Fejlesztői dokumentáció - GUI

TankTrouble

Ez az osztály az ősosztálya a játéknak.

public void addNewOwnRoom(Room newOwnRoom)

Ez a függvény eltárolja a játékos saját szobáját, amennyiben rendelkezik ilyen objektummal.

public void addNewListOfRemoteRooms(ArrayList newListOfRemoteRooms)

Ez a függvény frissíti a felfedezett szobák listáját úgy, hogy az előző listát minden elemét kitörli, majd a paraméterként megadott lista minden elemét hozzá adja a privát változóként eltárolt listához.

public ArrayList getListOfRemoteRooms()

Ez a függvény a privát változóban eltárolt elérhető szobákat kérdezi le.

public Room getOwnRoom()

Ez a függvény lekéri a játékos saját szobáját.

public boolean hasOwnRoom()

Ez a függvény lekérdezi, hogy van-e a játékosnak saját szobája. Amennyiben van akkor a vissza térési érték igaz, azaz true.

public String getThisPlayerName()

Ez a függvény lekéri ebben a játék kliensben játszó játékos játékosnevét.

public Player getThisPlayer()

Ez a függvény lekéri ebben a játék kliensben játszó játékos Player objektumát.

public void modifyPlayerName(String name)

Ez a függvény megváltoztatja az eltárolt játékosnevet.

MainMenuWindow osztály

Ez az osztály egy class-ot és egy form-ot foglal magába. A form-ban az alapvető funkciókat elrejtettem, ilyenek például a Java Swing könyvtár bizonyos objektumai, amivel képesek vagyunk GUI-t készíteni.

Konstruktor

A konstruktorban létrejön az az ablak, amiben minden építő eleme ennek az ablakban helyet fog foglalni. Továbbá itt társulnak hozzá a gombokhoz a lenyomáskor meghívódó

függvények, azaz action listenerek. A függvény nevek jól mutatják a megvalósított funkciót, így erre külön nem térek ki.

public void setMainMenuWindowFrameVisible()

Ez a függvény a main menu ablakát jeleníti meg. Ezt a függvényt abban az esetben kell meghívni, ha ezt az ablakot elrejtjük a felhasználó szeme elől, mint például játék közben, hogy ne legyen két ablak megnyitva.

private void loadMainMenuWindowImages()

Ennek a függvénynek a célja, hogy a project mappától számított `src/main/gui/resources/` elérési úton létező képeket megjelenítse a kijelölt helyén a főképernyőnek. Ennek a függvénynek hibakezelést egy esetben végeztük el, amikor a kép nem található a megadott elérési úton, így hibával zárva be a programot.

public void openCreateRoomWindow()

Ez a függvény felelős a főmenü eltüntetéséért és az új szoba létrehozása ablaknak a megjelenítéséért.

public void openListRoomsWindow()

Ez a függvény felelős a főmenü eltüntetéséért és az lobby listázó ablak megjelenítéséért.

public void setNewPlayerName()

Ebben a függvényben valósul meg a felhasználó nevének a megadása egy felugró ablak segítéségével. A felugró ablak egy beviteli felugró ablak, amibe a felhasználó tetszőleges szöveget írhat be, tetszőleges hosszúságban. Amennyiben a felhasználó beviszi a szöveget a függvény első körben ellenőrzi, hogy nulla nyitott és húsz zárt intervallumban helyezkedik-e el a bevitt szövegnek a hossza illetve ellenőrzi továbbá a bevitt szövegnek a specifikációban megállapított helyeségét, miszerint csak számokat és az angol ABC karaktereit tartalmazhatja a felhasználó neve. Ha a felhasználó rosszul adta meg a nevet, akkor egy végtelenített ciklusban újra megjelenik ez a felugró ablak, egy segítséggel, hogy miknek kell megfelelnie ennek a névnek. Amennyiben ezeknek a feltételeknek megfelelet a bevitt szöveg, abban az esetben megtörténik a felhasználó név tárolása, illetve engedélyezetté válik a szoba létrehozása és a szobák kilistázása funkció. Amennyiben a felhasználó nem adott meg a bevitt szöveg NullPointer-ként jelenne meg, így ezt a hibát le kellett kezelni. Ezt úgy valósítottuk meg, hogy a felhasználó felhasználónév nélkül tovább jut a főoldalra, viszont ott csak a kilépés lehetősége és a felhasználó név változtatása lehetőség jelenik meg számára.

public void quit()

Ez a függvény kilépteti a programot 0-s hibakóddal.

