Programozás alapjai 3. Dokumentáció

Harsányi Zsolt - XORZVV

2022

Tartalom

1 A játékról

1.1 Játék leírása

A 66 egy igen népszerű magyar kártyával játszott játék, általában 2 fő játszik egymás ellen. Röviden a lényege a nevéből adódóan 66 pont összegyűjtése hamarabb és annak bejelentése. A játékosok 5- 5 kártyával rendelkeznek, minden kártyának van egy állandó értéke. Minden körben 1-1 kártyát raknak le a játékosok, és akinek nagyobb értékű a kártyája annak pontszámához hozzáadódik a két letett kártya értéke. Természetesen a játékról 2-3 oldalt is lehetne írni, ez a leírás csak nagy vonalakban közelíti meg a játék komplexitását. Informálisabb leírás:http://www.gyerekjatekokrol.hu/jatekok/snapszer/

1.2 A játék felépítése

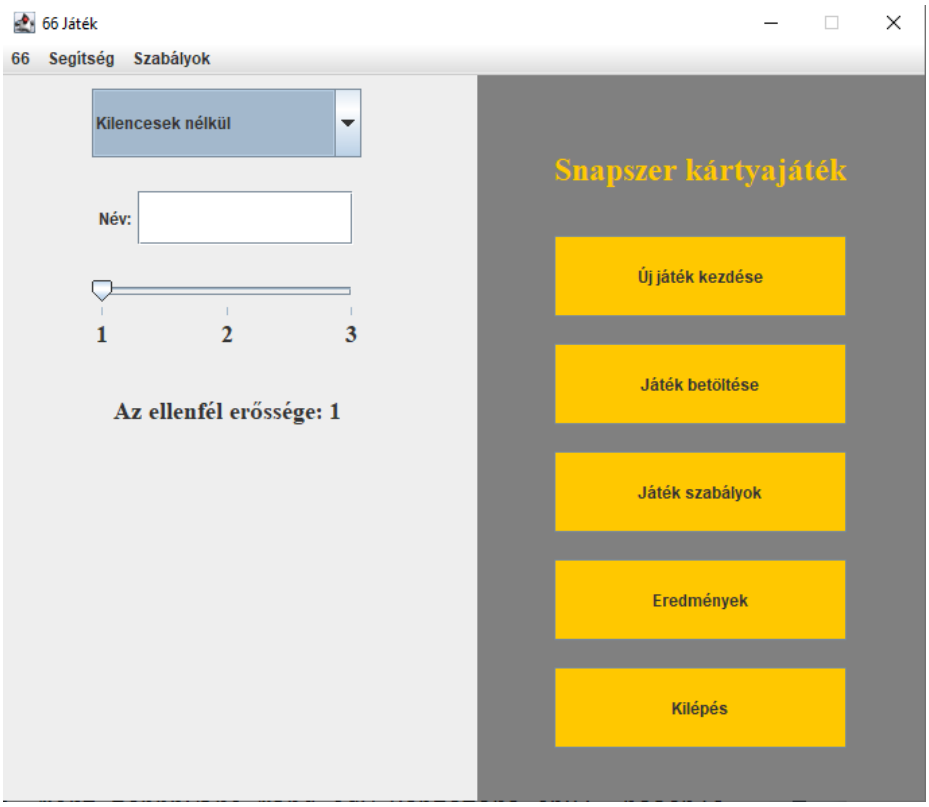
A játékban a meghatározott kártyajátékot lehet egy bot ellen játszani. A bot egy kezdetleges, nem túl okos, egyszerű logika alapján tesz kártyákat és válaszol kártyákkal a felhasználó által letett kártyákra. A játék SWING GUI támogatásával, a Collection keretrendszer segítségével valósul meg. A játékban láthatjuk a nálunk lévő kártyákat, amik .jpg kiterjesztéssel érhetőek el projekt könyvtárában, és ezekre kattintva léphetünk velük interakcióba. Ezen kívül, ha elfogynak a kártyák, vagy a felhasználó vagy bot eléri a 66 pontot, akkor véget ér a játék és kikerül a győztes. A játék egy egyszerű menüvel nyílik meg, játék folytatása vagy új játék kezdése edetén pedig a játék új ablakban zajlik.

