IChainChannel`] `playerChannel)`

Ermittelt die Spieler-Identifikation für einen Spieler

Parameters:

playerChannel - definiert den zu suchenden Spieler

Returns:

Spieler-Identifikation oder -1, falls der Spieler nicht in der Teilnehmerliste gefunden wurde

---

isPasswordValid

```
public boolean isPasswordValid(java.lang.String password)
```

Ermittelt, ob das übergebene Passwort für diese Sitzung gültig ist

Parameters:

password - definiert das zu prüfende Passwort

Returns:

true wenn das Passwort gültig ist. Anderenfalls false

---

isSizeValid

```
public boolean isSizeValid(int sizeX,
                           int sizeY)
```

Ermittelt, ob die Größe für dieses Spielfeld gültig ist

Parameters:

sizeX - definiert die maximale horizontale Größe
sizeY - definiert die maximale vertikale Größe

Returns:

true wenn die Größe gültig ist. Anderenfalls false

---

isStarted

```
public boolean isStarted()
```

Ermittelt, ob das Spiel bereits gestartet wurde

Returns:

true, wenn das Spiel bereits gestartet wurde

---

send

```
public boolean send(ChainMessage message)
```

Sendet eine Nachricht an alle Teilnehmer dieser Sitzung

Parameters:

message - Zu sendende Nachricht

Returns:

true wenn . Anderenfalls false

---

shiftPlayer

```
public void shiftPlayer()
```

Verschiebt den aktiven Spieler auf den nächsten Spieler der Teilnehmerliste

---

start

```
public void start()
```

Startet das Spiel. Alle Teilnehmer erhalten eine Start.Nachricht. Der erste Spieler in der Teilnehmer-Liste ist jetzt am Zug.

Class ChainSimulator

```
java.lang.Object
    |
    +--ChainSimulator
```

< Constructors > < Methods >

---

public class **ChainSimulator**
extends java.lang.Object

Startet Server und Client gleichzeitig auf Port 107

Author:

Manuel Siekmann

Version:

1.0

Constructors

ChainSimulator

```
public  ChainSimulator()
```

Methods

main

```
public static void main(java.lang.String[] args)
```

Interface IChainChannel

< Methods >

---

public interface **IChainChannel**

Definiert die Schnittstelle zum Kommunikationskanal

Author:
Manuel Siekmann

Version:
1.0

Methods

getAddress

```
public java.lang.String getAddress()
```

Ermittelt die Adresse der Gegenstelle

Returns:

Name der Gegenstelle

---

process

```
public boolean process()
```

Startet die Verarbeitung der empfangenen Daten.

Returns:

true, wenn die Verarbeitung erfolgreich war. Anderenfalls false

---

send

`public boolean` **`send`**`(ChainMessage msg)`

Sendet eine Nachricht an die Gegenstelle

Parameters:

msg - definiert die zu sendende Nachricht

Returns:

true, wenn die Nachricht erfolgreich gesendet wurde. Anderenfalls false

---

Interface IChainField

< Methods >

---

public interface **IChainField**

Definiert die Schnittstelle zu einem Element aus dem Raster des Spielfeldes

Author:

Manuel Siekmann

Version:

1.0

Methods

addAtom

`public boolean` **`addAtom`**`(int playerID)`

Erhöht den Zähler der Atome für dieses Element und validiert bzw. definiert den Spieler

Parameters:

playerID - ist die Identifikations-ID für den Spieler

Returns:

true, wenn der Zähler erfolgreich geändert wurde. Anderenfalls false

---

addAtom

```
public boolean addAtom(int playerID,
                       boolean force)
```

Erhöht den Zähler der Atome für dieses Element und validiert bzw. definiert den Spieler

Parameters:

playerID - ist die Identifikations-ID für den Spieler
force - true überspringt die Validierung und definiert den Spieler neu

Returns:

true, wenn der Zähler erfolgreich geändert wurde. Anderenfalls false

---

detonate

```
public boolean detonate()
```

Prüft, ob dieses Element überladen ist und verteilt die Atome auf die Nachbarfelder

Returns:

true, wenn das Element überladen war. Anderenfalls false

---

getAtomCount

```
public int getAtomCount()
```

Ermittelt die aktuelle Anzahl der Atome von diesem Feld

Returns:

