Gymnázium, Praha 6, Arabská 14

Obor : Programování

Téma : Počítačová hra

Jméno projektu : Hra DaVinciho Code

Datum : květen 2022

Vypracoval : Jakub Vagera 1.E

Prohlášení:

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne 3.května 2022 .....

Contents

1 Anotace

Ve své ročníkové práci se zabývám vytvořením počítačové hry, která je známá ze světa deskových her jako Da Vinciho Code. Tuto logickou hru vytvořil v roce 2004 Eiji Wakasugi a dostala jméno Coda. Hra v sobě kombinuje hlavolam a deduktivní princip, a ačkoliv narazíte ve hře na čísla, nemusíte se bát, že byste museli využívat matematiku nebo s nimi nějak počítat. Jedná se o hru, kde se hráč snaží uhodnout číselnou řadu (kód) svého soupeře. Počítačová hra je vytvořena ve dvou variantách. V první jednodušší variantě se nepoužívá karta "pomlčka" a ve druhé složitější variantě se používá karta "pomlčka". Hru je možné hrát v režimu dvou hráčů proti sobě nebo hráč proti počítači.

2 Slovník

Pomlčka - karta zastávající libovolnou hodnotu (wild card / žolík)
Bubble sort - třídící algoritmus
For loop - označení pro řídící strukturu spadající mezi cykly

3 Úvod

3.1 Zadání

Zadáním práce bylo vytvořit v programovacím jazyku Java počítačovou hru, která odpovídá principu deskové hry Da Vinciho Code. Na rozdíl od deskové hry, kde mohou hrát až čtyři hráči, je počítačová hra vytvořena pro dva hráče nebo hráče a počítač.

3.2 Pravidla hry

Jedná se o logickou hru s čísly. Ve hře jsou černé a bílé karty s hodnotou 0 až 11 a "tajná" karta s pomlčkou, ta slouží k matení soupeře. Každý hráč si na začátku vezme čtyři libovolné (bílé i černé) karty a neukazuje je soupeři. Potom karty seřadí vzestupně, nejmenší číslo vlevo, největší v pravo. Tato posloupnost je "tajný kód", který se protihráč snaží uhodnout. Pokud má hráč stejné číslo na bílé i černé kartě, tak nejříve umístí vlevo bílou kartu, pak teprve černou. Hodnota bílé barvy je vždy nižší. Zároveň se ve hře nachází i dvě karty (černá a bílá) s pomlčkou, které zastupují libovolnou číselnou hodnotu. Hráč na začátku tahu vylosuje ze společné hromádky jednu novou kartu a pokusí se uhodnout některé z čísel ze soupeřova kódu. Pokud hráč na tahu uhodne číslo skrývající se za kartou soupeře, tak soupeř sklopí (odkryje) uhodnutou kartu a hráč může dle její hodnoty a polohy odhalovat ostatní čísla. Dále může hráč pokračovat v hádání nebo ukončit svůj tah. Pokud hráč na tahu neuhodne číslo na kartě soupeře, tak si vylosované číslo zařadí na patřičnou pozici do svých karet. Počet čísel (délka kódu) u hráče na tahu se tím zvýší. Po neuhodnutí hraje soupeř. Pokud hráč odhalí všechny své karty (celý kód) tak pro něj hra končí. Vyhrává ten hráč, kterému zůstanou neuhodnuté karty.

Figure 1: Desková hra DaVinciho Code

4 Software výpočetní části

Softwérová část mého programu se skádá ze čtyř samostatných programů. První je program pro hraní hráče s hráčem, druhý je program pro hraní hráče s počítačem, třetí je pro hraní hráče s hráčem bez pomčlky a čtvrtý je hraní hráče s počítačem bez pomlček. U Programů hry hráče s počítačem počítač zastává funkci druhého hráče. V této dokumentaci budu pro lepší orientaci mluvit pouze o programu hra hráče s hráčem a programu hraní hráče s počítačem, jelikož jak program pro hraní hráče s hráčem bez pomlčky tak i program hraní hráče s počítačem bez pomlčky vychází z dříve zmiňovaných programů.