A leírás és felépítés megtalálható volt a specifikációban is, azonban a teljesség kedvéért itt is megjelenítettem.

2 Felhasználói útmutató

2.1 Menü

A játékot elindítva a játék menüjébe csöppenünk, ahol különböző opciók közül választhatunk.



2.1.1 Új játék kezdése

Miután a menü ablak bal részében beállítottuk a szükséges paramétereket az új játék kezdése gombra kattintva sikeresen meg kell nyílnia egy új ablakban ahol maga a játék folyamata zajlik.

2.1.2 Játék betöltése

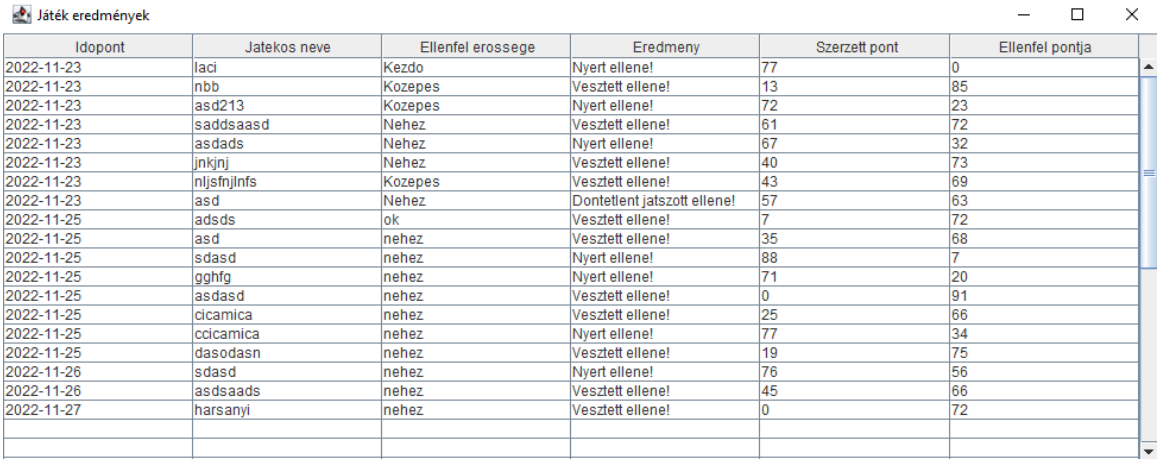
A bal panel adatainak kitöltése nélkül tudunk folytatni meglévő játékot. Ekkor ugyanúgy megnyílik a játék ablaka, azonban ugyanott fogjuk folytatni, ahol mentettük előtte.

2.1.3 Játék szabályok

Rá kattintva egy új ablak nyílik meg ami a játék szabályait tartalmazza.

2.1.4 Eredmények

Ablak, ami a játék eredményeit jeleníti meg.

