|  |  |
| --- | --- |
|  | **Eötvös Loránd Tudományegyetem**  **Informatikai Kar**  **Programozási Nyelvek és Fordítóprogramok Tanszék** |

„Huszonegy” kártyajáték megvalósítása

**Dr. Tejfel Máté Burkus András**

Egyetemi adjunktus Programtervező informatikus BSc

nappali tagozat

**Budapest, 2018**

Tartalomjegyzék

[1. Bevezetés 1](#_Toc513549759)

[2. Szakdolgozat leírása 2](#_Toc513549760)

[2.1. A játék leírása 2](#_Toc513549761)

[2.1.1. Alapjáték leírása 2](#_Toc513549762)

[2.1.2. A játék menete 2](#_Toc513549763)

[2.1.3. Módosítások az alapjátékhoz képest 3](#_Toc513549764)

[3. Felhasználói dokumentáció 5](#_Toc513549765)

[3.1. Előfeltételek 5](#_Toc513549766)

[3.2. Használat 5](#_Toc513549767)

[3.2.1. Felületek 5](#_Toc513549768)

[3.2.2. Beviteli elemek és megszorításaik 8](#_Toc513549769)

[3.2.3. Fellépő hibák 9](#_Toc513549770)

[4. Fejlesztői dokumentáció 11](#_Toc513549771)

[4.1. Programozási nyelv, fejlesztői környezet 11](#_Toc513549772)

[4.2. Tervezés 11](#_Toc513549773)

[4.3. Szerver 13](#_Toc513549774)

[4.3.1. Konfigurációs fájl 13](#_Toc513549775)

[4.3.2. Fő szerver 14](#_Toc513549776)

[4.3.3. Szál, mint asztal 15](#_Toc513549777)

[4.3.4. Előkészület a játékhoz 16](#_Toc513549778)

[4.3.5. Játék lebonyolítása 18](#_Toc513549779)

[4.3.6. A teljes állapotküldés formája 19](#_Toc513549780)

[4.4. Kliens 20](#_Toc513549781)

[4.4.1. Konfigurációs fájl 21](#_Toc513549782)

[4.4.2. Háttér 21](#_Toc513549783)

[4.4.3. Az üzenet feldolgozása 22](#_Toc513549784)

[4.4.4. Esélyszámolás 23](#_Toc513549785)

[4.4.5. A felületek megvalósítási elve 25](#_Toc513549786)

[4.4.6. Menü felülete 25](#_Toc513549787)

[4.4.7. Csatlakozási felület 25](#_Toc513549788)

[4.4.8. Játéktér 26](#_Toc513549789)

[4.4.9. Kártyalap megjelenítése 29](#_Toc513549790)

[4.4.10. Felugró ablakok 29](#_Toc513549791)

[4.5. Előforduló hibák és azok kezelése 30](#_Toc513549792)

[4.6. Tesztelés 31](#_Toc513549793)

[4.7. Továbbfejlesztési lehetőségek 33](#_Toc513549794)

[5. Összegző 35](#_Toc513549795)

[6. Irodalomjegyzék 36](#_Toc513549796)

# Bevezetés

Napjainkban a különféle kártyajátékok nagy népszerűségnek örvendenek. Számomra is kellemes kikapcsolódást jelent az effajta játékokkal való időtöltés. Ezeknek számos hagyományos asztali, valamint modern változata létezik, melyet számítógépen és egyéb okos készülékeken játszhatunk, például táblagép vagy okostelefon.

Szakdolgozatom témájának egy ilyen játékot, a Blackjacket, másnéven, ahogyan a címben is neveztem, Huszonegy nevű kártyajátékot választottam, mivel ez a kedvenc kártyajátékaim egyike. A játék állandó résztvevője a bank vagy osztó. Az osztó ellen játszhat egy vagy több játékos. Több játékos esetén a játékosok nem egymás ellen játszanak, hanem mindenki az osztó ellen egyénileg. A játékot hagyományosan egy ötvenkét lapos francia kártyával játszák, egyes változatokban akár nyolc pakli is játékban lehet.

A játékalkalmazást kliens-szerver architektúra alapján valósítottam meg, ahol a szerver, azaz a kiszolgáló szerepét az osztó, a kliens, azaz az ügyfél szerepét a játékosok töltik be.

A projekt elkészítése során első lépésként a szervert készítettem el, amellyel párhuzamosan készítettem egy grafikus felület nélküli, szöveges alkalmazást is, amely egy kezdetleges kliensalkalmazásnak felelt meg. A szervert úgy készítettem el, hogy az egyidőben képes legyen több játékasztal működtetésére, valamint több kliens kiszolgálására, ezáltal asztalonként legalább egy, de legfeljebb hat játékos vehet részt a játékban. A játék módja nem függ a játékosok számától.

Amint az osztó képes volt játék lebonyolítására, belefogtam egy grafikus felülettel már rendelkező kliens fejlesztésébe. A felületet egyszerűre és letisztultra, könnyen használhatóra terveztem. Lehetőséget adtam a már játékban nem lévő játékosok számára, hogy megfigyeljék a játék későbbi alakulását. Lehetővé tettem a játékosok és a megfigyelők egymás közötti kommunikációját, amelyet egy chat funkción keresztül valósítottam meg.

A dolgozat dokumentációjában megismertetem az alkalmazás felépítését, a megvalósítását, a működési elvét, a felhasznált technológiákat és az elkészített megoldásokat.

# Szakdolgozat leírása

A következő fejezetben be szeretném mutatni a Huszonegy játék hagyományos, valamint az általam megvalósított változatának a menetét és a szabályait.

## A játék leírása

### Alapjáték leírása

A játéknak többféle változata létezik. A kaszinókban többnyire az egyszerűbb változatokkal lehet találkozni, aminek az oka a limitált fizikai hely. Az internetes változatok lehetőséget adnak eltérő játékvariációk, különböző változatok kombinációjának megvalósítására. Én a hagyományos változatot szeretném bemutatni.

A játékot egy pakli, ötvenkét lapos francia kártyával játszák. A lapok értékei a következőképpen alakulnak: a számmal ellátott lapok értéke a rajtuk található szám; a bubi, a dáma és a király értékei tíz; az ász értéke lehet egy vagy tizenegy, attól függően, hogy melyik a kedvezőbb a kettő közül.

### A játék menete

Tovább folytatva a szót az alapjátékról, a játékot körökre bontva játszák, a köröket tétre játszák. Egy leosztás egy körnek felel meg. Egy leosztáson belül a cél az, hogy a játékosnál lévő kártyák értékeinek összege nagyobb legyen, mint az osztó kártyáinak az összértéke, de mindemellett ne lépje túl a huszonegyet.

A kör kezdetén a játékosok megteszik a tétjeiket. Általában a játék valamennyi változatánál meg van határozva egy minimum összegű tét, egyes változatokban a maximum is meg van határozva.

Miután minden játékos megtette a tétjét, az osztó minden résztvevőnek, önmagát is beleértve, oszt egy lapot a színével felfelé. Ezután oszt még egy lapot mindenkinek, de, önmagát kivéve, a játékosok lapjait ismét a színével felfelé osztja le. Az osztó a második lapját önmagának a színével lefelé osztja le, ennek az értéke rejtve marad egy ideig a játékosok elől.

Amint minden résztvevő birtokába került a két lap, elkezdődnek a játékosok lépései, melyek a következők lehetnek: lap kérése, megállás, duplázás, kettéosztás, biztosítás, feladás; egy játékos a lépéseit a megállás vagy feladás választásával fejezheti be.

* Lap kérése: ha a játékos ezt választja, kap egy új lapot, amennyiben a lapjai értékének az összege nem haladta meg a huszonegyet, ismét ő következik.
* Megállás: ha a játékos úgy dönt, hogy megáll, akkor a meglévő lapjaival folytatja a játékot és nem kérhet több lapot a továbbiakban.
* Duplázás: ha a játékos úgy látja az első két lapja alapján, hogy három lappal megnyerheti a játékot, akkor kérheti a duplázást, ekkor a tétjét megduplázza és kap még egy lapot, utána nem kérhet már több lapot.
* Kettéosztás: ha a játékos első két lapja csak a színében tér el, azaz párt alkot, akkor a játékos kérheti ezek kettéosztását és két külön kézzel játszik a továbbiakban, ami a tét megduplázásával is jár és mindkét kézhez kérhet tetszőleges számú lapot.
* Biztosítás: ha az osztó felfelé fordított lapja ász, a játékos kérheti a biztosítást, amely lényege az, hogy ha az osztó másik lapjának az értéke tíz, akkor a játékos visszakapja a feltett tétje felét; amennyiben a második lap értéke tíztől különböző, úgy minden esetben az osztó nyer.
* Feladás: ha a játékos a leosztott két lapja alapján úgy véli, hogy jobban jár a feladással, akkor megteheti ezt, ennek során a feltett tétjének a felét kapja vissza.

Ha a játékos lapjainak az összértéke meghaladta a huszonegyet, akkor a játékos elvesztette a kört.

A játékosok lépéseinek befejeztével következik az osztó, aki ekkor felfedi a második, eddig lefelé fordított lapját és a lap kérése, valamint a megállás közül választhat. A stratégiája függ a játékosok számától és azok lapjaitól, de mindig egyre törekszik: pozitív egyenleggel zárja a kört.

### Módosítások az alapjátékhoz képest

Munkámban a játék egy absztrakt verzióját szerettem volna létrehozni, melyben bárki könnyen eligazodhat és gyorsan megértheti, anélkül, hogy számolnia kellene közben bármit is. A megvalósítás során kihagytam a duplázás, kettéosztás, biztosítás, valamint a feladás lehetőségét, tehát a játékosok és az osztó egyaránt a lap kérése és a megállás közül választhatnak.