CreateRoomWindow

Ez az osztály felelős az új játék létrehozása funkció GUI megvalósítására. Önmagában ez az osztály egy felugró ablakot valósít meg.

Konstruktor

Ez a konstruktor az objektum létrejöttekor meghív egy felugró ablakot, ami megjeleníti a játokos nevéből kreált szoba nevet, illetve ezen a ponton a játékos megadhatja, hogy összesen, hány ember játszhat a szerverén egy időben. Amennyiben a CANCEL opciót választja akkor a főoldalon találja magát, viszont, ha az OK gombot választotta és a szám, amit beírt helyes, akkor létrehozza a program a szobát és tovább irányítja a játékost a lobbyba.

public void backToMainMenuWindow()

Ez a függvény felelős a hálózati felderítés leállítására, illetve a főmenü megjelenítésére.

public void createRoom(String roomName, int slots)

Ez a függvény létrehoz egy szobát a felhasználónak, amihez hozzárendeli a tulajdonos nevéből generált szoba nevet, illetve a szoba tulajdonságát, miszerint ez a szoba host ebben a kliensben. Továbbá létrejön egy lista, amiben a felcsatlakozott játékosok fognak helyet foglalni, továbbá hozzárendelődik a szobának a maximális kapacitása.

Ezek után a szoba létrehozója automatikusan hozzáadódik a szobához és a szoba elérhető lesz más kliensekből is. Legvégül pedig a program a játékost egy új felületre irányítja át, ami a lobby.

ListRoomsWindow

Konstruktor

Itt létrejön az ablak, illetve hozzáadódnak a különböző eseményeket lereagáló függvények.

private void addActionListeners()

Ez a függvény adja hozzá a gombokhoz a megnyomásukkor létrejövő esemény lereagálásához megírt függvényeket. Ilyen gombok például:

- vissza gomb, ami vissza irányít a főmenübe,
- keresés gomb, ami megjeleníti a felderített szobákat ebben az ablakban rádió gombok segítéségével,
- részvétel gomb, ami kikeresi a kiválasztott rádió gombot a listából és kezdeményezi a csatlakozást ehhez a szobához.

private void listRoomInThePanel()

public void joinChosenRoom(Room chosenRoom)

Ez a függvény hívja meg a hálózati kezelő réteg csatlakozásért felelős függvényét. Amennyiben sikeres volt a csatlakozás, abban az esetben a felhasználót egy másik ablakba irányítja át, ami a lobby.

public void backToMainMenuWindow()

Ez a függvény felelős az ablak megszüntetéséért és a főmenü megjelenítéséért.

WaitForGameStartWindow

Konstruktor

Ez hozza létre az ablakot, illetve adja hozzá azt a funkciót, ami szerint csak a szoba tulajdonosa indíthatja el a játékot. Ebben a konstruktorba foglalnak helyet a start és a kilépés gomboknak eseményeinek a lereagálása is.

public void setWaitForGameStartWindowFrameVisible()

Ez a függvény megjeleníti a lobby ablakát.

public void leaveRoom()

Ez a függvény megsemmisíti az ablakot, jelzi a hálózati rétegnek, hogy a jelenlegi kliens elhagyta a lobbyt, illetve a felhasználót visszairányítja a főmenübe.

public void startGame()

Ez a függvény elrejti ideiglenesen a lobby-t a felhasználó szeme elől, mivel játék befejeztével ide fogja őt visszadobni. Ekkor létrejön egy új játék szoba és a szerveren (ami mindig a szobát létrehozó játékos kliense) legenerálódik a pálya, a tankok helyzete és ez kirajzolódik ezen a játék kliensen. A hálózati réteg elkezdi a játékhoz szükséges adatokat küldeni a többi kliensre.

public void remoteGameStarted()

Ez a függvény felelős a lobby eltüntetéséért.

public void updateJoinedPlayerList(ArrayList list)

Ez a függvény egy játékosokból álló listából, ami a függvény paramétere megjeleníti szöveges formátumban az ablak közepén a csatlakozott játékosokat.