Anzahl der Atome

---

getNeighbour

```
public IChainField getNeighbour(int index)
```

Ermittelt ein Nachbarfeld

Parameters:

index - definiert den Index für ein Nachbarfeld im Bereich 0 - getNeighbourCount()

Returns:

Ein direkt an Kante liegendes Nachbarfeld

---

getNeighbourCount

```
public int getNeighbourCount()
```

Ermittelt die Anzahl der direkt an Kante liegenden Nachbarfelder

Returns:

Anzahl der Nachbarfelder

---

getPlayerID

```
public int getPlayerID()
```

Ermittelt die Identifikations-ID für den Eigentümer des Elements

Returns:

Identifikations-ID des Eigentümers

---

init

```
public void init(int x,
                 int y,
                 IChainField[] neighbours)
```

Initialisiert das Spielfeld

Parameters:

x - definiert die X-Koordinate im Spielraster
y - definiert die Y-Koordinate im Spielraster
neighbours - definiert eine Liste aller direkt an Kante liegenden Nachbarfelder

---

isOverloaded

```
public boolean isOverloaded()
```

Ermittelt, ob dieses Element überladen ist

Returns:

true, wenn das Element überladen ist. Anderenfalls false

---

isTaken

```
public boolean isTaken()
```

Ermittelt, ob dieses Element leer ist oder einem Spieler zugeordnet wurde

Returns:

true, wenn das Element einem Spieler gehört. Anderenfalls false

Interface IChainFieldUI

All Implemented Interfaces:
IChainField

< Methods >

---

public interface **IChainFieldUI**
implements IChainField

Methods

drawAtoms

```
public void drawAtoms(java.awt.Graphics g)
```

Zeichnet die Atome, entsprechend der Anzahl von IChainField.getAtomCount()

Parameters:

g - definiert den Context, auf dem gezeichnet werden soll

---

drawAtoms

```
public void drawAtoms(java.awt.Graphics g,
                      int offsetX,
                      int offsetY,
                      int offset)
```

Zeichnet die Atome an den Standard-Positionen.

Parameters:

g - Der Context, auf dem gezeichnet werden soll
offsetX - Offset der X-Koordinate im gegebenen Context
offsetY - Offset der Y-Koordinate im gegebenen Context
offset - definiert den Abstand der Atome in Abhängigkeit von Ihrer Position im Element in Richtung der Nachbarelemente

---

getNeighbour

`public IChainFieldUI getNeighbour(int index)`

Ermittelt ein direkt an Kante liegendes Nachbarfeld

Parameters:

index - vom Nachbarfeld muss in einem Bereich zwischen 0 - IChainField.getNeighbourCount() liegen

Returns:

Ein direkt an Kante liegendes Nachbarfeld

---

getPlayerColor

`public java.awt.Color getPlayerColor()`

Ermittelt die Farbe des Spielers, der dieses Feld besetzt

Returns:

Spielerfarbe

---

getX

`public int getX()`

Ermittelt die X-Koordinate vom ersten Pixel des User Interface

Returns:

X-Koordinate

---

getY

`public int getY()`

Ermittelt die Y-Koordinate vom ersten Pixel des User Interface

Returns:

Y-Koordinate

---

setExplosion

`public void setExplosion(boolean value)`

Definiert den Modus für das Spielfeld.

Parameters:

value - true zeigt anstelle der Atome eine Explosion, false zeigt

Interface IChainPitch

< Methods >

public interface **IChainPitch**

Definiert die Schnittstelle zum Spielfeld

Author:

Manuel Siekmann

Version:

1.0

Methods

addAtom

```
public boolean addAtom(int playerID,
                       int x,
                       int y)
```

Erhöht den Zähler der Atome für das Element an Position (X, Y) und validiert bzw. definiert den Spieler

Parameters:

playerID - Die Identifikations-ID für den Spieler
x - definiert die X-Koordinate vom Zielelement im Raster des Spielfeldes
y - definiert die Y-Koordinate vom Zielelement im Raster des Spielfeldes

Returns:

true, wenn der Zähler erfolgreich geändert wurde. Anderenfalls false

---

isDefeated

```
public boolean isDefeated()
```

Ermittelt den Status des Spielfeldes

Returns:

true, wenn das Spiel durch einen Sieg entschieden wurde. Anderenfalls false

Interface IChainPitchUI

All Implemented Interfaces:
IChainPitch

< Methods >

---

public interface **IChainPitchUI**
implements IChainPitch

Methods

createFieldComponent

`public ChainFieldPanel createFieldComponent()`

Erzeugt das User Interface für ein Element aus dem Raster des Spielfeldes

Returns:

Neues Element

---

translatePos

`public java.awt.Point translatePos(java.awt.Point pos)`

Transformiert die Koordinaten vom Raster des User Interface in Raster-Koordinaten des Spielfeldes

Returns:

Koordinaten im Raster des Spielfeldes

Interface IChainSession

< Methods >

---

public interface **IChainSession**

Definiert die Schnittstelle zu einer Spielsitzung

Author:
Manuel Siekmann

Version:
1.0

Methods

addPlayer

`public boolean` **`addPlayer`**`(IChainChannel playerChannel)`

Fügt einen neuen Spieler zur Teilnehmerliste einer Sitzung hinzu

Parameters:

playerChannel - definiert den neuen Spieler

Returns:

true wenn der Spieler erfolgreich der Teilnehmerliste hinzugefügt werden konnte. Anderenfalls false

---

checkCurrentPlayer

`public boolean` **`checkCurrentPlayer`**`(IChainChannel playerChannel)`

Ermittelt, ob der übergebene Spieler der gerade aktive Spieler ist

Parameters:

playerChannel - definiert den zu suchenden Spieler

Returns:

true wenn der übergebene Spieler der aktive Spieler ist. Anderenfalls false

---

close

`public void` **`close`**`(java.lang.String message)`

Beendet ein Spiel. Alle Teilnehmer erhalten eine Ende-Nachricht.

Parameters:

message - Text der Ende-Nachricht

---

getName

`public java.lang.String` **`getName`**`()`

Ermittelt den Namen einer Sitzung

Returns:

Name des Spiels

---

getPitch

`public IChainPitch getPitch()`

Ermittelt das Spielfeld dieser Sitzung

Returns:

Spielfeld

---

getPlayerCount

`public int getPlayerCount()`

Ermittelt die Anzahl der Spieler in der Teilnehmerliste dieser Sitzung

Returns:

Anzahl der Teilnehmer

---

getPlayerID

`public int getPlayerID(IChainChannel playerChannel)`

Ermittelt die Spieler-Identifikation für einen Spieler

Parameters:

playerChannel - definiert den zu suchenden Spieler

Returns:

Spieler-Identifikation oder -1, falls der Spieler nicht in der Teilnehmerliste gefunden wurde

---

isPasswordValid

`public boolean isPasswordValid(java.lang.String password)`

Ermittelt, ob das übergebene Passwort für diese Sitzung gültig ist

Parameters:

password - definiert das zu prüfende Passwort

Returns:

true wenn das Passwort gültig ist. Anderenfalls false

---

isSizeValid

```
public boolean isSizeValid(int sizeX,
                           int sizeY)
```

Ermittelt, ob die Größe für dieses Spielfeld gültig ist

Parameters:

sizeX - definiert die maximale horizontale Größe
sizeY - definiert die maximale vertikale Größe

Returns:

true wenn die Größe gültig ist. Anderenfalls false

---

isStarted

```
public boolean isStarted()
```

Ermittelt, ob das Spiel bereits gestartet wurde

Returns:

true, wenn das Spiel bereits gestartet wurde

---

send

```
public boolean send(ChainMessage message)
```

Sendet eine Nachricht an alle Teilnehmer dieser Sitzung

Parameters:

message - Zu sendende Nachricht

Returns:

true wenn . Anderenfalls false

---

shiftPlayer

```
public void shiftPlayer()
```

Verschiebt den aktiven Spieler auf den nächsten Spieler der Teilnehmerliste

---

start

```
public void start()
```

Startet das Spiel. Alle Teilnehmer erhalten eine Start.Nachricht. Der erste Spieler in der Teilnehmer-Liste ist jetzt am Zug.