A játék elején minden játékos megegyező számú zsetont kap. Az osztó a játékosok kezdetleges számának a függvényében kapja meg a zsetonjait, megoldva azt a problémát, hogy kevesebb számú játékos nehezebben foszthatja ki a bankot, ezzel előteremtve az esélyegyenlőséget a játékosok körében. Ugyanis az általam alkotott játék fő célja, hogy a játékosok elnyerjék az osztó összes zsetonját.

# Felhasználói dokumentáció

Ez a fejezet a felhasználók számára készült. Az olvasó megismerheti a kliensalkalmazás különböző felületeit és funkcióit, azok célját, valamint különböző megszorításait.

## Előfeltételek

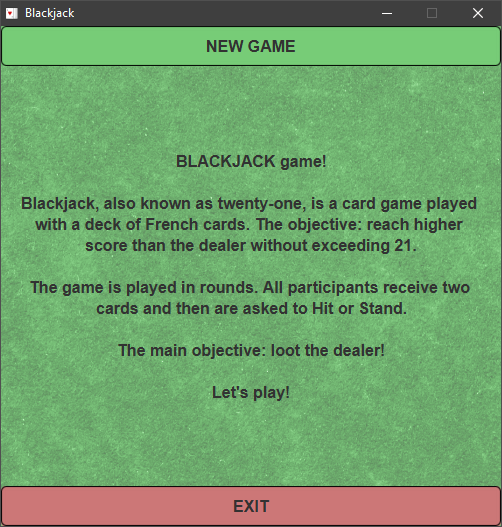
Az alkalmazás futtatásához szükség van Java SE Runtime Environment legalább 8-as verziójára, ugyanis ettől a verziótól kezdve része a JavaFX a Java SE API-nak, mely segítségével megvalósítottam a kliensalkalmazást. A játék használata nincs operációs rendszerhez kötve a Java programozási nyelvnek köszönhetően.

## Használat

A játékalkalmazás elindításához szükség van egy *client\_config.properties* nevű fájlra, amely a szerver eléréséhez szükséges adatokat tartalmazza. Ennek meglétében a szokásos módon el tudjuk indítani Blackjack.jar nevű futtatható állományt.

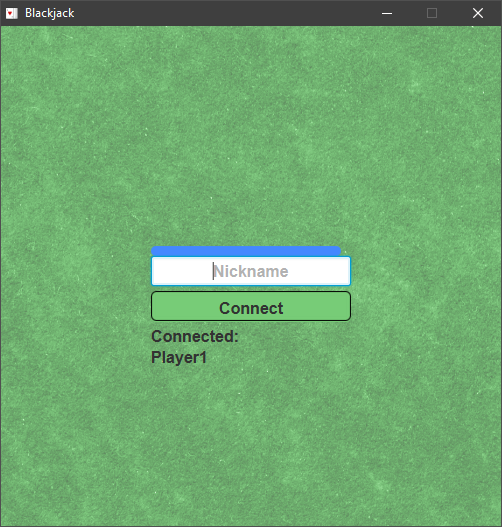
### Felületek

A játék három különböző felületet jeleníthet meg: menü, kapcsolódási felület, játékfelület. Egyidőben csak egy felület látható, illetve használható.



1. ábra. Menü

A játékot elindítva a menü (1. ábra) tárul a játékos elé. A menü három elemet tartalmaz: egy „NEW GAME” feliratú gombot, mellyel csatlakozni lehet a szerverhez és ezzel elindul a játék előkészítése; egy rövid leírást a játékról és annak menetéről; egy „EXIT” feliratú gombot, mellyel ki lehet lépni a játékból.



2. ábra. Csatlakozási felület

A menüben a „NEW GAME” feliratú gomb megnyomásával juthatunk a következő felületre (2. ábra), ekkorra már megtörtént a kapcsolatfelvétel a szerverrel. Ezen a felületen található egy szöveg bevitelére alkalmas mező, ahova a játékos által választott felhasználónevet kell beírni. A beviteli mező felett található egy sáv, amely mutatja a játékos számára a hátralévő időt a bevitelre. A beviteli mező alatt található egy „Connect” feliratú gomb, ezzel a gombbal küldhető el a kiválasztott név. Legalul található egy lista a már csatlakozott játékosok neveivel.



3. ábra. Játéktér játék közben

Amint minden játékos elküldte a kiválasztott nevet, megjelenik a játéktér (3. ábra) felülete. Ezt a felületet három fő részre oszthatjuk:

* a bal felső sarokban találhatjuk a chat funkció felületét, amelyhez tartozik egy, a korábbi üzenetek megjelenítésére alkalmas felület, egy beviteli mező, ahova a kívánt üzenetet írhatjuk, valamint egy üzenetet elküldésére szolgáló gomb.
* a bal alsó sarokban található az akciópanel. Ennek a panelnek két funkciója is van. Egyrészt a kör elején megjelenik itt egy felület, amelyen található: egy már korábbról ismert sáv, amely a hátralévő időt jelzi; egy csúszka, amellyel a tét mennyiségét szabályozhatjuk; egy beviteli mező, amellyel szintét a tét mennyiségét szabályozhatjuk; egy „Bet” feliratú gomb, amellyel a választott tétet elküldhetjük; egy „Skip the round” feliratú gomb, amellyel azt választhatjuk, hogy nem kívánunk részt venni az adott körben. Másrészt a játék további részében, amikor a játékos kerül sorra, megjelenik egy felület, amelyen található: egy hátralévő időt jelző sáv; egy „Hit” feliratú gomb, mellyel lapot lehet kérni, továbbá a gombon megjelenik annak a mértéke is, hogy hány százalék esély van egy olyan lapra, amellyel a játékos lapjainak összértéke nem lépi túl a huszonegyet; egy „Stand” feliratú gomb, mellyel be lehet fejezni a lépést, tehát a játékos nem kér több lapot.
* a jobb oldalon található a játékasztal. A bal felső sarokban található egy jelölőnégyzet „Help” felirattal, amelynek az a szerepe, hogy a játékos számára tippeket kínáljon fel. Az asztal felső részén található az osztó területe, amelyen a következő információk találhatóak: az osztó zsetonegyenlege, az osztó lapjai, valamint az osztó lapjainak az összértéke. Az asztal alsó részén találhatóak a játékosok területei, minden játékos saját területtel rendelkezik és a következő információk jelennek meg rajta: a feltett tét, a játékos neve, a zsetonegyenleg, a játékos lapjainak összértéké és végül a játékos lapjai. A játékos a saját területét zöld színű háttérrel, míg a többi játékos területét bordó színű háttérrel látja.

A következő táblázatban található egy leírás a játékos területén megjelenő ikonokról:

|  |  |
| --- | --- |
|  | Ez az ikon jelöli a soron következő játékost. Az adott játékos területe felett jelenik meg a tétrakás és a lépések alatt is. |
|  | Ez az ikon jelzi a játékos által feltett tét mennyiségét. |
|  | Ez az ikon jelzi a játékos zsetonegyenlegét. Tétrakásnál a feltett tét nem lehet magasabb a játékos zsetonegyenlegénél. |
|  | Ez az ikon jelzi a játékos lapjainak az összértékét. Amennyiben a játékos lapjai között található ász is és ezáltal több összérték van, amely nem lépi túl a huszonegyet, akkor ezek közül valamennyi megjelenik. |

### Beviteli elemek és megszorításaik

A használat során a játékos észre veheti, hogy bizonyos beviteli mezők esetén korlátozva vannak a beviteli lehetőségeik.

A kapcsolódási felületen található a becenév megadására alkalmas beviteli mező. A becenév legalább három, de legfeljebb tíz karakter lehet, valamint az angol ábécé kis- és nagy betűit, illetve az arab számjegyeket tartalmazhatja. Az Enter billentyű és a „Connect” feliratú gomb megnyomásával ugyanazt a hatást lehet elérni.

A chat beviteli mező tetszőleges karakter bevitelére alkalmas. Az Enter billentyű, a becenév megadásához hasonlóan, ugyanazt a hatást éri el, mint a boríték ikonnal ellátott, üzenet küldése gomb.

A tét rakása esetén a csúszka alatt található beviteli mező alkalmas billentyűzetről való bevitelre. A mező csak arab számokat engedélyez. A mező és a csúszka szinkronban vannak, tehát amint az egyik elem értékén változás történt, a másik elem értéke is megváltozik. Ha a mezőbe bevitt érték nagyobb, mint a tehető maximum tét, akkor az érték automatikusan a legmagasabb tehető tét értéke lesz.

Amíg a „Help” feliratú jelölőnégyzet be van jelölve, addig a segítség funkció aktív, ezáltal a játékos kap némi segítséget: amikor a játékoson a sor a lapkérés tekintetében a „Hit” feliratú gombon megjelenik annak az esélye, hogy a következő lappal a játékos lapjainak összértéke nem haladja túl a huszonegyet. Ha a játékos esélye legfeljebb 30% (harminc százalék) és a segítség funkció aktív, felugrik egy jóváhagyó ablak, mely feltesz a játékos számára egy megerősítő kérdést meggyőződve a játékos lépésének biztosságáról.

### Fellépő hibák

A játék elkészítése során ügyeltem a beviteli pontoknál a váratlan bevitelek elkerülésére. A játék futása során váratlan hiba keletkezhet a szerverrel való kapcsolatteremtés során, valamint annak elvesztése esetén.