GameWindow

Konstruktor

A konstruktorban létrejön az ablak, illetve azt az alapvető funkcionalitás letiltásra kerül, hogy space gomb megnyomása esetén a kilépés a játékból funkció hívódjon meg. Továbbá hozzáadódnak a gombok eseményeinek lekezelésére szolgáló függvények. Ebben a konstruktorban létrejön továbbá egy időzítő, ami megadott (100 ms)-os időközönként meghívódik és a tankok helyzetét, illetve a lövedékek pozícióját frissíti ebben az ablakban. Ez a függvény figyeli továbbá, hogy hány tank helyezkedik el a pályán, mivel ha csak egyetlen egy tank marad, akkor az a játékos megnyerte a jáátékot.

public void generateBattlefield()

Ez a függvény hívja meg a Battlefield osztály pályagenerálás függvényét.

public static Battlefield getBattlefield()

Ez a függvény visszaadja a legenerált csatateret.

public void setBattleField(Battlefield newBattlefield)

Ez a függvény egy csatateret állít be ebben az objektumban.

public void drawBattlefield()

Ez a függvény végzi el a csatatér koordinátáiból a kijelzőre történő kirajzolást, úgy hogy folyamatosan új JPaneleket ad hozzá ehhez az ablakhoz és ezeknek fal, vagy pálya tulajdonság esetén fekete illetve fehér hátteret ad.

public void updateScreen()

Ez a függvény egyszere végzi el a tankok helyzetének és a lövedékek helyzetének a kirajzolását.

public void updateTank()

Ez a függvény végzi el a Tank megjelenítését a tank jelenlegi koordinátája alapján. A megjelenítés úgy zajlik, hogy amennyiben a tank még nincs elpusztítva, akkor a tank helyén lévő JPanel hátterét módosítjuk és erre a JPanelre ráhelyezünk egy tank képet. A Tanknak a képe abba az irényba van elforgatva, amerre áll a tank.

public void updateMissile()

Ez a függvény végzi el a lövedék megjelenítését a koordinátája alapján. Ezt hasonlóképpen jelenítjük meg, mint a tankokat, annyi kivétellel, hogy ezt nem kell elforgatni.

public void removeMissileFromList()

Ez a függvény eltávolítja a megsemmisült lövedéket a lövedékek listájából.

public void removeTankFromList()

Ez a függvény eltávolítja a tankot a tankok listájából.

public static void setBattlefieldMissile(Missile newMissile)

Ez a függvény elhelyez a pályán egy új lövedéket, azáltal, hogy hozzáadja a lövedékek listájához.

private static BufferedImage rotateTankImage(BufferedImage bimg, Tank currentTank)

Ez a függvény végzi el e képnek a forgatását a tank pozíciója alapján, hogy a megjelenítéskor a tank lövegtornya mindig abba az irányba álljon, amerre a tank néz.

public JFrame getGameWindowFrame()

Ez a függvény visszaadja az ablak leíró objektumát.

Battlefield

Ez az osztály valósítja meg a pályát, amelyeket Field objektumokból épít fel, valamint tárolja a pályán lévő tankokat és lövedékeket egy listában. Valamint tartalmaz egy enum osztályt, amely a pályán található akadályok formáját adja meg.

public void generateBattleFieldPositioningXYCoordinate()

Ez a függvény felelős a pálya legenerálásért, a megadott X és Y dimenziók megadásával tudjuk beállítani a pálya méretét. A pálya peremét végig fallal veszi körül, majd random különböző formájú és adott mennyiségű akadályokat helyez el a harctéren.

public Field[][] getFields()

A pálya minden mezőjét visszaadja.

public int getMazeDimensionX()

