borító

1. Bevezetés //nincs kész

2. Szakdolgozat leírása

2.1. A játék leírása

2.1.1. Alapjáték leírása

2.1.2. A játék menete

2.1.3. Módosítások az alapjátékhoz képest

2.2. Programozási nyelv, fejlesztői környezet

3. Felhasználói dokumentáció

3.1. Előfeltételek

3.2. Használat

3.2.1. Felületek

3.2.2. Beviteli elemek és megszorításaik

3.2.3. Fellépő hibák

4. Fejlesztői dokumentáció //nincs kész

4.1. Tervezés

4.2. Szerver

4.2.1. Konfigurációs fájl

4.2.2. Fő szerver

4.2.3. Szál, mint asztal

4.2.3.1. Előkészület

4.2.3.2. Játék lebonyolítása

4.2.3.3. A teljes állapotküldés formája

4.2.3.4. Hibák és azok kezelése

4.3. Kliens

4.3.1. Konfigurációs fájl

4.3.2. Háttér

4.3.2.1. Üzenet feldolgozás

4.3.2.2. Esélyszámolás

4.3.3. Felületek

4.3.3.1. Menü

4.3.3.2. Csatlakozási felület

4.3.3.3. Játéktér

4.4. Tesztelés

4.5. Továbbfejlesztési lehetőségek

5. Összegző //nincs kész

6. Irodalomjegyzék

1. 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 [1] 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 és az osztó viselkedés nem függ a játékosok számától.

Amint az osztó képes volt játék lebonyolításába, 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.

2. Szakdolgozat leírása

A következő fejezetben be szeretném mutatni a Huszonegy játék hagyományos, valamint az általam megvalósított változatának a menetét és a szabályait. Továbbá meg szeretném ismertetni a felhasznált technológiákat és a fejlesztői környezetet.

2.1. A játék leírása

2.1.1. 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.

2.1.2. 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.

2.1.3. 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 választhat.

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.

2.2. 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 [2] keresztül érhető el. A kommunikációt socket-ek [3] segítségével 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.

3. Felhasználói dokumentáció

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

3.1. Előfeltételek

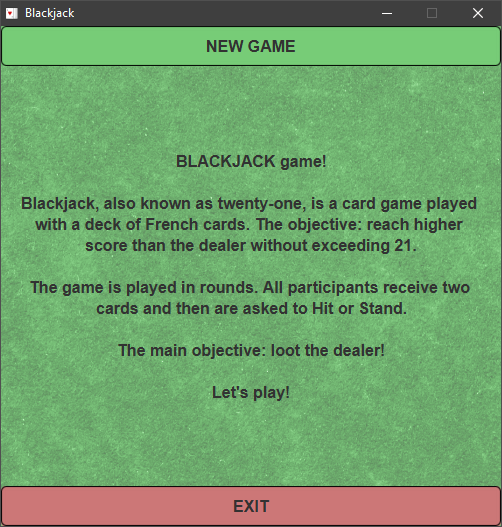
Az alkalmazás futtatásához szükség van Java SE Runtime Environment legalább 8-as verziójára, ugyanis ettől a verziótól kezdve része a JavaFX a Java SE API-nak.

3.2. Használat

A játékalkalmazás elindításához szükséges egy client\_config.properties nevű fájlra, amely a szerver eléréséhez szükséges adatokat tartalmazza. Ennek meglétében a szokásos módon tudjuk elindítani az alkalmazást.

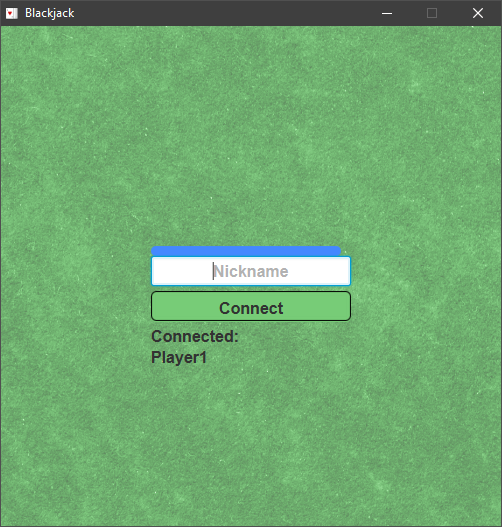
3.2.1. Felületek

A játék három különböző felületet tartalmaz: menü, kapcsolódási felület, játékfelület.



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

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 játékosra esik lépés kiválasztása, 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. |

3.2.2. 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.

3.2.3. 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.

4. Fejlesztői dokumentáció

4.1. Tervezés

4.2. 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 a parancssorban az alkalmazás főkönyvtárából futtatva a következő utasítással lehet Java bájtkódot[4] 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.

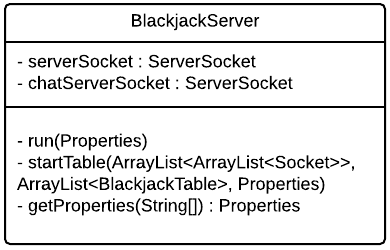
Az alkalmazás futtatásához a következő utasítást kell bevinni a parancssorba:  
java -jar Blackjack\_server.jar <konfigurációs fájl elérési útvonala - opcionális>  
Amennyiben a konfigurációs fájl elérési útvonala nincs megadva, úgy a szerver alapértelmezetten a default\_server.properties nevű fájlt 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.

4.2.1. Konfigurációs fájl

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

* name: a konfiguráció neve.
* 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.
* 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 kezdenek.
* 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.
* 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.