A kapcsolódás során előfordulhat, hogy a kliens nem tud csatlakozni a szerverhez, ennek oka, hogy a szerver eléréséhez szükséges információkat tartalmazó fájl adatai hibásak vagy a megadott címen nem található szerver. Ezt az alkalmazás egy üzenettel jelzi a felhasználó számára.

A kapcsolat elvesztése esetén a játékos számára sajnos véget ért a játék, amiről az alkalmazás szintén egy üzenet formájában tájékoztatja a játékost.

# Fejlesztői dokumentáció

## Programozási nyelv, fejlesztői környezet

Játékalkalmazásom elkészítéséhez a Java alapú technológiákat használtam fel. A Java programozási nyelvet az Oracle Corporation nevű, amerikai székhelyű cég fejleszti. A Java nyelvvel az egyetemi oktatás során ismerkedhettem meg a következő tantárgyak keretein belül: Programozási nyelvek II. JAVA, Programozási technológia 1., Programozási technológia 2.

A szerveralkalmazásomat a Java SE (Java Standard Edition) 8-as verziója felhasználásával készítettem el. A szerver a kliens által webes, HTTP protokollon keresztül érhető el. A kommunikációt socket-eksegítségével tettem lehetővé.

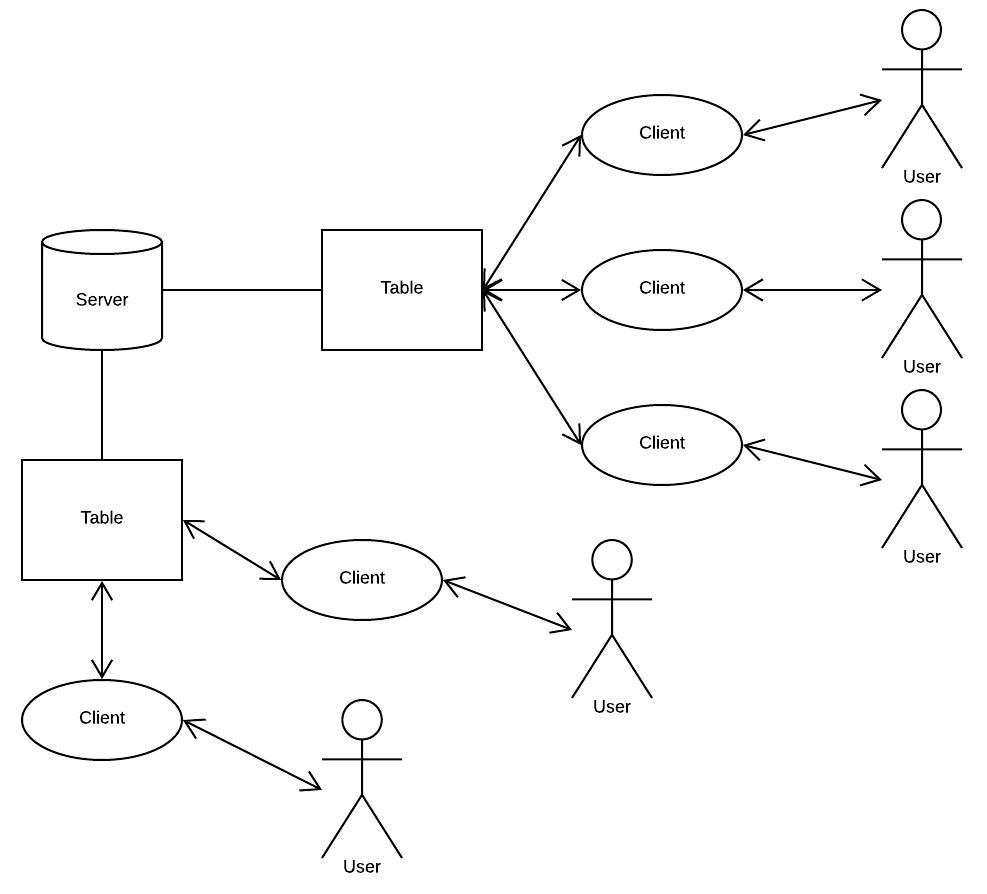
A kliensalkalmazást egy Java alapú API (alkalmazásprogramozási felület, angolul application programming interface), a JavaFX 8-as verziója segítségével valósítottam meg. A JavaFX grafikai és médiacsomagok készlete lehetővé teszi a fejlesztők számára különböző platformokon működő kliensalkalmazások létrehozását.

A fejlesztés során a Visual Studio Code kódszerkesztőt használtam, melyet a Microsoft Corporation, amerikai székhelyű cég fejleszt. A gördülékeny, gyors és hatékony fejlesztés érdekében a Visual Studio Code alkalmazás fejlesztői lehetővé tették a különböző bővítmények használatát, ennek köszönhetően használhattam a Java Extension Pack nevű bővítményt.

## Tervezés

A tervezés során felállítottam néhány követelményt, melynek a munkámnak meg kellett felelnie:

* az elkészült munkában legyen egy elkülönített szerver és egy kliens, melyek külön-külön használhatatlanok, de együtt egy teljesértékű játéknak feleljenek meg.
* a szerver paraméterezhető legyen, azaz képes legyen különböző beállításokkal is elindulni, mely beállításokat egy konfigurációs fájlból olvasson be.
* a szerver egyszerre több játék lebonyolítására is képes legyen.
* egy játékon belül egyszerre több játékos is tudjon játszani.
* szálak használata, úgy a szerver, mint a kliens oldalon, azaz egyszerre több funkció is használható legyen.
* a játékosok tudjanak kommunikálni egymással.
* a kliensnek rendelkezni kell egy grafikus felülettel.
* a grafikus felület jól átlátható és könnyen kezelhető legyen.
* a játék adatait, információit, állását a szerver tárolja, ezáltal védett legyen a kliensen keresztül érkező csalási kísérletek ellen.
* a kliens csak a számára fontos információkat tárolja.
* a felhasználó elől elrejtett adatok ténylegesen rejtve legyen, azaz a szerver csak a szükséges adatokat közölje, például a szerver ne küldje el az osztó lefelé fordított lapja kilétét.
* a beviteli mezők csak a szükséges karaktereket engedélyezzék.



4. ábra. Szerver-kliens-felhasználó kapcsolat

## Szerver

A játékalkalmazás egyik fő része a szerver. A kliensalkalmazás önmagában nem képes játék lebonyolítására, elengedhetetlen egy szerver is a használatához. A szervert úgy készítettem el, hogy képes legyen egyidőben párhuzamosan több játékasztalt is működtetni, emellett egy játékasztal több játékos fogadására is alkalmas legyen.

Az alkalmazás kódja *.java* kiterjesztésű fájlokban van eltárolva. Ebből a kódból az alkalmazás főkönyvtárából futtatva a következő parancssori utasítással lehet Java bájtkódot generálni:

javac blackjackserver/BlackjackServer.java

A kapott bájtkódokat tartalmazó *.class* kiterjesztésű fájlokból a következő paranccsal lehet létrehozni futtatható *.jar* kiterjesztésű tömörített állományt:

jar cfe Blackjack\_server.jar blackjackserver.BlackjackServer blackjackserver/\*.class

A bájtkód generálásához, valamint a tömörített állomány elkészítéséhez szükség van a Java SE Development Kit legalább 8-as verziójára.

A szerveralkalmazás futtatásához a következő utasítást kell bevinni a parancssorba:

java -jar Blackjack\_server.jar <konfigurációs fájl elérési útvonala - opcionális>

Amennyiben a konfigurációs fájl elérési útvonala nincs megadva, úgy a szerver alapértelmezetten a *default\_server.properties* nevű fájlt fogja használni, melynek hiányában az alkalmazás egy hibaüzenetet ír ki a parancssorra, majd kilép. Ennek a futtatásához a Java SE Runtime Enviromnent legalább 8-as verziójára van szükség, valamint szükséges egy konfigurációs fájl megléte is.

### Konfigurációs fájl

A fejlesztés során lehetővé tettem a szerver külső paraméterezhetőségét. Ennek megoldásaként a szerver elindításához szükségessé tettem egy *.properties* kiterjesztésű fájl jelenlétét is. Ebben a konfigurációs fájlban hét adatnak kell szerepelni, melyek tetszőleges sorrendben lehetnek megadva a következő megkötésekkel: minden adat új sorban kezdődik és *kulcs=érték* formában van megadva. A szerver használatához szükséges adatok:

* name: a konfiguráció neve.
* port: egy portszám, melyre a szerver várja a kliensek csatlakozását a játékhoz való kommunikáció megteremtéséhez.
* chatport: egy portszám, melyre a szerver várja a kliensek csatlakozását a chathez tartozó kommunikáció megteremtéséhez.
* maxplayer: egy egész szám, kötelezően legalább egy, de legfeljebb hat, mely meghatározza, hogy egy asztalhoz hány játékos csatlakozhat.
* startmoney: egy pozitív egész szám, mely meghatározza, hogy a játékosok milyen zsetonmennyiséggel kezdik a játékot.
* multiplier: egy pozitív egész szám, tulajdonképpen egy nehézségi mutató, amely az osztó kezdeti zsetonmennyiségét határozza meg és a következő adatok szorzata alapján áll elő: játékosok száma, játékosok kezdeti zsetonmennyisége, nehézségi szorzó (multiplier).
* starttimeout: egy pozitív egész szám, mely meghatározza azt, hogy az első játékos csatlakozásától számított hány másodperc leteltével induljon el a játék, amennyiben a játékosok száma még nem érte el a maximum limitet.