A pálya X dimenzióját adja vissza.

public int getMazeDimensionY()

A pálya Y dimenzióját adja vissza.

public void generateTanks()

Minden játékoshoz létrehoz egy tankot és ezket elmenti a tankok listájába.

public void addPlayerTankControl()

Meghívja a tank *addControl()* függvényét, ezáltal a megfelelő játékosnak adva az adott tank fölötti irányítást.

public ArrayList getListOfTanks()

A tankok listáját adja vissza.

public void setListOfTanks(ArrayList tanks)

Beállítja a tank listát már egy meglévő lista alapján.

public ArrayList getListOfMissiles()

A lövedékek listáját adja vissza.

public int[][] generateBarrier(barrierType barrier)

Különböző akadályokat hoz létre, amelyek lehetnek L, H, T és V formájúak.

public void mergeSubcoordsIntoTheBattleField(int starX, int starY, int[][] subCoord)

A létrehozott akadályokat elhelyezi a harcmezőn.

Field

Ezek a harcmező egy adott mezőjét leíró osztály, amely tartalmazza a x és y koordinátákat, valamint a típusát, ami lehet fal vagy út.

public void setType(FieldType type)

Az adott mező típusát állítja be.

public FieldType getType()

Az adott mező típusát adja vissza.

public void setX(int x)

Az adott mező X koordinátáját állítja be.

public int getX()

Az adott mező X koordinátáját adja vissza.

public void setY(int y)

Az adott mező Y koordinátáját állítja be.

public int getY()

Az adott mező Y koordinátáját adja vissza.

MovingObject

Ez az osztály az ősosztálya, azon mozgó objektumokhoz tartozó osztályoknak, amelyek a játékban résztvesznek.

Az osztály tartalmaz egy enum osztályt, amely a jobb, bal, fel és le irányokat valósítja meg. Ezen felül még tartozik hozzá 4 különböző változó: - position: Publikus változó és a Field osztály egy objektuma, azaz azt adja meg, hogy a mozgó objektum a pálya melyik mezőjén helyezkedik. - direction: Az előbb ismertetett enum osztályból származik és a mozgó objektum irányát adja meg, azaz merre halad vagy néz. - destroyed: Egy boolean típusú változó és azt jelzi, hogy a mozgó objektum megsemmisült-e vagy sem. - owner: A Player osztályból objektum, amely a mozgó objektum tulajdonosát adja meg.

Tank

A tankot, mint mozgó objektumot valósítja meg, a MovingObject osztályból származik, ezért annak változóit örökli, azokon felül, pedig tartalmaz még egy JLabel objektum változót, amely a megjelenítéséért felel.

Konstruktor

A konstruktor futása során a tank pozícióját a generált harcmező egy véletlenszerű mezőjére állítja, az irányát is véletlenszerűen választja meg, a destroyed változója false értéket veszi fel, valamint tulajdonosaként az adott játékost jelöli meg.

public void addControl()

Feljogosítja a tank tulajdonosát a tank irányítására. A mozgáshoz a nyíl, a lövéshez a SPACE billentyűket állítja be.

public void shootMissile()

A tank irányába egy lövedék objektumot hoz létre, majd változóit, a pozícióját, az irányát, a tulajdonosát és az elpusztítottság állapot tank ugyenezen változóinak értékével egyenlővé teszi.

public void moveTankToNextPosition(int KeyCode)

Ez a függvény egy int változót kap paraméterként, amely az adott billentyű kódja. Amennyiben az adott tank objektum kap a mozgatásához szükséges parancsot, az adott billentyű által, ez függvény valóstítja meg a mozgást. Elsőként azt ellenőrzi, hogy az irány megfelel-e az mozgás irányának, ha nem akkor az adott irányba fordítja, ha igen, akkor azt ellenőrzi, hogy a következő pozíció fal-e vagy út, ha út akokr tovább lép, ha fal, akkor pedig nem történik semmi.

public void destroyTank()

A tank elpusztításáért felel, azaz a destroy változót true-ra állítja. Az elpusztított tank tulajdonosának felugrik egy ablak, ahol kiválaszthatja, hogy a továbbiakban a játékot a végéig tovább nézi vagy visszalép a főmenübe.

public JLabel getThisTankJLabel()

A tank objektum thisTankJLabel változóját adja vissza.

public Tank getTankPosition()

A tank objektum position változóját adja vissza.