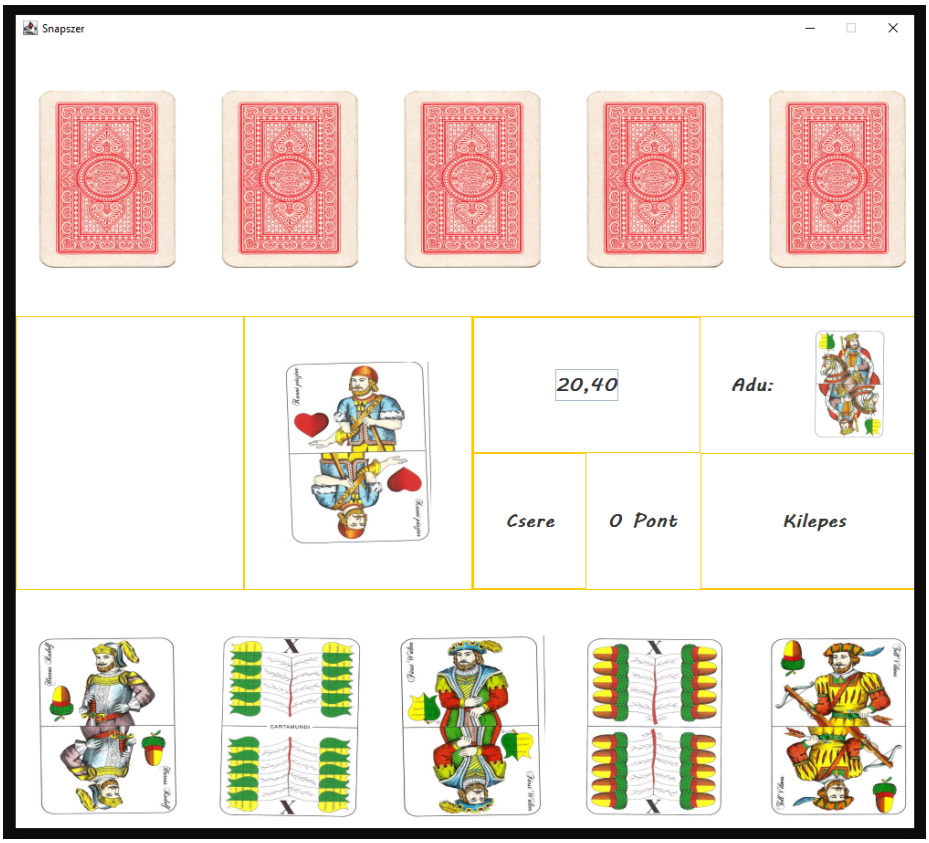


Idopont	Jatekos neve	Ellenfel erossege	Eredmeny	Szerzett pont	Ellenfel pontja
2022-11-23	laci	Kezdo	Nyert ellene!	77	0
2022-11-23	nbb	Kozepes	Vesztett ellene!	13	85
2022-11-23	asd213	Kozepes	Nyert ellene!	72	23
2022-11-23	saddsaasd	Nehez	Vesztett ellene!	61	72
2022-11-23	asdads	Nehez	Nyert ellene!	67	32
2022-11-23	jnkjnj	Nehez	Vesztett ellene!	40	73
2022-11-23	nljsfnjlnfs	Kozepes	Vesztett ellene!	43	69
2022-11-23	asd	Nehez	Dontetlent jatszott ellene!	57	63
2022-11-25	adsds	ok	Vesztett ellene!	7	72
2022-11-25	asd	nehez	Vesztett ellene!	35	68
2022-11-25	sdasd	nehez	Nyert ellene!	88	7
2022-11-25	gghfg	nehez	Nyert ellene!	71	20
2022-11-25	asdasd	nehez	Vesztett ellene!	0	91
2022-11-25	cicamica	nehez	Vesztett ellene!	25	66
2022-11-25	ccicamica	nehez	Nyert ellene!	77	34
2022-11-25	dasodasn	nehez	Vesztett ellene!	19	75
2022-11-26	sdasd	nehez	Nyert ellene!	76	56
2022-11-26	asdsaads	nehez	Vesztett ellene!	45	66
2022-11-27	harsanyi	nehez	Vesztett ellene!	0	72

2.1.5 Kilépés

Megnyomva kilépünk a menüből és a játékból.

2.2 Játék



Felosztás: Az ablak alján kártyáinkat, tetején az ellenfél kártyáit, középen a lerakott kártyákat és a játékhoz tartozó interakciókat találjuk.

A játék körökre van osztva. Minden körben kártyát kell raknunk az ellenfél kártyája ellen, vagy a körben először. Ahhoz hogy lerakjunk egy kártyát a kártyára kattintva azt be kell húznunk középen az első dobozba.