Példa egy helyes konfigurációs fájl tartalmára:

name=config1

port=1234

chatport=1235

maxplayer=5

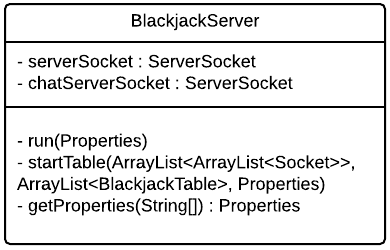
startmoney=2000

multiplier=5

starttimeout=300

Az alkalmazás nem ellenőrzi a konfigurációs fájl helyességét.

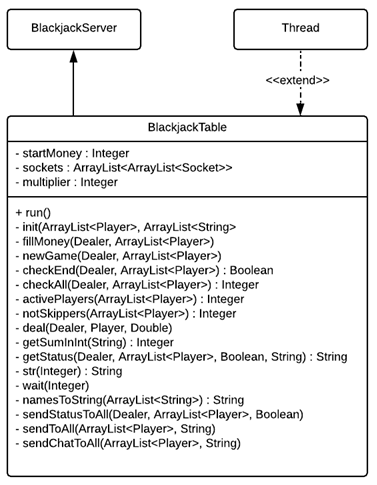
### Fő szerver



5. ábra. BlackjackServer osztály UML diagramja

A fő szerver négy feladatot lát el. Induláskor beolvassa a konfiguráció adatait, majd ezek alapján elindít két socket szervert: egyiken a játékkal kapcsolatosinformációk, a másikon pedig a chat üzenetek folynak. Továbbá kliensek csatlakozását fogadja, valamint elegendő kliens csatlakozása vagy a csatlakozásra adott idő lejárta után elindít egy játékasztalt, melyet egy szál reprezentál.

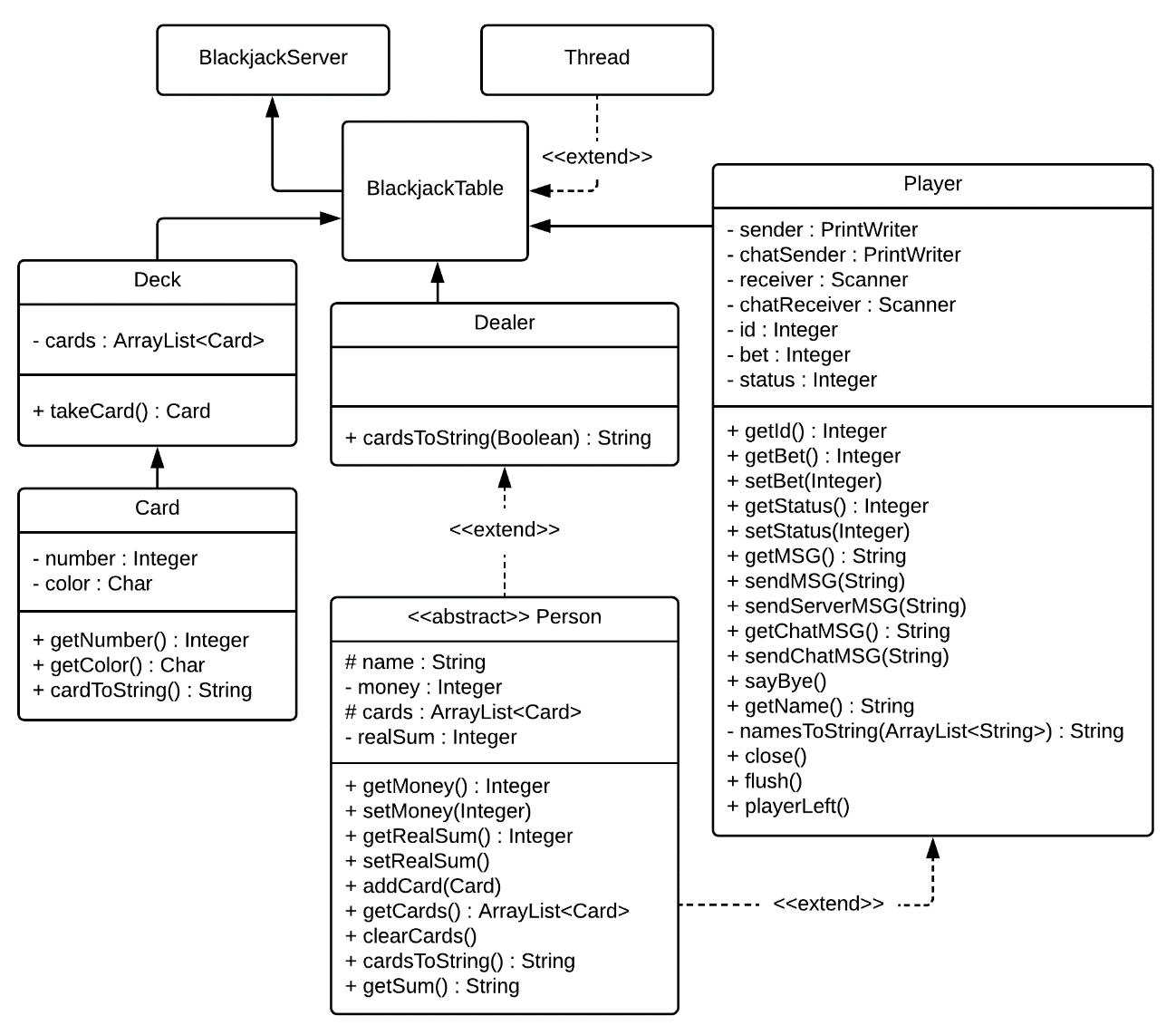
### Szál, mint asztal



6. ábra. BlackjackTable osztály UML diagramja

Több játék párhuzamos kezelése érdekében minden játékasztal külön szálon indul el. Egy játék alatt újabb szálak indulnak a folyamatos használat és a gördülékeny futás érdekében, melyek lehetővé teszik több funkció egyidejű használatát, mint például játék közbeni chat beszélgetés. Amint az adott asztalon véget ért a játék, a szál, valamint az elindított újabb szálak megszűnnek létezni.

### Előkészület a játékhoz



7. ábra. A teljes szerveralkalmazás UML diagramja

A BlackjackTable osztálynak, mely a játék lebonyolítását végzi, a konstruktora két paramétert vár: első a kapcsolódott kliensek socket-jeinek listája, a második a játékasztal kezdeti értékei. A konstruktor lefutása után a fő szerver elindítja a szálát, ezzel elindul a játék, melynek első lépése az előkészület.

Először egy Dealer típusú objektum jön létre, mely a Person ősosztályból van származtatva. A továbbiakban ez fogja kitölteni az osztó szerepét. A Person osztály egy résztvevő alapvető adatait, valamint az azokhoz tartozó metódusokat írja le. Ilyen adat például a résztvevő birtokában lévő lapok listája vagy a résztvevőnél lévő zsetonmennyiség.

Az osztó létrejötte után meghívódik az init nevű, visszatérési érték nélküli metódus, mely paraméterei a már létrehozott Player típusú objektumok egy listában tárolva, valamint ezen objektumok *name* attribútumai, azaz a játékosok nevei szintén egy listában tárolva. A Player objektum a játékos alapvető adatait tárolja, mint például a socket kapcsolatai. A nevek külön eltárolására azért van szükség, mert egy játékasztalnál nem játszhat két azonos nevű játékos és az ellenőrzést ez a lista alapján végzi a kliens. A konstruktor három adatot vár: a játékos két socket kapcsolatát, egy azonosítót és a már létrehozott játékosok neveit tartalmazó listát. A metódus számon tartja a már létrehozott játékos objektumok számát, ez alapján kapja minden játékos az azonosítóját. Az objektumok létrehozása a kapcsolódási sorrend alapján történik.

Amint az összes játékos objektuma létrejött, minden kliens megkapja az egyedi azonosítóját üzenetben, valamint létrejön mindenki számára egy-egy szál a chat-szolgáltatása üzemeltetése céljából. Ezek után minden játékos, valamint az osztó megkapja a kezdeti zsetonjait és indulhat a játék.

### Játék lebonyolítása

A játék levezetése körökre bontva történik. Egy kör lebonyolítását egy elöltesztelős ciklus magjának egyszeri lefutása a hajtja végre. Az aktuális kör abban az esetben indul, ha teljesülnek az előfeltételei: legyen még aktív játékos, tehát legyen legalább egy olyan játékos az asztalnál, akinek van még felhasználható zsetonegyenlege; az osztó zsetonegyenlege nullától nagyobb legyen. Amennyiben a két feltétel közül legalább egy nem teljesül, a játék véget ért.

A kör első lépése egy új kártyapakli létrehozása, melyet egy Deck típusú objektum reprezentál. A Deck osztálynak egy adattagja van: egy Card típusú objektumokat tartalmazó lista. A Deck konstruktora meghívása során összeállít egy ötvenkét lapból álló kártyapaklit, majd azokat összekeveri véletlenszerűen. Egy Card objektumnak két adattagja van: a színe és a magassága.

Következő lépésként következik a játékosok tétrakása, mely során a szerver egy számot tartalmazó szöveges üzenetet vár a klienstől. A tétet elmenti a játékos objektumán található *bet* adattagba a *setBet* metódusa segítségével.