Missile osztály

A lövedéket, mint mozgó objektumot valósítja meg, a MovingObject osztályból származik, ezért annak változóit örökli, azokon felül, pedig tartalmaz még egy JLabel objektum változót, amely a megjelenítéséért felel. A változóinak értékét nem a konstruktor, hanem a Tank osztály *shootMissile()* függvénye adja meg.

public void updateMissilePosition()

A lövedékek mozgatásáért felel és 150 ms-ként van meghívva minden lövedék objektum esetén. Minden lövedék a direction változónak megfelelő irányba mozog, amíg falnak vagy Tanknak nem ütközik. Minden esetben ellenőrzi, hogy a következő mező fal-e vagy út, ha fal, akkor megsemmisül, azaz hívja a *destroyMissile()* függvényt. Ellenkező esetben ellenőrzi, hogy a következő mezőn tartózkodik-e tank objektum, amennyiben nem, akkor halad tovább, ha pedig igen, akkor mind a tank *destroyTank()*, mind a lövedék *destroyMissile()* függvénye meghívódik.

public void destroyMissile()

A lövedék elpusztításáért felel, azaz a destroy változót true-ra állítja.

public JLabel getThisMissileJLabel()

A lövedék objektum thisTankJLabel változóját adja vissza.

public Field getMissilePosition()

A lövedék objektum position változóját adja vissza.

Player

Ez az osztály a játékosok azonosításáért felel.

Változók: - name: Publikus, sztring típusú változó, amely a játékos nevét adja meg. - ip: Publikus, Inet4Address típusú változól, amely a játékos ip-címét tárolja. - id: Publikus, int típusó változó, amely a játékos egyedi azonosításáért felel.

Konstruktorok

A konstruktor futása során az id változó értéke egy random számmal lesz egyenlő, ha a konstruktor kap egy sztringet is paraméterként, akkor a name változó is beállításra kerül.

public void setName(String name)

A játékos nevét állítja be.

public void setIp(Inet4Address ip)

A játékos ip-jét állítja be.

NetworkController

Ez az osztály felelős a hálózati kommunikáció összefogásáért, mind a szerver, mind a kliens működésért felelős programrészek itt futnak össze.

public void startDiscovery()

A (lokális) hálózaton elérhető szobák felkutatásának elindítására használható függvény. Amikor éppen nincs folyamatban csata, és a program nem szerver módban van, akkor folyamatosan keresi a hálózaton az elérhető szobákat. Mivel a teljes hálózat végigpingelése adott esetben hosszú időbe is telhet, ezért hasznos, ha folyamatosan fut ez a funkció.

public void stopDiscovery()

A startDiscovery() függvény párja, a hálózaton elérhető szobák felkutatását lehet vele megállítani.

public void startExternalDiscoveryService()

Ha a programban szervert hozott létre a felhasználó, akkor a programnak válaszolnia kell a kívülről érkező kérdésekre, hogy elérhető-e szoba ezen a gépen. Ennek a függvénynek a segítségével lehet elindítani a szoba "felfedezhetőségét".

public void stopExternalDiscoveryService()

A startExternalDiscoveryService() függvény párja, a felfedezhetőséget lehet vele megszűntetni.

public boolean joinRoom(Room room)

Ezzel a függvénnyel lehet kliens módban kezdeményezni a (hálózaton már felfedezett) `room` szobához való kapcsolódást. A boolean visszatérési érték azt mutatja, hogy sikeres volt-e a csatlakozás.

public void closeRoom()

Szerver módban a létrehozott szobát lehet megszűntetni ennek a függvénynek a használatával mindaddig, amíg a szobában nem indult el a játék, hanem csak a lobby-van várakoznak a csatlakozott játékosok.

public static boolean handleExternalJoinRequest(Object data, InetAddress address)