4.1 Náhodné rozdání karet

Jako první část programu jsem vytvořil array pole s 26 čísly od 0 do 12 (dvanáctka představuje pomlčku), abych rozlišil zdali se jedná o karty černé nebo bílé, přidal jsem k bílým číslům konstantu 0,3 a k černým číslů konstantu 0,5 (tím je můžu snadno seřadit podle své hodnoty). Pak jsem z pole čísel začalal vybírát náhodná čísla do array **cislarandom**[gee16], abych zamezil opakování čísel se stejnou hodnotou, vždy po hádání jsem dané číslo z array pole ubral a pole array zkrátil.

Následně jsem první čtyři čísla z array **cislarandom** přiřadil ke hráči číslo jedna (do řady **karta**) a další čtyři jsem přiřadil ke hráči dvě (do řady **karta2**). Dále jsem si přes for loop zjistil zdali některý z hráčů nemá kartu o hodnotě 12. Pokud některý z hráčů má kartu s touto hodnotou dostane novou náhodnou hodnotu od 0 do 11 a následně k této hodnotě bude přidáno 0.2 (pokud se jedná o bílou kartu) nebo 0.4 (pokud se jedná o černou kartu). Celý tento proces probíhá v metodách *before_start_player* a *before_start_Computer*.

4.2 Srovnání karet hry hráče s hráčem

Program zde využívá bubble sort od nejmenšího čísla po největší i od největšího po nejmenší číslo [bus], jelikož zde přechází z obrazovky jednoho hráče na obrazovku druhého hráče bubble sort se bude měnit (jednou dopředu, podruhé pozpátku). Karty se srovnávají v programu ve třech místech a to v metodě before*before_start_player*, a vždy po ukončení tahu hráče v metodách *player1_correcting* a *player2_correcting* (kvůli výměně a přidání nové karty).

4.3 Srovnání karet hry hráče s počítačem

Program zde využívá bubble sort od nejmenšího čísla po největší i od největšího po nejmenší číslo. Pro karty hráče je bubble sort od nejmenšího po největší číslo a pro počítač je bubble sort od největšího po nejmenší číslo. Karty se srovnávají v programu ve třech místech a to v metodách *before_start_Computer*, *player2_correcting* a *player1_correcting*

4.4 Způsob určování barvy

Velice podstatný faktor v mém programu hraje způsob určovaní barvy čísel. Ten určuji přes podmínku if tak že, pokud je zaokrouhlené číslo rovno tomu samému číslu vynásobené deseti od, kterého odečteme 5 nebo 4 (číslo s přidaným 0.4 představuje pomlčku) a které je následně vydělené znovu deseti, tak je číslo černé viz. příklad kódu níže.

```
if (Math.floor(karta[x]) == (((karta[x]*10)-5)/10) ||
(Math.floor(karta[x]) == ((karta[x]*10)-4)/10)){
    System.out.println("Karta je cerna");
}

if (Math.floor(karta[x]) == (((karta[x]*10)-3)/10) ||
(Math.floor(karta[x]) == ((karta[x]*10)-2)/10)){
    System.out.println("Karta je bila");
}
```

4.5 Způsob určování pomlčky

Pro určení pomlčky použiji stejný způsob jako na určení barvy (s tím že vím, že bílá pomlčka má hodnotu 0.2 a černá 0.4) viz. příklad kódu níže.

```
if (Math.floor(karta[x]) == (((karta[x]*10)-4)/10) ||
(Math.floor(karta[x]) == ((karta[x]*10)-2)/10)){
    System.out.println("Karta je pomlcka")
}
```

4.6 Integer break1

Integer break1 počítá kolik kroků programu zbývá dokud nedojdou karty na losování. Zamezí tak, aby se program po vyčerpání volných karet nezacyklil.

4.7 Volba nových karet u hry hráče s hráčem

Každý hráč si na začátku svého tahu musí vzít novou kartu. Na vyhledávání slouží int vy5 (pro hráče 1) a vy6 (pro hráče 2). Pokud je int 1 tak si hráč volí černou kartu, pokud je int 0 tak si hráč volí bílou kartu. Výběr karet probíhá v metodách *player2_colorpicking* a *player1_colorpicking*.