Class ModulTest

```
java.lang.Object
    |
    +--ChainPitch
        |
        +--ModulTest
```

All Implemented Interfaces:
IChainChannelListener, IChainPitch, java.lang.Runnable

< Constructors > < Methods >

---

public class **ModulTest**
extends ChainPitch
implements IChainChannelListener, java.lang.Runnable

Constructors

ModulTest

```
public  ModulTest(java.lang.String name,
                  java.lang.String password)
```

Methods

addAtom

```
public static boolean addAtom(ModulTest player,
                              int x,
                              int y)
```

---

addAtom

```
public static boolean addAtom(ModulTest player,
                              int x,
                              int y,
                              boolean waitForSlot)
```

Definiert ein neues Atom an der Koordinate x, y für den Spieler player

Parameters:

player - identifiziert den Spieler
x - Koordinate im Raster
y - Koordinate im Raster
waitForSlot - true, wenn die Methode ausgeführt werden soll, sobald der Spieler am Zug ist oder Abbruch nach 2000 Millisekunden

Returns:

false, wenn das Atom erfolgreich definiert werden konnte. Anderenfalls true

---

addAtom

```
public synchronized void addAtom(int x,
                                 int y)
```

Definiert ein neues Atom der Koordinate x, y

---

getLastPlayerID

```
public synchronized int getLastPlayerID()
```

Spieler-Identifikation des letzten Spielzugs

---

getPlayerID

```
public synchronized int getPlayerID()
```

Spieler-Identifikation

---

isError

```
public synchronized boolean isError()
```

true, wenn der Server einen Fehler gemeldet hat

---

isPendingAtom

`public synchronized boolean **isPendingAtom**()`

true, wenn eine Nachricht noch nicht verarbeitet wurde

---

isReady

`public synchronized boolean **isReady**()`

true, wenn ein Spiel erstellt wurde

---

isStarted

`public synchronized boolean **isStarted**()`

true, wenn ein Spiel gestartet wurde

---

main

`public static void **main**(java.lang.String[] args)`

Einsprungspunkt für den Modultest. Ausgabe an stdout.

---

onClosed

`public void **onClosed**(ChainMessage e)`

---

onConnected

`public void **onConnected**(ChainMessage e)`

---

onReceived

`public void **onReceived**(ChainMessage e)`

---

run

```
public void run()
```

---

simulateComplexGame

```
public static boolean simulateComplexGame(ModulTest player1,
                                          ModulTest player2)
```

Simuliert zwei gleichzeitige Sitzungen

Returns:

true, wenn die Simulation. Anderenfalls false

---

simulateSimpleGame

```
public static boolean simulateSimpleGame(ModulTest player1,
                                         ModulTest player2)
```

Simuliert zwei gleichzeitige Sitzungen

Returns:

true, wenn die Simulation. Anderenfalls false

---

sleep

```
public static void sleep(long millis)
```

Bewirkt, dass die Ausführung des Threads millis Millisekunden ausgesetzt wird

Parameters:

millis - Wartezeit in Millisekunden

---

test_100_player

```
public static boolean test_100_player()
```

Testet 100 Spieler in 50 gleichzeitigen Spielsitzungen

Returns:

true, wenn der Test erfolgreich bestanden wurde. Anderenfalls false

---

test_complex_game

`public static boolean` **`test_complex_game`**`()`

Testet ein komplexes Spiel

Returns:

true, wenn der Test erfolgreich bestanden wurde. Anderenfalls false

---

test_input_validator

`public static boolean` **`test_input_validator`**`()`

Testet die möglichen Nutzereingaben und Fehlermeldungen

Returns:

true, wenn der Test erfolgreich bestanden wurde. Anderenfalls false

---

test_parallel_game

`public static boolean` **`test_parallel_game`**`()`

Testet zwei gleichzeitige Sitzungen

Returns:

true, wenn der Test erfolgreich bestanden wurde. Anderenfalls false

---

test_simple_game

`public static boolean` **`test_simple_game`**`()`

Testet ein einfaches Spiel

Returns:

true, wenn der Test erfolgreich bestanden wurde. Anderenfalls false

---

wait

`public static boolean` **`wait`**`(ModulTest player)`

---

wait

```
public static boolean wait(ModulTest player,
                           boolean started,
                           boolean win,
                           int delay)
```

Wartet auf eine gestartete Spielsitzung und die Abarbeitung aller zu sendenden Nachrichten

Parameters:

player - identifiziert den Spieler, auf den gewartet wird
startet - true, wenn zusätzlich auf den Spielstart gewartet werden soll
win - true, wenn zusätzlich auf den Spielsieg gewartet werden soll
delay - zeit in Millisekunden

Returns:

false, wenn bis zu einem Timeout gewartet wurde. Anderenfalls true

---

wait

```
public static boolean wait(ModulTest player,
                           int delay)
```

INDEX

A

C

D

G

I

M

O

P

R

S

T

W