Szerver mód esetén ha a hálózaton a szobához (amiben még nem indult el a játék, csak a lobby-ban várakoznak a felhasználók) csatlakozási kérés érkezik, ez a függvény hívódik meg, és csatlakoztatja az új játékost a szobához, ha lehetséges, vagy elutasítja a kérelmet.

public void leaveLobby()

Akár szerver, akár kliens módban, mikor a játékosok a lobby-ban várakoznak a játék elindulására, ezzel a függvénnyel lehet kilépni a lobby-ból. Kliens esetén csak kilép a felhasználó a játék előszobájából, míg szerver mód esetén megszűnik a szerver, és az összes felhasználó a saját kezdőképernyőjére kerül.

public void broadcastGameStarting()

A játék kezdetekor ezen a függvényen keresztül lehet a csatlakozott kliensek felé jelezni, hogy elkezdődött a játék.

public void leaveGame()

Akár szerver, akár kliens módban ezzel a függvénnyel lehet kilépni az éppen futó csatából. Kliens esetén csak az adott felhasználó lép ki a játékból, míg szerver mód esetén megszűnik a szerver, és az összes felhasználó a saját kezdőképernyőjére kerül.

private void sendClientLeave()

A program kliens módja esetén ezen a függvényen keresztül lehet jelezni a szervernek, hogy a felhasználó kilép a szobából.

public void broadcastClientLeave(Message msg)

A szerver ezen a függvényen keresztül továbbítja minden kliensre az egyik kliens lecsatlakozásának tényét (amit a sendClientLeave() függvényen keresztül kapott). A lecsatlakozás tényéről minden kliens értesül, kivéve maga a lecsatlakozó kliens.

private void broadcastServerStopping()

Szerver mód esetén, ha játék közben megszűnteti a játékot a szervert létrehozó felhasználó, ezen a függvényen keresztül kerül kiküldésre ennek a ténye az összes csatlakozott kliensre.

private void stopClientFunctions()

Ezzel a függvénnyel lehet a kliensfunkcióit leállítani a programnak, ez bontja a szerverhez csatlakozó socket-et, és minden I/O stream-et.

public void sendKeyPress(int key)

Ha a felhasználó lenyomja valamelyik irányító gombot (fel, le, jobbra, balra, space) a játék közben, akkor ez a függvény küldi be az adott gombnyomás tényét a szervernek, hogy az továbbíthassa az összes kliensbe (és ő maga is feldolgozhassa).

public void broadcastKeyPress(PlayerKeyPress playerKeyPress, ClientConnection source)

Szerver módban ezen a függvényen keresztül továbbítja a NetworkController a beérkezett gombnyomást minden kliensbe (kivéve abba, ahonnan maga a gombnyomás érkezett).

ServerTransmitThread

Szerverként való működés esetén minden egyes klienshez létrejön egy ilyen szál, ez a felelős azért, hogy a szükséges adatokat elküldje a hozzá tartozó kliensnek.

private enum ServerState

Szerver mód esetén a program állapotgépként működik, ez a pillanatnyi állapotait leíró felsorolt típus.

public ServerTransmitThread(ObjectOutputStream objectOutputStream, ClientConnection connection, Player player)

Az osztály konstruktora, ami átveszi a már megnyitott ObjectOutputStream-et, és a csatolt klienshez tartozó játékost. Az illető játékosnak/kliensnek küldendő adatokat ez a szál fogja küldeni a kapott kimenő adatfolyamon keresztül.

public void run()

A szál futását megvalósító függvény. Ez a függvény a szerver állapotának megfelelően küldi az információkat a kliensnek.

Amíg a szerver játékindulás előtti állapotban van, a szál a lobby pillanatnyi állapotát küldi periódikusan a kliensnek, tehát a csatlakozott játékosok listáját.

Mikor a játék indul, ez a függvény küldi el a kliensnek a generált pályát, és a pályán elhelyezkedő tankokat.