Výběr barvy funguje tak, že program za pomoci for loopu projde řadu **ciaslarandom** a zjistí zdali se v řadě nachazí černá či bílá (záleží na hodnotě) kladná karta stejné barvy. Pokud tomu tak je číslo z řady **cislarandom** se zapíše do int ran1 nebo ran2 (zase záleží na hráči na tahu), číslo v řadě **číslarandom** vynásobíme -1 a for loop ukončíme. Je-li vybrané číslo 12.5 nebo 12.3 (tedy pomlčka) hráč si zvolí i hodnotu pomlčky (více s kapitole Software grafické části - Výběr). Pokud tomu tak není, přidáme do pomocného intu cerna či bila + 1.

Jesliže int cerna či bila jsou větší nebo rovno 18, karty s danou barvou došly a hráč si musí zvolit jinou barvu.

4.8 Volba nových karet počítačem u hry hráče s počítačem

Volba nových karet počítačem u hry s počítačem funguje na stejném principu jako volba nových karet u hry hráče s hráčem s tím rozdílem, že hodnota barvy (0 nebo 1) je náhodně brána z rozmezí 0 až 1[Bae21]. Pokud dojde vybraná barva, rozmezí se sníží (dojde-li bíla, rozmezí se sníží na 0 až 0, dojde-li černá rozmezí se sníží na 1 až 1).

4.9 Hádání karty u hry hráče s hráčem

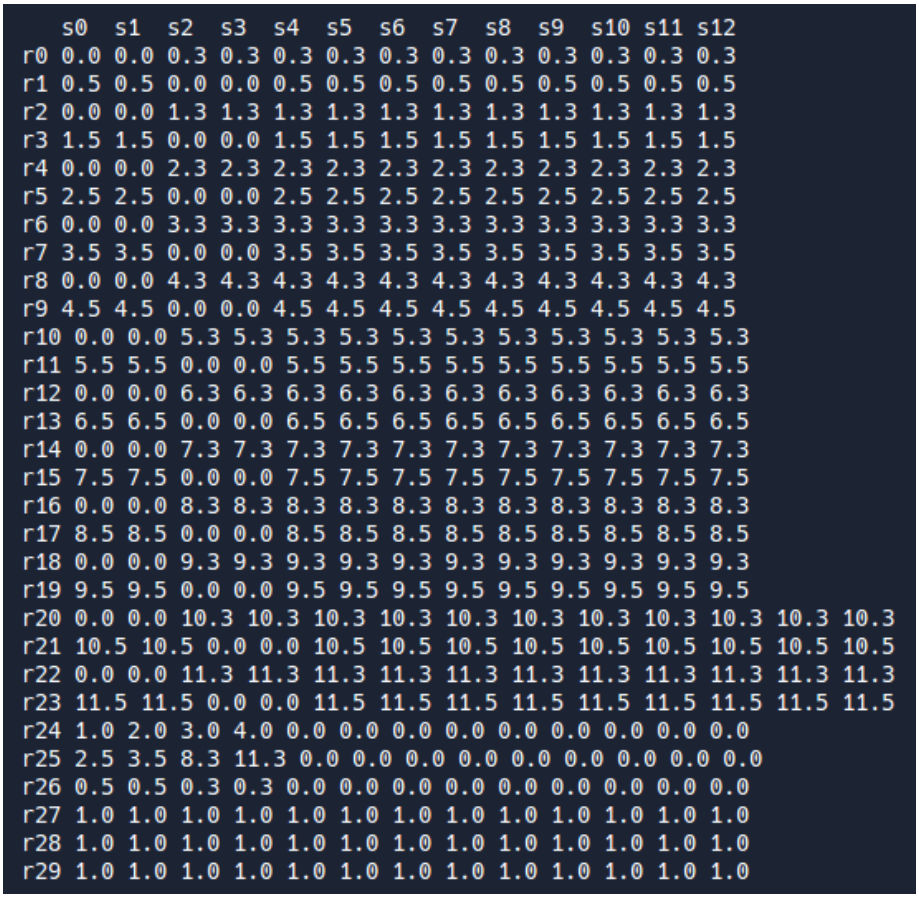
Abychom mohli hádat kartu, potřebujeme znát dva údaje: Na jaké pozici je hádané karta, a které číslo (nebo pomčku) chceme na kartu dosadit. Pokud se karta na dané pozici rovná hádané hodnotě karty, přepíšeme hodnotu v poli na zápornou (vynásobíme -1). Jestliže hádáme pomlčku, zjistíme zda-li karta na hádané pozici obsahuje pomlčku. To zjistíme pomocí if podmínky, a pokud tomu tak je, vynásobíme kartu -1. Pokud se karta na hádané pozici nerovná hádané hodnotě, tak pokračujeme na tah dalšího hráče.