20,40 - Ha olyan kör van ami előtt mi nyertünk, tehát mi rakunk először akkor bemondhatjuk ha van 20, vagy 40-et érő király,felső páros a kezünkben. Ehhez a gombra kattintva azonnal láthatjuk, ugyanis megváltozik a pontunk ha tényleg nálunk van a két kártya. Ezek után csak ebből a kettő kártyából játszhatunk ki, tehát középre csak a király vagy felső kártyát húzhatjuk be.

Csere - Ha van alsó a kezünkben, aminek színe megegyezik az adu színével, akkor a játék

folyamán a gombra kattintva bármikor cserélhetjük a kettőt.

Adu: - A szöveg mellett láthatjuk a játék aduját. Pár kör után eltűnik az ikon, miután az is kiosztásra került.

Kilepes - Rá kattintva egy felugró panel kérdez meg minket, hogy akarjuk-e menteni az állásunkat. Ha igen, később betölthetjük ugyanazt.

3 Osztályok

Az alkalmazás osztályai két csoportra bonthatók, logikai és grafikaira. Mélyebben a logikai osztályokat fogom tárgyalni, ugyanis a grafikai osztályok véleményem szerint nem érdemelnek akkora figyelmet, mert többnyire mind egy kaptafára épül, hasonló metódusokkal és szinte ugyanolyan attribútumokkal. Tehát azokat csak említés szinten tárgyalom, azt leírva mit valósít meg. A játék logikai részével azonban mélyebben foglalkozok, ugyanis az korántsem olyan triviális, több kérdést felvet.

Megjegyzés: A legtöbb setter/getter függvénynek kihagytam a dokumentálását, ugyanis azok triviálisan ugyanarra a logikára épülnek.

3.1 CardList

3.1.1 Leírás

A kártyákat tárolja ez az osztály. Inicializáláskor eldönthető, hogy kilencesek nélkül, illetve kilencesekkel együtt játsszunk. Emellett több, a kártyákon végzett műveletet valósít meg.

3.1.2 Attribútumok

cardList:ArrayList-String	Az összes kártyát tárolja, melyeket majd eloszt a játékosok között.

3.1.3 Metódusok

+CardList(gameType: int) - Konstruktor, feltölti a listát a kártyákkal. A gameType adja, hogy kilencesek kerüljenek, vagy ne a listába. Ezután random sorrendet épít a listában a kártyák között (megkeveri őket).

+burn(index: int) - Annyit éget a kártyából, amekkora számot adunk neki. Azaz annyi elemet fog kitörölni az első helyről.

+getValue(card: String): int - Visszaadja az adott kártya értékét.

+isAdu(card: String, adu: String): boolean - Visszaadja, hogy az adott kártya adu kártya-e.

+compareCards(placedCards: String[], user: User, bot: Bot, prevWinner: int, adu: String): int - Összehasonlítja a játék szabályai szerint a két letett kártyát, és ez alapján ad pontot a felhasználónak vagy az ellenfelének.

3.2 GameLogic

3.2.1 Leírás

A Játék általános logikájáért felelős.

3.2.2 Attribútumok

-window: GameWindow	A játék grafikai megjelenítéséért felelős
-cardList: CardList	A kártyákért felelős
-user: User	A felhasználó szerepkör megvalósítása
-bot: Bot	Az ellenfél a játék során
-aduCard: String	Az adu kártya a játék folyamán
-playedCards: String[]	Az aktuálisan kijátszott két kártya(minden körben változik)
-rounds: int	A játék köreinek számát reprezentálja
-twentyForty: String[]	Ha a felhasználó 20-at, vagy 40-et mond be, ideiglenesen tároljuk a két kártyáját
-gameType: int	Megadja, hogy kilencesekkel, vagy kilencesek nélkül játszunk-e.
-prevWinner: int	Megadja, ki nyert az előző körben.