A játék végén, vagy ha a host játékos már lobby állapotban megszűnteti a szobát, akkor is ez a thread tájékoztatja erről a klienst.

private void sendClosingMessage()

Ez a függvény küldi ki azt az üzenetet, hogy a szerver leállt (akát lobby, akár csata közbeni módban).

private void sendLobbyUpdate()

Ez a függvény küldi a kliensnek a játékra várakozó felhasználók listáját.

private void sendStartingMessage()

Ez a függvény tájékoztatja a klienst a játék indulásáról, és küldi el a generált pályát és a tankokat.

public void stopServer()

Ezzel a függvénnyel lehet leállítani az adott szerver-kliens kapcsolatot.

public void startGame()

A játék elindítására szolgáló függvény, csak az állapotgép állapotát változtatja, majd a főszál ennek megfelelően küldi a szükséges adatokat.

public void sendKeyPress(PlayerKeyPress playerKeyPress)

Ezen a függvényen keresztül lehet egy kliensnek elküldeni egy másik kliensben történt gombnyomást.

public void sendServerStopping()

Ezen a függvényen keresztül lehet a csatlakoztatott kliensnek a szerver leállításáról szóló üzenetet küldeni.

public void sendClientLeave(Message msg)

Ezen a függvényen keresztül lehet a klienst egy másik kliens kilépéről tájékoztatni.

ServerReceiveThread

Szerverként való működés esetén minden egyes klienshez létrejön egy ilyen szál, ez a felelős azért, hogy a szükséges adatokat fogadja a hozzá tartozó klienstől.

public ServerReceiveThread(ObjectInputStream objectInputStream, ClientConnection connection)

Az osztály konstruktora, ami átveszi a már megnyitott ObjectInputStream-et, és a csatolt klienshez tartozó kapcsolatot leíró objektumot. Az illető játékostól/klienstől fogadott adatokat ez a szál fogja fogadni a kapott bejövő adatfolyamon keresztül.

public void run()

A szál futását megvalósító függvény. A szál várakozik bejövő adatra, majd egy fogadott üzenet estén továbbítja a megfelelő osztályoknak a kapott információt.

public void stopServer()

Ezzel a függvénnyel lehet leállítani az adott szerver-kliens kapcsolatot.

ClientReceiveThread

Kliensként való működés esetén létrejön egy ilyen szál, ez a felelős azért, hogy a szükséges adatokat fogadja a szervertől.

public ClientReceiveThread(ObjectInputStream objectInputStream)

Az osztály konstruktora, ami átveszi a már megnyitott ObjectInputStream-et. A szervertől fogadott adatokat ez a szál fogja fogadni a kapott bejövő adatfolyamon keresztül, majd feldolgozza azokat, illetve továbbítja az azokhoz kapcsolódó objektumoknak.

public void run()

A szál futását megvalósító függvény. A szál várakozik bejövő adatra, majd egy fogadott üzenet estén továbbítja a megfelelő osztályoknak a kapott információt.

public void stopReceive()

Ezzel a függvénnyel lehet leállítani az adott szerver-kliens kapcsolatot.

ClientTransmitThread

Kliensként való működés esetén létrejön egy ilyen szál, ez a felelős azért, hogy a szükséges adatokat küldje a szerver felé.

public ClientTransmitThread(ObjectOutputStream objectOutputStream)

Az osztály konstruktora, ami átveszi a már megnyitott ObjectOutputStream-et. Ez a szál fogja elküldeni a szervernek a szükséges adatokat.

public void run()

A szál futását megvalósító függvény. A szál egy FIFO listában tárolja a szervernek küldendő üzeneteket. Mikor a listában van elküldendő üzenet, olyankor elküldi.

public void sendMessage(Message msg)

Ezzel a függvénnyel lehet üzenetet küldeni a szállal a szervernek.