4.10 Hádání karty počítačem

Aby počítač mohl hádat a určit karty protihráče musí znát možnosti z kterých může vybírat. Proto jsem si jako první vytvořil dvojrozměrné pole **arr** o velikosti 24 x 13, pak následně do každého sloupce zapsal za pomocí for loopu v metodě *before_start_Computer* všechny možné hodnoty soupeře tedy od 0.3 do 11.5 (každý jeden sloupec představuje potenciální možnosti pro danou hádanou kartu viz Figure 2., do nového sloupce se přidává nová karta protiháče). Následně jsem si uvědomil, že budu potřebovat další řádek (r24)

pole pro určení pozice karty v řadě druhého hráče (jelikož protihráč si bere nové karty, potřebuji tedy nový sloupec se všemi možnými hodnotami pro danou kartu).

Následně jsem přidal další řádky: r25 pro hádané čísla kvůli lepší orientaci, r26 na určení barvy dané karty (v řádku se nachází buď 0.3 nebo 0.5) r27 na určení černé pomlčky, r28 na určení bílé pomlčky (později se ukázalo, že tyto dva řádky nejsou potřeba k fungování programu) a nakonec r29 na určení zdali je karta již odhalena.



	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
r0	0.0	0.0	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
r1	0.5	0.5	0.0	0.0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
r2	0.0	0.0	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3
r3	1.5	1.5	0.0	0.0	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
r4	0.0	0.0	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3
r5	2.5	2.5	0.0	0.0	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5
r6	0.0	0.0	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3
r7	3.5	3.5	0.0	0.0	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5
r8	0.0	0.0	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3
r9	4.5	4.5	0.0	0.0	4.5	4.5	4.5	4.5	4.5	4.5	4.5	4.5	4.5
r10	0.0	0.0	5.3	5.3	5.3	5.3	5.3	5.3	5.3	5.3	5.3	5.3	5.3
r11	5.5	5.5	0.0	0.0	5.5	5.5	5.5	5.5	5.5	5.5	5.5	5.5	5.5
r12	0.0	0.0	6.3	6.3	6.3	6.3	6.3	6.3	6.3	6.3	6.3	6.3	6.3
r13	6.5	6.5	0.0	0.0	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5
r14	0.0	0.0	7.3	7.3	7.3	7.3	7.3	7.3	7.3	7.3	7.3	7.3	7.3
r15	7.5	7.5	0.0	0.0	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5
r16	0.0	0.0	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3
r17	8.5	8.5	0.0	0.0	8.5	8.5	8.5	8.5	8.5	8.5	8.5	8.5	8.5
r18	0.0	0.0	9.3	9.3	9.3	9.3	9.3	9.3	9.3	9.3	9.3	9.3	9.3
r19	9.5	9.5	0.0	0.0	9.5	9.5	9.5	9.5	9.5	9.5	9.5	9.5	9.5
r20	0.0	0.0	10.3	10.3	10.3	10.3	10.3	10.3	10.3	10.3	10.3	10.3	10.3
r21	10.5	10.5	0.0	0.0	10.5	10.5	10.5	10.5	10.5	10.5	10.5	10.5	10.5
r22	0.0	0.0	11.3	11.3	11.3	11.3	11.3	11.3	11.3	11.3	11.3	11.3	11.3
r23	11.5	11.5	0.0	0.0	11.5	11.5	11.5	11.5	11.5	11.5	11.5	11.5	11.5
r24	1.0	2.0	3.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
r25	2.5	3.5	8.3	11.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
r26	0.5	0.5	0.3	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
r27	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
r28	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
r29	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Figure 2: Graficky znázorněné pole **arr** před začátkem hry s vyřazenými hodnotami, které se mi nehodí

4.10.1 Vyřazení karet s jinou barvou

Pro další určení hádaného čísla musíme vyřadit nehodící se karty, tedy pro čísla s přidanou desetinou hodnotou 0.5 musíme z sloupce vyřadit všechna čísla s desetinou hodnoutou 0.3 a opačně. Toto vyřazení uděláme za pomocí for loopu.