3.2.3 Metódusok

+GameLogic(newOrLoad: int, gameType: int, name: String, difficulty: int) - Konstruktor, amely beállítja a tagváltozók értékét, aszerint hogy új játékot kezdünk, vagy betöltünk egy elmentett játékot. Egyes attribútumokat a paraméterek alapján ad át, mellette alaphelyzetbe hozza a játékot.

+initWindow() - Inicializálja, és alaphelyzetbe állítja a játék ablakát.

+Game() - A játékmenetért felelős eljárás. Mindent attól függően csinál, hogy nyert-e valaki, vagy épp ki vitte a lapokat az előző körben. Ha valaki megszerezte a 66 pontot, vagy elfogyott mindkét fél kártyája akkor ki hirdeti a győztest.

+handleNewCards() - A kör után új kártyát oszt mindkét fél számára. Pár körig ez triviális, ugyanis csak a pakli tetejéről kell káryát húzni, azonban ha elfogy a kártya az adunak is megfelelő helyre kell kerülnie.

+actionPerformed(e: ActionEvent) - Az interakciók megvalósítása, a különböző gombok lenyomása során. Ilyenek például a játék befejezése, az alsó kicserélése, a 20 vagy 40 bejelentése.

-saveGame() - A játék elmentése szerializálás segítségével. Amit szerializálunk az az aktuális GameLogic osztály.

-loadGame(): GameLogic - A mentett GameLogic osztály visszatöltése, betöltése.

+checkForWinner(): int - Függvény, amely megállapítja, ki nyerte a játszmát. 0 - Ha a felhasználó sikeresen legyőzte ellenfelét, 1 - ha vesztett, -1 - ha döntetlen.

+declareWinner(winner: int) - Eljárás, amely ki hirdeti a győztest, utána pedig bezárja az alkalmazást.

-writeToResults(result: String) - IO művelet, ami kiírja a results.txt-be a játszma adatait, miután vége lett annak. Tartalma a szerzett pontok, az ellenfél nehézsége, stb...

3.3 Player

3.3.1 Leírás

Absztrakt ősosztály, egy játékost valósít meg és az ő által végzett műveleteket. Leszármazottai a Bot, amely az ellenfélt reprezentálja és a User, ami minket felhasználót.

3.3.2 Attribútumok

-checkedForPoints: int	Flag, ami azt jelzi hogy bemondta-e a játékos a 20 vagy 40-et. Ha igen akkor nem rakhat utána csak a király, vagy felső közül.
-points: int	A játékos pontjai. Ha eléri a 66-ot, megnyeri a játszmát.
-cardsInHand: ArrayList-String	Lista, a játékos kezében tartott kártyák összessége.Ha 5-nél kevesebb kártyája lesz, a listában 0 jelenik meg az üres helyeken.

3.3.3 Metódusok

+Player(list: CardList) - Konstruktor, mely beállítja a tagváltozók értékét és feltölti a játékoshoz tartozó listát a pakliból.

+replaceCard(cardList: CardList, card: String, notFromList: String) - Egy bizonyos kártyát helyettesít egy másikkal a játékos kezében

+aduAlsoSwitch(cardList: CardList, adu: String): String - Metódus, amely kicseréli az adu alsót az adu kártyával, ha a játékosnál van alsó. Visszaadja az új adu értékét, ha létrejött a csere.

+twentyOrForty(cardList: CardList, adu: String): String[] - 20-al, vagy 40-el növeli a játékos pontszámát, ha egyszerre van nála felső és király is. Ha mindkettő az adu színe, akkor 40-el. Visszaadja a felsőt és királyt további műveletek elvégzéséhez.

+getValues(list: CardList): int[] - Visszaad egy int tömböt, ami a kártyák értékét reprezentálja sorban.

+hasAdu(list: CardList, adu: String): boolean -Logikai értéket ad vissza arra, hogy a játékos kártyái között van-e adu kártya.

3.4 User

3.4.1 Leírás

A Player osztály leszármazottja, a felhasználót valósítja meg. Ami nincs feltűntetve az attribútumok és metódusok között, azt mind örökölte a Player osztályból.