ClientConnection

Minden kliens csatlakozásakor lérejön egy ilyen objektum, ami a kapcsolat adatait írja le. Ez az osztály fogja össze a kliens küldő és fogadó adatfolyamait.

public ClientConnection(Socket socket,

ObjectInputStream objectInputStream, ObjectOutputStream objectOutputStream, Player player)

Az osztály konstruktora, ez veszi át a már megnyitott adatfolyamokat, és a kapcsolt játékost, illetve hozza létre a játékoshoz tartozó küldő és fogadó szálakat.

public void closeConnection()

Ha egy másik kliens kliép a játékból, akkor ezen a függvényen keresztül lehet eltávolítani őt a csatlakozott játékosok közül.

public void stopServer()

Ezzel a függvénnyel lehet leállítani a kliens kapcsolatát, mind az adatok küldését, mind azok fogadását.

Message

A kliensek és a szerver közötti kommunikációt ezek az üzenetek valósítják meg. Minden üzenetnek van egy típusa és egy adattartalma, a szerver vagy a kliens az üzenet fogadása után az üzenet típusa alapján dolgozza fel a benne lévő adatot.

public enum MessageType

Felsorolt típus, az üzenet típusát mutatja meg.

Message(MessageType type, Object data)

Maga az üzenet, aminek van egy típusa, és egy tetszőleges (akár `null`) adattartalma.

BattlefieldBuildData

public BattlefieldBuildData(Battlefield battlefield, ArrayList tanks)

Új játék indulásakor ez a segédosztály használatával lehet a kliensekbe elküldeni a szerveren generált pályafelépítést és tankpozíciókat.

PlayerKeyPress

Segédosztály, ami egy gombnyomást ír le, de hozzárendeli a felhasználót is, akinél a gombnyomás volt.

public PlayerKeyPress(Player player, int key)

Konstrultor, ezen keresztül lehet létrehozni a példányt.

IncomingConnectionHandlerThread

Ez a szál felelős a kívülről érkező kapcsolatok fogadásáért.

public IncomingConnectionHandlerThread()

Konstruktor, ezen keresztül indítható el a kapcsolatok beérkezésének figyelése.

public void run()

A szál futását megvalósító függvény, egy socket-en keresztül bejövő kapcsolatra vár, majd ha van ilyen, létrehoz egy `RoomConnectionHelper` szálat, és átadja neki a socket-et, majd újra figyelni kezd.

public void stopListening()

A bejövő kapcsolatok fogadásának megállítására szolgáló függvény.

RoomConnectionHelper

A bejövő kapcsolatok feldolgozásáért felelős objetkum, szerverként való működés esetén ez csatlakoztatja be a felhasználót a szobába.

RoomConnectionHelper(Socket socket)

Konstruktor, ami a kapott socket-en keresztül becsatlakoztatja a klienst a szobába.

DiscoveryService

A hálózaton elérhető szobák felfedezéséért felelős osztály.

public DiscoveryService()

Konstruktor, csak a szál elindítására használjuk.

public void run()

A szál futását megvalósító függvény.

Klies mód esetén az osztály további függvényeit felhasználva felkutatja a hálózaton elérhető szobákat, majd frissíti a játék ősosztályában a szobalistát.

Szerver mód esetén ez szál küldi ki a szoba adatait a külső (kliens módban futó) programoknak.

private ArrayList checkAvailableNetworks()

A hálózati szobafelfedezéshez ez a függvény vizsgálja meg, hogy a számítógép milyen hálózatokra kapcsolódik.

private ArrayList checkSubNet(String hostAddress)

A hálózati szobafelfedezéshez ez a függvény vizsgálja meg, hogy az adott alhálózaton melyik címeken van elérhető végpont.

private Room checkIfRemoteRoomAvailable(String host)

A hálózati szobafelfedezéshez ez a függvény vizsgálja meg, hogy a már megtalált elérhető végponton van-e aktív, kapcsolódó felhasználókra váró szoba.

public void startDiscovery()

A szobák keresésének elindítására szolgáló függvény.

public void stopDiscovery()

A szobák keresésének leállítására szolgáló függvény.

public void startExternalDiscoveryService()

Szerverként való működés esetén a hálózaton szobákat felfedezni próbáló kliensek kiszolgálásának elindítására szolgáló függvény.

public void stopExternalDiscoveryService()

Szerverként való működés esetén a hálózaton szobákat felfedezni próbáló kliensek kiszolgálásának leállítására szolgáló függvény.