4.10.2 Pořadník pro karty

Jelikož každý sloupec v poli **arr** má v sobě možné hodnoty pro danou kartu, musíme pro další hádání zjistit, na jaké pozici se tyto karty v kódu nacházejí (jelikož nová karta protihráče může rozhodit předchozí pozice karet). Na to jsem si vytvořil pořadník. Pořadník funguje tak, že ve for loopu zjištuji podmínkou if kolik karet je před nově přidanou kartou, následně pak posunu v pořadníku (v řádce 24 v řadě pole **arr**) karty s vyšší nebo stejnou hodnotou.

4.10.3 Vyřazení nehodících se karet

Abych zlepšil hádací algoritmus, musel jsem vyřadit hodnoty, které se k daným kartám nehodí. Toto vyřazení funguje tak, že pro vyřazení hodnot od nejvyšších pozic (tedy zleva do prava) začnu u karty nejvíce na levo, která může mít maximální hodnotu 11 pak si zjistím jakou barvu má karta vpravo vedle dané karty. Pokud má karta stejnou barvu, musí mít maximální hodnotu karta vpravo o jednu menší než je maximální hodnota předchozí karty. Pokud karta vpravo od karty na které se nacházím je bílá a karta na které se nacházím je černá, tak karta vpravo má stejnou maximální hodnotu jako karta na které se nacházím. Pokud toto uplatníme například u karet na obrázku Figure 3, tak zjistíme že maximální hodnota, kterou může mít karta na první pozici zleva je 11, hodnota kterou může mít karta na druhé pozici zleva je 10, hodnota kterou může mít karta na třetí pozici zleva je také 10, a hodnota kterou může mít karta na čtvrté pozici zleva může být pouze 9.

Tento postup se dá ještě uplatnit od nejmenší hodnoty po nejvyšší (zprava doleva) s tím, že budeme přidávat nejnižší maximalní hodnotu k dané kartě.

Tuto problematiku jsem vyřešil přes for loop s podmínkou if a pomocnými proměnými int barva a int novabarva.

Figure 3: Vizualizace možné kombinace karet druhého hráče

4.10.4 Výběr hádáné karty počítačem

Nejprve algoritmus vybere první volnou pozici pomocí for cyklu v pořadníku na řádku r24. Pak v příslušném sloupci tabulky **arr** najdu první hodnotu která není 0. Porovnám ji se soupeřovou hodnotou karty a pokud se rovnají (tedy počítač uhodnul soupeřovu kartu) tak v příslušném sloupci vynuluji řádky r0 až r23 a r29. Pokud se vybraná buňka nerovná hodnotě soupeřovy karty, tak nastavím buňku v tabulce **arr** na hodnotu 0.[gee22]

4.11 Přidání nové karty mezi existující karty hráče

Pro přidání nové karty do řady karta nebo karta2 musíme řadu rozšířit. Proto si nejdřív z dané řady přepíšeme čísla do pomocé řady kartare či karta2re. Pak danou řadu rozšíříme o jednu pozici. Následně pak z pomocné proměné přepíšeme čísla zpět do původní řady a do posledního pole řady přidáme novou hodnotu. Toto přiřazení probíhá v metodách *player1_correcting*, *player2_correcting* a *player2_correcting_computer*

5 Software grafické části

5.1 Menu

Menu je počáteční bod programu [Č]. Jsou v něm dvě tlačítka. První je tlačítko Start, které zavře současnou scénu a spustí scénu Výběru hry menu. Druhé je tlačítko End, které ukončí program.