A tétrakás után az osztó két lapot oszt minden résztvevő számára, a saját lapjai közül csak az egyiket közli a játékosokkal, a másik lapjának színe és magassága rejtve marad. Ekkor ismét a játékosokon a sor. Egy játékos lépéseinek két dolog vethet véget: a lapjainak az összértéke elérte vagy túllépte a huszonegyet; a játékos a megállás (*„Stand”*) mellett döntött. A lépés a következőképpen néz ki: a kliens közli a játékossal a választási lehetőséget a lap kérése (*„Hit”*) és a megállás között, ezek után vár a játékos döntésére; ha a játékos a lap kérése mellett dönt, a kliens ezt közli a szerverrel, majd az osztó oszt egy új lapot, ezután, amennyiben még a feltételek engedik a lépést, a játékos újra dönthet.

A játékos lépései után következik az osztó, mely a következő logikát követi:

* ha valamennyi játékos túllépte a huszonegyet, az osztónak nincs további dolga, megnyerte a kört.
* ha az osztó lapjainak összértéke legalább tizenhét, akkor az esély jó lapra nagyon alacsony, ezért az osztó nem kockáztat és véget vet a körnek, attól függetlenül, hogy nyereséges-e az adott kör.
* ha az osztó lapjainak az összértéke még nem érte el a tizenhetet, minden lépése előtt mérlegel, elvégez egy számítást: egy szám típusú változóba összesíti minden egyes játékos alapján, hogy adott állásnál nőni vagy csökkeni fog az egyenlege. A módszer az, hogy összeveti a saját lapjainak összértékét az adott játékos lapjainak összértékével: ha veszítene a játékossal szemben, levonja az összesített értékből a játékos által feltett tétet, viszont, ha nyerne, akkor hozzáadja a feltett tétet. Ha a számítás végén az összesített érték nagyobb nullánál, akkor megáll, azaz nyereséges számára a kör.

### A teljes állapotküldés formája

A szerver az információkat szöveges üzenetek formájában küldi a kliensnek. Az üzenetet elkészítő függvény tervezése során gondolnom kellett a tagolhatóságra, ugyanis a kliensnek a játéktér legenerálásához ezt az üzenetet kell visszafejteni.

Az üzenet két fő részre bontható: az első fele az osztó adatait tárolja, míg a második a játékosok adatait. Ezt a két részt egy kukac karakter választja el.

Az osztó következő adatait tartalmazza az üzenet ebben a sorrendben pontosvessző karakterrel elválasztva:

* az osztó zsetonegyenlege,
* az osztó birtokában lévő lapok szöveges leírása szóközzel elválasztva,
* az osztó lapjainak összértékei,
* az osztó lapjainak hasznos összértéke.

Egy lap szöveges leírását a Card osztály *cardToString* metódusának meghívásával lehet megkapni, amely a lap színéből és magasságából áll egy kettőspont karakterrel elválasztva. Lapok összértékei: az összes lehetséges összérték, amely nem haladja meg a huszonegyet, amennyiben létezik ilyen, különben a legkisebb huszonegyet meghaladó érték. A különböző összértékek akkor fordulhatnak elő, ha a számolt lapok között szerepel legalább egy ász. Lapok hasznos összértéke: az összes lehetséges összérték közül a legnagyobb, mely nem haladta meg a huszonegyet, amennyiben létezik ilyen, különben, az előzőhöz hasonlóan a legkisebb huszonegyet meghaladó összérték.

Az üzenet második fele a játékosok adatait tárolja. A különböző játékosok adatait kettőskereszt karakter választja el. Egy játékos adatai a következő sorrendben jelennek meg pontosvessző karakterrel elválasztva:

* a játékos azonosítója,
* a játékos neve,
* a játékos státusza,
* a játékos zsetonegyenlege,
* a játékos által feltett tét,
* a játékos lapjai szövegesen leírva, szóközökkel elválasztva,
* a játékos lapjainak összértékei,
* a játékos lapjainak hasznos összértéke.

A játékos státuszai között öt státuszt lehet elkülöníteni: 0 – a játékos lépés előtt áll, de még nem ő következik; 1 – az aktuális soron következő játékos; 2 – a játékos lépése már befejeződött; 3 – a játékos kihagyja az adott kört; 4 – a játékos már kiesett, de még jelen van, mint megfigyelő. A 0, 1 és 2 státusz úgy tétrakásnál, mint a lapkérés időszaka alatt érvényesek.

## Kliens

A kliensalkalmazás a játék másik fontos alkotóeleme. Fő feladata, hogy megteremtse, valamint vezényelje a kommunikációt a felhasználó és a szerver között. A szervertől kapott információkat egy grafikus felületen közli a játékossal és ugyanezen a felületen lehetőséget ad a válaszadásra.

Az kliensalkalmazás kódja szintén *.java* kiterjesztésű fájlokban van eltárolva. Az alkalmazás főkönyvtárából futtatva a következő parancssori utasítással lehet Java bájtkódot generálni:

javac blackjackclient/BlackjackClient.java

A kapott bájtkódokat tartalmazó *.class* kiterjesztésű fájlokból a következő paranccsal lehet létrehozni futtatható *.jar* kiterjesztésű tömörített állományt:

jar cfe Blackjack.jar blackjackclient.BlackjackClient blackjackclient/\*.class blackjackclient/media/\*

Az alkalmazást el tudjuk indítani parancssoron keresztül is a következő utasítással:

java -jar Blackjack.jar

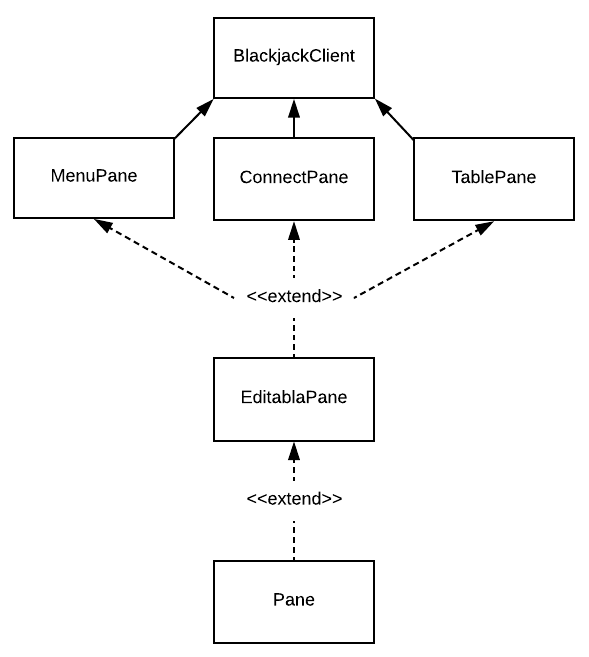
### Konfigurációs fájl

A szerveralkalmazáshoz hasonlóan a kliensalkalmazás elindításához is szükség van egy *.properties* kiterjesztésű konfigurációs fájlra, viszont a kliens esetében mindössze három adatra van szükség. Az adatokat külön sorokba írva *kulcs=érték* formában kell megadni. Az adatok a következők:

* host: a szerver eléréséhez szükséges webcím.
* port: egy portszám, melyre a szerver várja a kliensek csatlakozását a játékhoz való kommunikáció megteremtéseként.
* chatport: egy portszám, melyre a szerver várja a kliensek csatlakozását a chathez tartozó kommunikáció megteremtéseként.

A konfigurációs fájl hiányában parancssori indítás esetén kiír egy hibaüzenetet a parancssorra és leáll. Egyéb indítás esetén nem értesíti a felhasználót a fájl hiányáról.

### Háttér



8. ábra. A kliensalkalmazás osztályai.

A kliensalkalmazás működése egyszerű: induláskor beolvassa a konfigurációs fájlban található adatokat, létrehozza a menü felületét, ahol egy gombnyomással lehet csatlakozni a szerverhez. Miután ez megtörtént, a kliens egy dolgot tesz: üzenetet vár a szervertől és az üzenet tartalma alapján létrehoz egy felületet, melyet megjelenít a játékos számára. Az üzenetből megállapítható, hogy a szerver vár-e választ a játékostól: ha vár, akkor a kliens által megjelenített felületen egyértelmű kérdés, egyéb esetben a kliens küld egy visszaigazoló üzenetet a szerver számára.

### Az üzenet feldolgozása

Az üzenet minden esetben egy hat karakterből álló előtaggal kezdődik, mely meghatározza az üzenet típusát. Az előtagok mindig aláhúzás karakterrel kezdődnek és a következők lehetnek:

* \_gtnm\_: a szerver a játékos becenevét várja, azaz megjelenik a kapcsolódási felület és elindul az időkorlát visszaszámlálója. Az üzenet további részében a már kapcsolódott játékosok nevei vannak felsorolva kettőskereszt karakterrel elválasztva.
* \_nms\_\_: a már csatlakozott játékosok kapják ezt az üzenetet minden újabb játékos csatlakozása esetén és a kliens ezeknek a listáját jeleníti meg a játékos számára.
* \_ynm\_\_: minden játékos megkapja ezt az üzenetet a csatlakozása után. Az üzenet a játékos becenevével folytatódik. Erre azért van külön szükség, mert amennyiben a játékos nem adja meg a becenevét, akkor a szerver létrehoz egyet számára, mely a „Player” szóval kezdődik és egy számmal végződik, ami a játékos sorszáma.
* \_id\_\_\_: a szerver ezzel az üzenettel közli a játékossal az azonosítóját. Az üzenet a játékos azonosítójával folytatódik.
* \_svms\_: a szerver ezzel az üzenettel fontos információkat közölhet a játékossal, mint például kör végi eredmény. Az üzenet tehát egy közleménnyel folytatódik.
* \_strt\_: a kliens ennek az üzenetnek a hatására hozzá létre a játéktér felületét. Az üzenetnek nincs további folytatása.
* \_stat\_: a szerver ezzel az üzenettel közli az játékasztal aktuális állapotát, a kliens ennek hatására frissíti a játéktér felületét. Az üzenet további részében az osztó, valamint a játékosok adatai találhatóak tagoltan.
* \_bet\_\_: a tétrakás alkalmával a soron következő játékos kapja meg ezt az üzenetet, a szerver a tét összegét várja válaszban. Ennek az üzenetnek a hatására megjelenik a tétrakás felülete. Az üzenet további részében a játékasztal állása található.
* \_turn\_: a játékosok lépései, azaz lap kérése vagy megállás, alkalmával a soron következő játékos kapja meg ezt az üzenetet, a szerver a játékos lépését várja válaszban. Ennek az üzenetnek a hatására megjelenik az akciópanel felülete. Az üzenet további részében a játékasztal állása található.
* \_chat\_: a szerver ezzel az üzenettel közli a játékosokkal, ha új chat üzenet érkezik. Az üzenet további részében az üzenet küldőjének beceneve és üzenete található. A becenév kapcsos zárójelek közé van téve.
* \_bye\_\_: a kliens akkor kapja ezt az üzenetet, ha véget ért a játék az adott asztalon, tehát minden játékos vagy az osztó elveszítette az összes zsetonját.

