IHR WOLLT ALSO EINEN SERVER FÜR EUER *MARVELOUS MASHUP*?

OFFENSICHTLICH, SONST WÄRT IHR NICHT HIER!

HIER BEKOMMT IHR EINEN KLEINEN EINBLICK IN DEN PERFEKTEN SERVER

- UND ZWAR DEN VON TEAM 23.

Team 23 – Sopra 2021

UNSER MOTTO:
SIMPLIZITÄT & STABILITÄT SCHÜTZT VOR LABILITÄT

- Von den Besten der Besten programmiert (von Studierenden in Regelstudienzeit!)
- Umfassende JavaDoc Kommentare für leichte Verständlichkeit (u.a. von Adrian ‚JavaDoc' Gröber persönlich)
- Perfekte Abstimmung an das Standardisierungsdokument
- Ausgefeilte UnitTests für optimale Funktionstüchtigkeit
- Testabdeckung von unglaublichen 69% (Nice!)
- Schönheit und Trivialität des Codes (siehe folgende Codebeispiele)

CODEBEISPIEL DER DOKUMENTATION

Hier ist unsere absolut bezaubernde Dokumentation zu sehen: Erst eine ausführliche Beschreibung, dann ein Beispiel, natürlich den Autor und zum Schluss die Parameter und Return-Werte mit Erklärungen.

```java
/**
 * This method checks if the client follows the correct succession of messages as specified in the network protocol.
 * In detail, it looks into the Attachment object attached to the WebSocket and checks if the checkpoint variable
 * has stored the correct messageType. If the Attachment object is null or if the messageType is wrong it returns
 * false.
 * EXAMPLE: If the onMessage method receives a PlayerReady message, the previous message of the client had to be a
 * HelloServer message which means that this method checks if the checkPoint variable equals the HELLO_SERVER type.
 * If the attachment is valid the checkpoint is updated and the Attachment is reattached to the WebSocket.
 *
 * @author Adrian Groeber
 *
 * @param conn the WebSocket for which the attachment object should be verified
 * @param previous the messageType which is expected to be in the checkPoint variable of the Attachment
 * @param current the messageType which has to be filled into the checkPoint variable of the Attachment
 *
 * @return returns true if the Attachment is valid which means there is no protocol violation
 */
public boolean checkAttachment(WebSocket conn, MessageType previous, MessageType current){
    Attachment attachment = conn.getAttachment();

    //the last message sent by the client has to be a HelloServer message otherwise a protocol violation is detected
    if(attachment == null || attachment.checkPoint != previous){
        return false;
    }
    else{
        attachment.checkPoint = current;
        conn.setAttachment(attachment);
        return true;
    }

}
```

Team 23

CODEBEISPIEL FÜR KOMPLEXE SACHVERHALTE

Hier wird überprüft, ob ein Feld von einem anderen Feld aus gesehen werden kann. Das ist besonders wichtig für das Stan-Lee-Event.

```java
/**
 * This method checks if a line of sight exists between two fields. For that it checks
 * every field that is in the line of sight and returns false if a Placeable on one
 * of the fields blocks the line of sight.
 * The algorithm is inspired by: https://www.redblobgames.com/grids/line-drawing.html
 *
 * @author Luka Stoehr
 *
 * @param a Start Position
 * @param b End Position
 *
 * @return Whether the line of sight is unobstructed between field a and b.
 */
public boolean checkLineOfSight(Position a, Position b){
    int ax = a.getX();
    int ay = a.getY();
    int bx = b.getX();
    int by = b.getY();

    int deltaX = bx - ax;
    int deltaY = by - ay;

    if(deltaX == 0 && deltaY == 0) return true;

    int length = max(abs(deltaX), abs(deltaY));
    int numberOfChecks = 10*length;

    for(int i = 0; i <=numberOfChecks; i++){
        double t = ((double) i)/numberOfChecks;    // t goes from 0 to 1
        //Linearly interpolate the coordinates, t=0 is start field, t=1 is end field
        double currentX = ax + t*(bx-ax);
        double currentY = ay + t*(by-ay);
        if(currentY == 0.5 || currentX == 0.5) continue;    // This coordinate is on the grid/between two fields
        if((int)Math.round(currentX) == ax && (int)Math.round(currentY) == ay) continue; //This is the start field
        if((int)Math.round(currentX) == bx && (int)Math.round(currentY) == by) continue; //This is the end field

        if(!seeThroughField(new Position((int)Math.round(currentX), (int)Math.round(currentY)))) return false;
    }
    return true;
}
```

Team 23

NOCH NICHT ÜBERZEUGT?

Dann schaut euch noch unsere umfangreiche Dokumentation an!

Wir haben ein JavaDoc-HTML-File und ein UserManual.

Falls ihr immer noch Bedenken bei eurer Entscheidung habt, wendet euch gerne an unser Team!

Unsere Hauptansprechpartner:

Luka Stöhr, luka.stoehr@uni-ulm.de

Adrian Gröber, adrian.groeber@uni-ulm.de

Team 23