5.2 Výběr hry menu

Ve výběru hry menu se nachází dvě tlačítka. Tlačíko s nápisem vs Computer a vs Player. Po zmáčňutí tlačítka s nápisem vs Computer se načte scéna *vyběr_pomlcky* [KM13] ve které si může hráč vybrat zdali chce hrát s počítačem s pomlčkou či bez pomlčky. Po zmáčňutí tlačítka s nápisem vs Player se načte scéna *vyběr_pomlcky_hrac* ve které si může hráč vybrat zdali chce hrát s hrát s pomlčkou či bez pomlčky.

5.3 Tlačítko start

Tlačítko start u hry s hráčem i u hry hráče s počítačem má za funkci načtení prvních čtyř karet obou hráčů a tlačítek pro výběr černých a bílých karet. Zároveň i určí barvu karet a zobrazí jejich hodnotu, případně pomlčku, pro hráče který je na tahu.

5.4 Karty hráčů

Karty jsem si zobrazil tlačítky[www]. Každé tlačítko má vlastní jméno a svůj vlastní Action event. Lichá tlačítka jsou určena pro hráče jedna a sudá tlačítka jsou určena pro hráče dvě. Zároveň tlačítka pro hráče jedna mají řadu btnk1 a poradnik1 a pro hráče dvě mají řadu btnk2 a poradnik2 (tyto řady určují pozici karty za daným tlačítkem po přeskupení)

5.5 Bílé a černé tlačítko u hry hráče s hráčem

Po zmáčknutí bílého tlačítka se spustí metoda *player1_colorpicking* či *player2_colorpicking* (opět záleží na tahu hráče), ta zjistí zdali je jestě k dispozici nějáká bíla karta, a pokud ano, tak bílé tlačítko zmizí a na jeho místě se ukáže tlačítko karty hráče s vylosovanou hodnotou (k určení správného vybraného tlačítka karty pomáhá pomocná proměná ds). Pokud není k dispozici žádná bílá karta tak tlačítko ztrátí svou funkci. Pro černé tlačítko to funguje stejně jako pro bíle.

5.6 Výběr hodnoty pro kartu s pomlčkou

Pokud počítač vylosuje kartu s pomlčkou, tak se zobrazí tlačítko vyber a textové pole txt2, které slouží hráči aby si vybral hodnotu pomlčky. Není-li hodnota pomlčky vybrána program vám nedovolí pokračovat. Je-li ovšem vybrána hodnota pomlčky, tlačítko i textové pole zmizí. Následně se hodnota kterou si hráč vybral přiřadí v metodě *player1_correcting* či *player2_correcting*.

5.7 Hádání u hry s hráčem

Hráč na tahu si musí nejprve vybrat černou nebo bílou kartu a pokud je zobrazená pomlčka tak k ní musí nastavit i její hodnotu. Pak musí vybrat kartu soupeře kterou chce hádat a do textového pole zadat hádanou hodnotu (0 - 11, popřípadě pomlčku). Pokud hráč všechny tyto podmínky splňuje tak může kliknout na tlačítko stopper, které za pomocí if podmínek ověří zdali je vše správně zadáno. Pokud by některá z podmínek neplatila, v chatu se ukáže proč nelze pokračovat v hádání. Pokud ovšem všechno platí, spustí se metoda *player1_position* či *player2_position*. Pokud hráč karu uhodnul tak program vyřadí funkčnost (btn.setDisable) tlačítka za uhodnutou kartou a zobrazí v tlačítku hodnotu karty či pomlčku. Pokud ovšem hádání nebylo úspěšné, program zneviditelní prvky na hracím poli (tlačítka karet, vyhledávač atd.) a zobrazí tlačítko enter.

5.8 Přeskupení karet u hry hráče s hráčem

Po zmáčnutí tlačíka enter se tlačítko zneviditelní a všechny prvky na hracím ploše se vrátí zpět do původního stavu. Zároveň se i zobrazí nové tlačítko s kartou kterou si hráč zvolil. V této metodě je

podstatný int move který určuje který číslo hráče na tahu a int ds, který určuje v jaké části programu jsem.

5.8.1 Přehození karet

Po každém zmáčknutí enter se prohodí souřadnice y tlačítek karet (spodní karty jdou nahoru, horní jdou dolů). K tomu mi pomáhá int ds, který následně až karty dojdou (až ds se bude rovnat 22) pomůže program dostat do stavu kdz se nevybírají žádné nové karty.