A kliens a listában szereplő üzenetek közül az utolsó kettő kivételével mindegyikre küld egy válaszüzenetet. Amennyiben a szerver nem vár fontos üzenetet a játékostól, mint például annak beceneve vagy tétje, akkor a kliens csak egy üres üzenetet küld, jelezve, hogy megkapta a szerver üzenetét. Ilyen eset lehet például, ha a szerver az asztal aktuális állapotát közli.

### Esélyszámolás

A lapkérés során a kliens segítséget nyújt a játékos számára, amennyiben a játékos ezt igényli. Ennek az igénynek a kérését a játékos egy jelölőnégyzet megjelölésével teheti meg. A segítségnyújtás lényege, hogy a kliens jelzi a játékos számára, hogy hány százalék az esély arra, hogy a következő húzott lappal a játékosnál lévő lapok összértéke nem haladja meg a huszonegyet. Amennyiben a játékos lapjainak több összértéke lehet, azaz van a játékos lapjai között legalább egy ász, akkor a kliens a legkisebb összérték alapján számolja az esélyt.

Az esélyszámolása kliens által ismert lapok alapján történik, azaz az osztó lefelé fordított lapja nem befolyásolja ezt, ugyanis a szerver ennek kilétét nem közli a kliens felé, amíg nem szükséges. Az ász értéke az esélyszámolás során végig egy. Ennek oka, hogy a legtöbb esetben azt egynek számolva az összérték még nem lépi át a huszonegyet, viszont tizenegynek számolva már igen. A lapok színe, illetve magassága nem játszik szerepet ebben, csak az értéke, tehát az azonos értékű lapokat egyformán lehet kezelni.

Az algoritmus elején létrejön egy tíz elemű tömb. Az elemszám azért tíz, mert tíz különböző értékű lap van a játékban, ennek megfelelően a tömbben az elem indexe a lap értékétől egyel kevesebb, például az ász értéke egy, indexe nulla. A tömböt feltölti a különböző lapok számával: tíz értékűből tizenhat darab, így a kilencedik elem tizenhat, a többi esetén négy darab. Összegyűjti az asztalon lévő ismert lapokat, kigyűjti ezek értékét és kivonja a darabszámot a tömbben az értéknek megfelelő elemből. A tömbből kigyűjti azon lapok számát, melyekkel a játékos lapjainak összértéke nem lépi túl a huszonegyet, valamint az összes , még nem ismert lap számát. A megfelelő lapok számának és az összes lehetséges lap számának ismeretében kiszámolja a megfelelő lap esélyét.

Egy példa az esélyszámolásra:

Egy játékos van játékban. Az osztónál egy lap ismert: egy pikk király. A játékos kezében három lap van: egy pikk hármas, egy káró ász és egy kőr bubi. Ekkor tehát a játékos lapjainak összértéke tizennégy és huszonnégy lehet, de mivel a számítás során az ász értékénél csak az egyet vesszük figyelembe, ezért tizennégy.

A tömb, mely a lapok számát tartalmazza:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Darab | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 16 |

Az asztalon lévő lapok ismeretében ezen lapok számát összesítve, majd kivonva a tömbből a következő változást eredményezi:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Darab | 3 | 4 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 14 |

Mivel a játékos lapjainak összértéke tizennégy, így minden lap megfelelő, mely értéke legfeljebb hét:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Darab | 3 | 4 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 14 |

Ebből kiszámolva a megfelelő lapok száma huszonhat, az összes lehetséges lap száma negyvennyolc. Ezek hányadosa 0.542, azaz 54.2% (ötvénnégy egész két tized százalék), kerekítve ötvennégy százalék.

### A felületek megvalósítási elve

A szülőfelületeket, melyen a szükséges felületek, panelek találhatóak, a JavaFX csomag Pane objektumának egy általam módosított változatával valósítottam meg. A módosítás annyiból állt, hogy a Pane ősosztály metódusait kibővítettem két metódussal: az egyik a panelek pontos méretének megadását, a másik a panelek vizuális megjelenésének egyes attribútumait változtatja meg egy metóduson belül, mely egyébként több metódushívással oldható meg. Ez az osztály szolgál a menüt, a csatlakozási felületet és a játékteret megjelenítő osztály ősosztályaként.

Újabb felületek megjelenítése a szerver által küldött üzenetek alapján lehetséges. A felületek létrehozása után újabb üzenetek érkezése során nem generálódik le ismét a teljes felület, hanem csak egyes panelek frissülnek.

### Menü felülete

A menü felülete a három különböző felület közül a legegyszerűbb: három elem található rajta. A felső részen található egy Button objektum, mely a játékindítás első lépését indítja el: a csatlakozási folyamatot. A gombon *„NEW GAME”* felirat szerepel. A felület középső részén egy Label objektum található, mely egy rövid, angol nyelvű leírást jelenít meg a játékról. A felület alján egy *„EXIT”* feliratú Button objektum helyezkedik el, melynek megnyomásával el lehet hagyni a játékot.

### Csatlakozási felület

A csatlakozási felület már egy kicsit bonyolultabb, összetettebb, mint a menü felülete. Három különböző állapotot lehet megkülönböztetni. Az első és harmadik állapot csak abban az esetben jelenik meg, ha több játékos is kapcsolódik az adott asztalhoz.

Mivel több játékos esetén a szerver egyszerre csak egy játékostól várja a becenevét válaszban, így az első az első állapot, amikor a kliens csatlakozott a szerverhez, de még nem kapta meg a szervertől az üzenetet, hogy el kell küldenie a játékos becenevét. Ekkor a felületen egy üzenet jelenik meg a játékos számára, miszerint várakozni kell a szerverre. Az üzenetet egy Label objektum jeleníti meg.

A második állapot, amikor a szerver a játékos becenevét várja. Ennek létrejöttekor a kliens létrehoz egy Timer típusú objektumot, mely egy TimerTask objektum ütemezését végzi, valamint egy már említett TimerTask objektumot, mely megadott időintervallumonként végrehajtja a magjában található utasításokat. Az említett magban található egy utasítás, mely frissíti az időjelző sáv állapotát, valamint egy másik utasítás, mely akkor fut le, amikor a visszaszámláló elérte a nullát, feladata pedig, hogy jelezze a szerver számára egy *„#player”* üzenettel, hogy a játékos nem adott meg becenevet, tehát azt a szervernek kell hozzárendelni. A visszaszámláló indulásával egyidőben megjelenik a sáv, mely a már említett hátralévő időt jelzi. Ezt egy Pane típusú panel valósítja meg, mely a szélessége az idő csökkenésével szintén csökken. A sáv alatt helyezkedi el egy TextField típusú elem, mely a becenév megadására szolgáló beviteli mezőt valósítja meg. Az elemhez hozzá van rendelve egy komponensfigyelő, mely új karakter leütése során ellenőrzi, hogy az új karakter illeszkedik-e egy, a komponensfigyelő magjában elhelyezett reguláris kifejezésre. Ammenyiben egyezést talál, hozzáadja a mező tartalmához, ellenkező esetben figyelmen kívül hagyja. A figyelő másik feladata, hogy figyelje a mező tartalmának hosszát. Amint ez az érték eléri a tízet, a továbbiakban már nem ad hozzá újabb karaktert. A beviteli mező alatt található egy Button típusú gomb, mely a név elküldése feladatot látja el. A feladat akkor hajtható végre, ha a mezőben található nevet még nem viseli egy játékos sem a már csatlakozott játékosok közül, valamint, ha a mező tartalmának a hossza legalább három. Valamely feltétel be nem teljesülése esetén a gomb megnyomása figyelmen kívül van hagyva. A gomb alatt elhelyezkedik egy Label típusú szöveg megjelenítésére alkalmas elem, mely a már csatlakozott játékosok neveinek listáját tartalmazza.

A harmadik állapot, amikor a játékos már csatlakozott. Ekkor egy Label objektum megjeleníti a játékos számára a már csatlakozott játékosok nevét.

### Játéktér

A játéktér felülete a legösszetettebb a három közül. A felület négy fő részre osztható, melyeket egyenként külön panelra hoz létre a kliens.