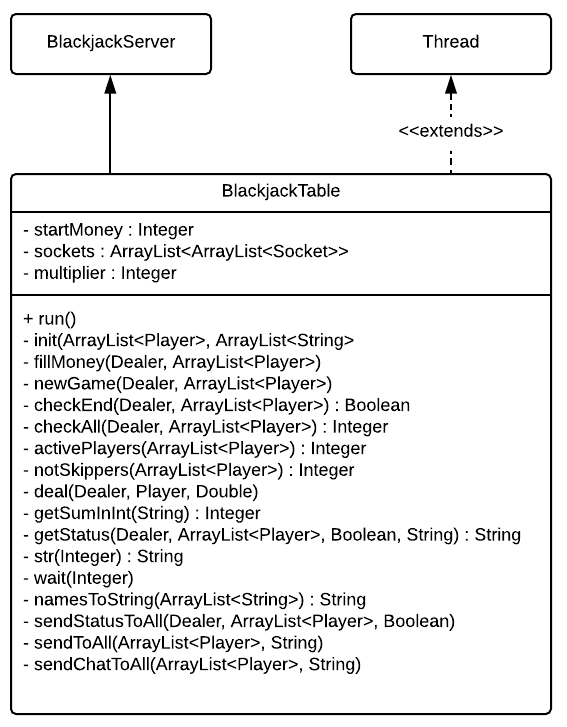
4.2.2. Fő szerver



4. 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.

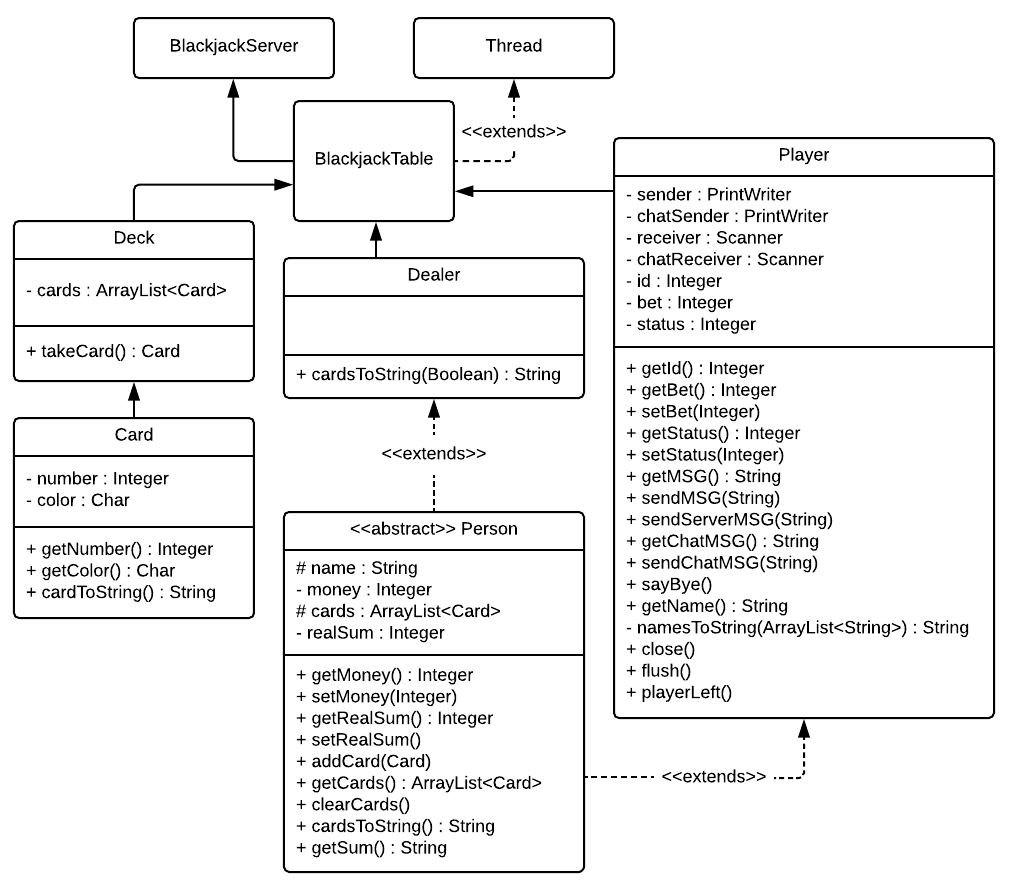
4.2.3. Szál, mint asztal



5. ábra. BlackjackTable osztály UML diagramja

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

4.2.3.1. Előkészület



6. 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 zsetonösszeg.

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 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 lehet 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, 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.

4.2.3.2. 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 pakli, 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, és 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.

4.2.3.3. 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 „@” karakter választja el.

Az osztó következő adatait tartalmazza az üzenet ebben a sorrendben „;” 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. 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 „#” karakter választja el. Egy játékos adatai a következő sorrendben jelennek meg „;” 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 kihagyta 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.

4.2.3.4. Hibák és azok kezelése

4.3. Kliens

4.3.1. Konfigurációs fájl

4.3.2. Háttér

4.3.2.1. Üzenet feldolgozás

4.3.2.2. Esélyszámolás

4.3.3. Felületek

4.3.3.1. Menü

4.3.3.2. Csatlakozási felület

4.3.3.3. Játéktér

4.4. Tesztelés

4.5. Továbbfejlesztési lehetőségek

5. Összegző

6. Irodalomjegyzék

[1] Wikipedia, https://hu.wikipedia.org/wiki/Kliens-szerver\_architektúra

[2] Wikipedia, https://hu.wikipedia.org/wiki/HTTP

[3] Wikipedia, https://hu.wikipedia.org/wiki/Socket

[4] Wikipedia, https://hu.wikipedia.org/wiki/Java\_bájtkód