5.8.2 Pořadník

Hlavním cílem pořadníku je zařadit nové tlačítko na správnou pozici k ostatním tlačítkům. Tento krok provedeme pomocí for loopu, zjistíme si kolik karet je před nově přidanou kartou, následně pak posuneme v pořadníku karty s vyšší nebo stejnou hodnotou viz. kódu níže.

```
ran2_count = 0;
for (int t = 0; t < karta2.length; t++) {
    if (karta2[t] == ran2) {
        ran2_count++;
        break;
    } else {
        ran2_count++;
    }
}
ran2_count--;
System.out.print(ran2_count);
poradnik2[karta2.length - 1] = ran2_count;
for (int w = 0; w < karta2.length - 1; w++) {
    if (poradnik2[w] >= ran2_count) {
        poradnik2[w]++;
    }
}


position2_count = ran2_count;
btnk2[karta2.length - 1] = position2_count;

if (btnk2[0] >= position2_count) {
    tt = (int) btn2.getLayoutX();
    btn2.setLayoutX(tt + 130);
    btnk2[0]++;
}
if (btnk2[1] >= position2_count) {
```

```
    tt = (int) btn4.getLayoutX();
    btn4.setLayoutX(tt + 130);
    btnk2[1]++;
}
...
```

5.8.3 Změna barev

Jelikož přesouváme karty nahoru a dolu tak se bude barva tlačítek měnit (obrátí se hodnoty karet). Proto jsem si vytvořil for loop, který následně zjistí zdali je hodnota pozice karty rovna pro hodnotu pořadníku pro danou kartu viz. příklad dole. Pokud tomu tak je tak jsem určil zdali se jedná o kartu o černou nebo bílou následně jsem podle toho přiřadil tlačítku danou barvu[htm]

```
for (int i = 0; i < karta2.length; i++) {
   if (btnk2[0] == i) {
      System.out.print("Karta2 pozice na i je rovna btnk2[0] ");
   } else if (btnk2[1] == i) {
      System.out.print("Karta2 pozice na i je rovna btnk2[1]");
....
```

5.8.4 Funkčnost karet

Funkčnost karet se při změně hráče vrátí všem kartám zpět, ale vyřadí se fukčnost uhodnutých karet. K tomu mi pomůže for loop a pomocná řada **disable1** či **disable2** (ty určují která karta je uhodnutá). Zároveň se u uhodlých karet zobrazí text s její hodnotou.

5.8.5 Změna textu

Při změně hráče dochází k tomu, že text (v kterém je hodnota dané karty či pomlčka) tlačítek karty hráče, který byl na tahu se zakryje (přepíšu text dané karty na prázdnou hodnotu) a text tlačítek karty hráče který je na tahu se naopak zobrazí. Toto zakrytí a odkrytí jsem docílil za pomocí for loopu s pořadníkem který určí hodnotu či pomlčkou za danou kartou.

5.9 Chat

Chat v mém programu je ve skutečnosti seskupení pěti labelů. V chatu se zobrazuje hádání obouch hráčů a zobrazení zdali je potřeba něco doplnit v hádání. Pokaždé když je do chatu přidaná nová informace label [gee21] přepíše informaci pod sebou (label text5 si vezme text z labelu text4, label text4 si vezme text z labelu text3 atd.)

6 Konečné zhodnocení

Tuto práci hodnotím pozitivně. Dosáhl jsem všech požadovaných cílů a splnil jsem zadání práce.

References

[Bae21] Baeldung. Generating random numbers in a range in java, Apr 2021.

[bus] Java bubble sort descending order example.

[gee16] Generating random numbers in java, Oct 2016.

[gee21] Javafx: Label, Apr 2021.

[gee22] Java do-while loop with examples, Apr 2022.

[htm]

[KM13] Kapilkp, Filipe Marques, and Member 13392940. Open a new jframe on a button click, Mar 2013.

[www] Java jbutton - javatpoint.

[Č David Čápka. Lekce 2 - fxml a první formulářová aplikace v javafx.