A bal felső sarokban elhelyezkedő Pane típusú panel a chat felületét jeleníti meg, ami három elemet tartalmaz: egy ScrollPane típusú panelt, mely lehetővé teszi a benne elhelyezett elemek görgetését, amennyiben azok mérete nagyobb, mint a szülőpanel mérete; továbbá tartalmaz egy TextField típusú beviteli mezőt, mely a játékos chatüzenetének a bevitelére használható; a harmadik elem egy Button típusú gomb, mely egy borítékot ábrázoló ikont jelenít meg és a feladata a megadott chatüzenet elküldése a kliens számára. Az üzenet elküldése csak akkor lehetséges, ha a beviteli mező nem üres, ellenkező esetben a gomb megnyomásának nincs egyéb hatása. A ScollPane panel tartalmaz egy Label típusú elemet, mely megjeleníti a chat üzeneteit. Az üzenetek egy ArrayList típusú listában vannak eltárolva és minden újabb üzenet hozzáadódik a lista végéhez.

A bal alsó sarokban található egy Pane típusú panel, az akciópanel. Ennek tartalma változó: lehet a tétrakás felülete, lehet a lapkérés felülete, valamint lehet üres is. A tétrakás felülete öt elemből ál. A létrejöttekör a kliens ismét létrehoz egy TimerTask típusú objektumot, melymegadott idő múlva jelzi a szerver számára, hogy a játékos nem küld tétet, azaz kihagyja a kört, valamint egy Timer típusú objektumot, mely ennek az ütemezését végzi. Megtalálható rajta a korábbról ismert hátralévő időt jelző sáv, mely egy Pane panel és a TimerTask objektum frissíti az állapotát. Továbbá egy Slider típusú csúszka, mellyel a tét mennyiségét lehet szabályozni, a minimum értéke nulla, a maximum pedig a játékos teljes zsetonmennyisége, továbbá egy egység egy zsetonnak felel meg. A csúszkával szinkronban működik az alatta található TextField típusú, tét bevitelére alkalmas mező. A szinkron abban rejlik, hogy a csúszka pozíciójának változása során a mezőben megjelenik a csúszka pozíciójának számértéke, valamint a mező tartalmának a játékos általi változása során a csúszka alkalmazkodik az új értékhez. A mező esetén a bevitel korlátozás alatt van, azaz betűt nem lehet bevinni, valamint, ha a tehető maximum tétnél nagyobb szám kerül a mezőbe, automatikusan a tehető maximum érték jelenik meg benne. Előfordulhat, hogy a mező üres, ekkor a csúszka értéke nulla és a tét elküldése gomb figyelmen kívül hagyja a megnyomást. A mező alatt elhelyezkedik két Button típusú gomb, az egyik a már ismert tét elküldése, a másik egy *„Skip”* feliratú gomb, mely megnyomásával a játékos kihagyja az aktuális soron következő kört.

A lapkérés felülete az előzőnél egyszerűbb. Szintén hozzá van rendelve egy Timer, valamint egy TimerTask objektum, utóbbinak ismét az a feladata, hogy jelezze a szerver számára, hogy a játékos nem reagált, de ebben az esetben már a lapkérés során és automatikusan a *„#stop”* üzenetet küldi el, azaz a játékos nem kaphat több lapot. Legfelül megtalálható a hátralévő időt jelző sáv, alatta pedig két Button típusú gomb: az egyik a lap kérése, mely a *„#card”* üzenetet küldi a szerver számára, a másik a megállást jelző gomb, mely a *„#stop”* üzenetet küldi. A lap kérése gombom, amennyiben a játékos igényli a segítséget, megjelenik annak az esélye, hogy a következő húzott lap megfelelő lesz a játékos számára. Ezt az igényt egy jelölőnégyzet bejelölésével kérheti a játékos, melynek kilétéről később lehet olvasni. A segítségkérésnek még egy funkciója van: ha a játékos esélye a jó lapra legfeljebb harminc százalék, a kliens egy megerősítő ablakot jelenít meg a játékos számára.

A játéktér felületének jobb oldalán a játékasztal helyezkedik el, melyet a felhasználó egy felületnek gondolhat, de valójában két külön felületről van szó: az osztó asztalrészének a felülete, valamint a játékosok asztalfelülete.

Az osztó felületének a bal felső sarkában helyezkedik el a korábbról már ismert, segítségkérés igényének jelzésére szolgáló CheckBox típusú jelölőnégyzet. Továbbá a felületen megtalálható még három darab HBox típusú panel, melynek előnye, hogy a rajta található elemeket horizontálisan, azaz vízszintesen egymás mellé helyezi. Az első panel tartalmazza az osztó zsetonegyenlegét, mely egy Label típusú elem, valamint egy hozzá tartozó ikont, mely egy Image típusú kép objektum megjelenítésére szolgáló ImageView elem. A második panel az osztó kártyalapjait jeleníti meg egy sorban. A lapokat külön-külön egy panel jeleníti meg, mely szerkezetéről a következő fejezetben lehet olvasni. A harmadik panel az osztó lapjainak összértékét jeleníti meg egy Label segítségével, valamint ehhez is tartozik egy ikon.

A játékosok területeinek megjelenítésére szolgáló felület egy HBox típusú panel, mely a játékosok számával megegyező számú Pane típusú panelt tartalmaz. Ezek a panelek tartalmazzák a játékosok összes szükséges adatát. Amennyiben az adott játékos a soron következő játékos, a panelje felett megjelenik egy nyilat ábrázoló Image objektum, melyet egy ImageView típusú objektum jelenít meg. A játékos panelének a felületén legfelül található egy HBox típusú panel, mely tartalmaz egy ikont és a játékos aktuális körben feltett tétjének a mennyiségét egy Label elem felületén. Feltett tét hiányában nulla jelenik meg. Alatta található egy Label típusú elem, mely a játékos nevét jeleníti meg. A név alatt ismét egy HBox panel található, mely a játékos zsetonmennyiségét jeleníti meg egy Label elemen, valamint mellette található az ehhez tartozó ikon. Ezalatt található egy HBox panel, mely a játékos lapjainak az összértékét, valamint ennek az ikonját jeleníti meg. Lapok hiányában nulla jelenik meg. Végül a panel alján egy ScrollPane típusú panel található, mely a játékos lapjainak a megjelenítéséért felel.

### Kártyalap megjelenítése



9. ábra. Kártyalap kinézete.

Egy kártyalap (9. ábra) megjelenítéséhez szükség van egy panelre. Erre a célra egy BorderPane típusú panelt választottam, mely előnye, hogy a panel öt szektorra van felosztva: négy oldalsó rész és a közepe. A panel szélessége ötven képpont, magassága hetven képpont, sarkai három képpont sugárú körcikk szerint vannak kerekítve. A széleit egy képpont vastagságú fekete vonal határolja. A panel három elemet tartalmaz. Középen található egy Image objektumú kép, melyet egy ImageView objektum jelenít meg. Ez felelős a lap színének megjelenítéséért. A panel középső szektorában kapott helyet. Ezen kívül a lap magassága megtalálható a felső szektorban balra igazítva, valamint az alsó szektorban jobbra igazítva. Ezeket egy Label elem jeleníti meg a magasság karakterjelzésével.

A játékos elől rejtett kártyalap egy egyszínű háttérű panel, melyet egy három képpont vastagságú vonal határol.

### Felugró ablakok

A játék során a kliens különböző üzeneteket jeleníthet meg a játékos számára felugró ablakok használatával. A szerver a megjelenítendő üzeneteket „\_svms\_” előtaggal látja el és ezek a következő esetek bekövetkezése során érkezhetnek:

* kör végén: az üzenetben közli, hogy az adott játékos vagy az osztó nyerte a kört kettejük közül; közli a részleteket is, azaz, ha valaki túllépte a huszonegyet, kiírja, hogy mennyivel, viszont, ha egyikük sem, akkor mindkettőjük pontszámát is; ha a játékos két lap segítségével elérte a huszonegyet, akkor a játékos a *„BLACKJACK”* üzenetet kapja meg.
* a játékos zsetonegyenlegének elfogyása esetén: tudatja a játékossal, hogy elfogytak a zsetonjai és kiesett a játékból.
* a játék végén: tudatja a játékossal, hogy véget ért a játék, valamint közli, hogy a játékot az osztó vagy a játékosok nyerték.

Ezek az üzenetek alapvetően tíz másodpercig láthatók, utána automatikusan eltűnnek, azonban a játékos a felugró ablakon található gomb segítségével azonnal eltüntetheti azt.

A szerver és a játékos között megszakadt kapcsolat esetén, valamint abban az esetben, ha a csatlakozás során a kapcsolatot nem sikerült létrehozni, a kliens egy felugró ablakban közli a történteket a játékossal.

Az információt közlő felugró ablakokon kívül megjelennek még megerősítő ablakok is. Ez két esetben lehetséges. Az első eset, amikor a játékos be akarja zárni a játék ablakát. Ekkor a felugró ablakban megjelenik a kérdés, mely a játékos döntését kérdőjelezi meg, meggyőződve annak biztosságában. A megerősítő gomb megnyomásával bezárul a játékfelület és megjelenik a menü. A másik eset, amikor a lapkérés során a aktív a segítség funkció, továbbá a játékos esélye megfelelő lapra legfeljebb harminc százalék és a játékos a lap kérése mellett dönt. A megerősítő ablakok esetén a játékos bizonytalanságát jelző, úgynevezett *mégse* gomb megnyomására eltűnik a felugró ablak.

## Előforduló hibák és azok kezelése

A tervezés és fejlesztés során számos futásidejű problémával kellett számolnom, valamint ezeket megfelelően kezelnem kellett. Az egyik legfontosabb teendő ezzel kapcsolatban a felhasználó számára megjelenített beviteli mezők korlátozása.