3.4.2 Attribútumok

-name: String	A játékos egyedi neve, amelyet a játék elején állíthat be

3.5 Bot

3.5.1 Leírás

A Player osztály leszármazottja, az ellenfelet valósítja meg. Az osztály lényege reagálni a felhasználó által tett lapokra és rakot lapni, amire a felhasználó reagál és ennek komplexitását különböző nehézségi szinteken érzékeltetni. Ami nincs feltűntetve az attribútumok és metódusok között, azt mind örökölte a Player osztályból.

3.5.2 Attribútumok

-difficulty: int	Az ellenfél nehézsége, amelyet a konstruktor állít be.

3.5.3 Metódusok

+placeCard(prevWinner: int, list: CardList, adu: String, placedCard: String, user: User): String - Az osztály szívének metódusa, lényege egy kártya visszaadása, amit letett az ellenfél. Ez lehet kezdeményezés, reagálás. Nehézségi szinthez fogja kötni a metódus, milyen segédfüggvényeket használjon ahhoz, hogy minél okosabban tegyen le kártyát.

+minValueIndex(values: int[]): boolean - Visszaadja a paraméterül kapott tömb legkisebb elemének a helyét, vagyis gyakorlatban azt a kártyát az ellenfél kezében, ami a legkisebb értéket képviseli.

+placeAgainst(list: CardList, card: String): String - Ha kártyát raktunk, az ellenfél e függvény szerint fog válaszolni, ha legalább kettes nehézségű.

+placeBetter(list: CardList,user: User, adu: String): String - Hármas nehézségnél lép életbe. Ha az ellenfél kezdeményez, a legkisebb értékű lapja helyett igyekszik jobb lapot tenni.

+placeRandomCard(): String - A kéznél lévő kártyái közül választ random egyet, azt adja vissza

3.6 GameWindow

A játékot jeleníti meg grafikusan.

3.6.1 Attribútumok

+setDeck(deck: JLabel[], cardsInHand: ArrayList-String, card: String) - A játékpanelhez tartozó két dekket tölti fel a kártyák képeivel

3.7 CardDeck

A játék ablak tetején és alján elhelyezkedő kártyákat jeleníti meg.

3.7.1 Attribútumok

+removeListener(cards: String[], cardsInHand: ArrayList-String) - Bizonyos kártyákról eltávolítja a listenert, hogy azokat ne lehessen letenni az asztalra.

3.8 DragMouseAdapter

Adapter osztály, mely figyeli hogy egy kártyát az asztalra tettünk-e. Ha igen az asztalon lesz, vagyis megkapja az ikonát.

3.9 MidPanel

A játék ablakának középső része, ami a játék lényegi információit és interakcióit hordozza. Két kártya tartó, az ellenfél és a felhasználó kártyáinak, különböző gombok különféle műveletekhez, a pontszámunk és hasonlókat tartalmaz.

3.10 MenuWindow

Az alkalmazás indításakor megjelenő ablak. Különféle gombokat és beállítható elemeket tartalmaz.

3.11 GameRules

A menüből nyitható ablak, amely egy egyszerű beágyazott html leírást tartalmaz a játék szabályaival kapcsolatban.

3.12 OptionsWindow

A játékhoz beállítható információk összessége, amely a menüben fellelhető beágyazva. Itt állítható az ellenfél erőssége, a nevünk és hogy milyen típusú 66-ot szeretnénk játszani.