Az kliensalkalmazás két esetben vár a játékos billentyűzetéről szöveges bevitelt: a játékos becenevének megadása során, valamint a többi játékos számára küldött chat üzenet bevitele során. Előbbi csak az angol ábécé kis és nagy betűit, valamint arab számjegyeket tartalmazhat. Ennek oka, hogy az az aktuális állás közlése során a szerver az üzenetet kukac, kettőskereszt, kettőspont és pontosvessző karakterek alapján tagolja, így amellett döntöttem, hogy az írásjeleket nem fogja engedélyezni az alkalmazás. A chat üzenetben bármely, tetszőleges karakter előfordulhat, erre nincs semmilyen megszorítás.

Szám megadása egy esetben lehetséges a felhasználó billentyűzete által: a tét megadása során. A tét összegének megadása egy szöveges beviteli mező, valamint egy csúszka által lehetséges. A mező és a csúszka szinkronban van, azaz, ha a csúszka állása változik, azzal együtt változik a beviteli mező tartalma is, valamint ugyanígy fordított esetben. A beviteli mező csak számjegyeket engedélyez és a mezőben található szám értéke nem haladhatja meg a maximum tehető tét értékét.

A fejlesztés során ügyeltem arra, hogy adott időben csak a szükséges információk, valamint panelek jelenjenek meg. Ezek kezelésére a megoldást abban láttam, hogy egy panel használata során, amennyiben arra a továbbiakban már nincs szükség, az interakció pillanatában a kliens eltünteti.

Az egyik legváratlanabb hiba, ami felmerülhet, az a szerver és a kliens kapcsolatának megszakadása. Ebben az esetben a játékos sajnos kiesik a játékból, nem folytathatja azt. A kliens egy felugró ablak segítségével tudatja a játékossal ezt a tényt. Ekkor a szerver megszünteti a játékos számára létrehozott objektumot, mely a játékos adatait és kapcsolatait tartalmazza.

## Tesztelés

A szervert, valamint a klienst fejlesztés során folyamatosan teszteltem és amíg egy megírt kódrészlet nem felelt meg a teszteknek, nem kezdtem el egy újabb írását. A tesztek ellenőrzéséhez folyamatosan minden adatot, változást, állapotot megjelenítettem a parancssorban és sajátkezűleg ellenőriztem azok helyességét.

A szerver esetén több dologra kellett figyelni. A még kezdetleges szerver esetén a legfontosabb tesztesetek a következők voltak:

* hiányzó paraméter helyes kezelése.
* az adatok helyes tárolása.
* a játékosok állapotának elmentésére szolgáló objektumok helyes létrehozása, valamint állapotváltozások esetén azok helyes frissítése.
* a kliens számára küldött üzenet helyes összeállítása, valamint tagolása, főleg állapotközlés esetén.

A későbbiekben, amikor a szerver már képes volt körökre bontott játék levezénylésére, újabb tesztesetek merültek fel:

* az állapotok helyes kezdeti inicializálása.
* egy kör pontos levezénylése.
* az osztó döntéseinek feltételei és a lépéseinek észszerűsége.
* az aktuális kör végén új kör indítása, amennyiben a feltételek azt engedélyezik.
* a klienssel megszakadt kapcsolat helyes kezelése mind játék közben, mind pedig még a játék megkezdése előtt.

A kliens esetén a tesztelés már bonyolultabb volt, ugyanis ott a kinézetre és a helyes működésre egyaránt figyelni kellett. Elsőként a felületeket és azok kinézetét készítettem el, a következő dolgokra kellett figyelnem:

* minden adat, felirat jól láthatóan és észszerű pozícióban jelenjen meg.
* a különböző felületek tartalma észszerű legyen.
* különböző elemek ne fedjék egymást, kivéve, ha a megoldás ezt igényli, például kártyalapok egymás felé helyezése a játékosok lapjai esetén.
* tartalmak, funkciók megfelelően elkülönüljenek egymástól.

A kliens működését ellenőrző tesztesetek:

* hiányzó konfigurációs fájl kezelése.
* a kapcsolat megszakadásának megfelelő kezelése.
* a szervertől kapott üzenetek helyes értelmezése, azaz a hat karakterből álló előtagot minden esetben helyesen válassza le.
* az előtagtól függő események helyes végrehajtása.
* az állapotot közlő üzenetek megfelelő tagolása és azok helyes megjelenítése.
* a játékos általi választ nem igénylő üzenetek helyes automatikus megválaszolása.
* pontos esélyszámolás.
* a felületek, panelek csak szükségszerű megjelenítése, például amikor az adott játékos nem a soron következő, ne jelenjen meg számára sem a tétrakás, sem a lapkérés felülete.
* párhuzamosan, azaz különböző szálon történő műveletek megfelelő működése; ez különösen fontos azon funkcióknál, melyek használata során egy visszaszámláló is folyamatban van, például tét küldése vagy az elmulasztása adott időn belül.
* a kliens által tárolt adatok megfelelő frissítése, például új chat üzenet érkezése során.
* a játékos válaszát váró beviteli mezők korlátozásai, például a tétrakás során csak számot lehet megadni.

A tesztelés során felmerülő valamennyi hibát elemeztem, megterveztem, valamint végrehajtottam az elhárítását.

## Továbbfejlesztési lehetőségek

A fejlesztés során számos továbbfejlesztési lehetőség jutott eszembe, melyek a projekt tervezése során nem vetődtek fel, ezért úgy gondoltam, hogy ennek is szentelek egy fejezetet.

A szerver és egy kliens megszakadása során a játékos számára véget ért a játék, nincs lehetőség annak folytatására, tehát a visszacsatlakozásra. Ennek orvosolására egy fejlesztési lehetőség lenne esélyt adni a visszacsatlakozásra egy időkereten belül. A játékos visszakapná a kapcsolat megszűnése előtt birtokolt zsetonjait és azokkal folytathatná a játékot.

Az általam elkészített játék nem tartalmaz minden funkciót, lehetőséget, melyek részei a legnépszerűbb változatoknak. Továbbfejlesztéssel integrálni lehet a duplázás, kettéosztás, biztosítás, valamint feladás lépéseit a játékban. Ezekkel létre lehetne hozni többféle játékmódot és a kliens dönthetne, hogy melyikkel szeretne játszani.

A játék jelen állapotban csak az egy asztalnál tartózkodó játékosok számára teszi lehetővé a kommunikációt egymással. Egy lehetőség lenne az összes, adott pillanatban játékban lévő játékos számára egy közös kommunikációs csatorna létrehozása, így különböző asztalnál tartózkodó játékosok is tudnának kommunikálni egymással.

Egy adatbázis létrehozása, valamint a játékkal való összekapcsolása lehetővé tenné a játékosok számára a regisztrációt, így a korábban nyert zsetonösszegeket el lehetne menteni és később a játékosok ezek alapján tudnának játszani: gyarapítani azt és versenyezni a többi játékossal, a zsetonegyenlegek összegét illetően. A zsetonegyenleg kimerülése esetén a játékos kaphatna bizonyos időközönként zsetonokat a szervertől, így megelőzve azt, hogy a játékos újra regisztráljon a játszás érdekében.

A kliensoldalon létre lehetne hozni különböző animációkat, melyek a játék vizuális élményén javítanának. Ilyen animáció lehetne lapok osztása során a kártyalap tényleges mozgása, tétrakás során a tét megjelenítése és mozgása.

Hangok hozzáadása a játékhoz szintén a felhasználói élményt javítását erősítené. Különböző hangok a figyelemfelkeltést segíthetik elő, valamint egy nyugtató háttérzene a kellemes időtöltés egyik lehetséges eleme.

Jelenleg nem módosítható időkorlátok módosítási lehetősége a szerver által. Ugyanis jelenleg a szerver üzenetek csak tíz másodpercig jelennek meg, valamint a játékosnak a tétrakás és a lapkérés esetén fél perc gondolkodási ideje van, gyakorlatban ezek nem mindig elegendő időkorlátok.

# Összegző

Szakdolgozatom elkészítése során egy teljesértékű játék létrehozása volt a cél. Ennek a játéknak a Huszonegy nevű kártyajátékot választottam, ugyanis ez egy világszerte ismert játék és én is szívesen töltöm a szabadidőmet ennek a játéknak a gyakorlásával.

A játék elkészítéséhez a Java programozási nyelvet, valamint ezen alapuló technológiákat választottam. A döntésem oka, hogy a Java nyelv tökéletesen illet a játék problémájának a megoldásához, ugyanis objektumorientált nyelvről van szó, melyet előszeretettel használnak a fejlesztők, valamint a könyvtárai lehetővé teszik és elősegítik a hálózati programozást.

A kliens megvalósításához a JavaFX könyvtárat választottam, ugyanis ez a könyvtár tartalmaz számos grafikai elemet, melyek használata egyszerű és széleskörűen használható

A játékprogram megvalósítása előtt felállítottam bizonyos kritériumokat, megszorításokat, melyeket be kellett tartanom. Ezek a feltételek a játék helyes működését segítik elő és megkönnyítik a kliens használatát.

Játékalkalmazásom fejlesztése mellett párhuzamosan elvégeztem a tesztelés lépéseit is így csökkentve a fejlesztés későbbi lépéseiben felmerülő hibák előfordulási esélyét.

Az elkészítéssel töltött idő véleményem szerint pozitív tapasztalatokkal szolgált és rengeteg új dolgot tanulhattam belőle, valamint gyakorolhattam, fejleszthettem a Java nyelv használatában és a programozásban szerzett eddigi tudásomat. Meg vagyok elégedve a munkám végeredményével és sikeresnek tekintem az elém tárult akadályok leküzdését.

# Irodalomjegyzék