3.13 MenuBar

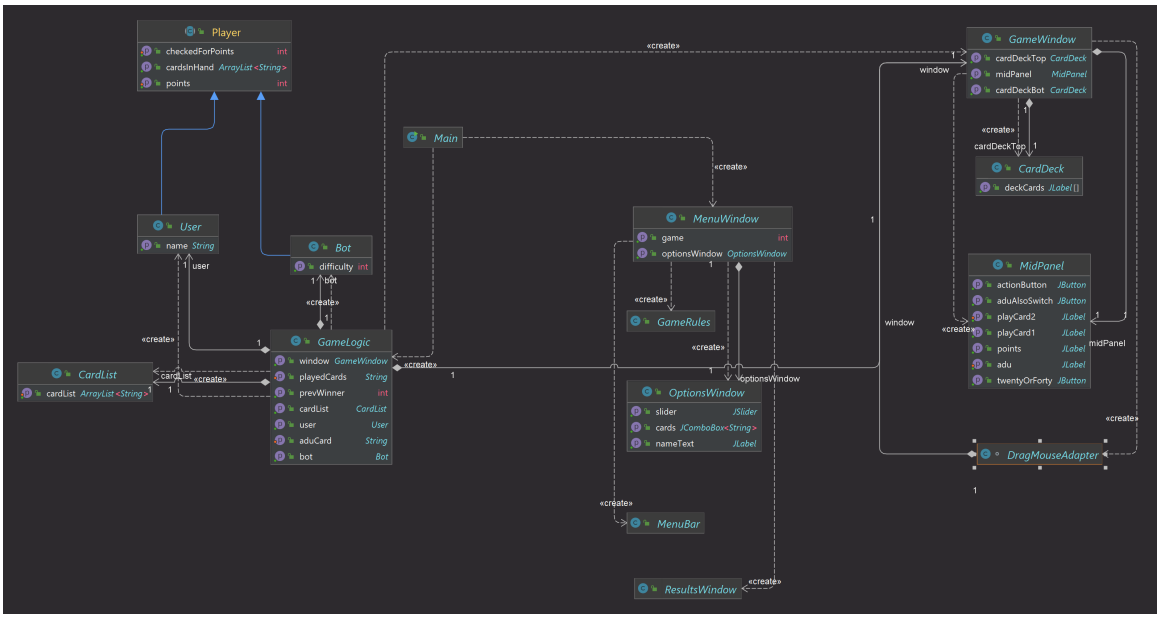
A menühöz tartozó menubar, vagyis az ablak tetején megjelenő opciók összessége, a menü ablakba ágyazva.

Megjegyzés: A megvalósítása nem feltétlen igényelt külön osztályt, azonban az átláthatóság kedvéért külön osztályt kapott, így végképp újrafelhasználható.

3.14 ResultsWindow

A menüből megnyitható ablakok egyike, lényege a játékban elért eredmények megjelenítése.

4 Osztálydiagram



Megjegyzés: A diagram nem tartalmazza az osztályok metódusait és mezőit, ugyanis nagyon olvashatatlanná tennék az összképet. Azonban a dokumentáció mellé csatolni fogom azt is a teljesség kedvéért.

5 Tesztek

AduAlsoSwitchTest - Teszteli, hogy jól működik-e az aduAlsoSwitch függvény.

BotPlaceAgainstTest - Teszteli, hogy az ellenfél jó kártyát rak-e a mi rakott kártyánkra(kellően komplex-e a válasz).

CardValueTest - Teszteli, hogy a getValue() helyes értéket ad-e vissza egy bizonyos kártyához.

CheckForWinnerTest - Teszteli, hogy adott pont felett helyes játékost ítél-e meg a függvény győztesnek.

CompareCardsTest - Egyszerű teszt annak eldöntésére, hogy a rakott kártyákra helyesen adja a függvény a pontokat a két játékos között.

DeckSizeTest - Ellenőrzi, hogy a pakli helyesen fogy-e az osztások során.

HasAduTest - Ellenőrzi a hasAdu() függvény helyességét.

IsAduTest - Teszt az isAdu() függvény helyességének ellenőrzésére.

PlayerEmptyHandsTest - Ellenőrzi, hogy tényleg nincs-e már kártyája a játékosnak.

PlayerMinValueIndexTest - Teszt a minValueIndex() függvény helyességének ellenőrzésére.

TwentyOrFortyTest - Ellenőrzi, hogy helyesen növeli-e a pontokat a twentyorForty() függvény.