1. Požadavky na systém a spuštění

Pro spuštění a správný běh aplikace je vyžadováno nainstalované JRE 7. Hardwarové požadavky jsou alespoň Intel Pentium processor T4500, 1 GB RAM a minimální rozlišení displeje 700x500 pixelů.

Spuštění se provede poklepáním na soubor „gkh.jar“.

2. Spuštění již vytvořené hry

Hra, kterou chceme hrát, se musí nacházet v adresáři „hry“. Dále v adresáři „obrazky“ se musí nacházet obrázky všech karet použitých ve hře.

Přepneme se do obrazovky pro zahájení hry a to přes „Soubor“ -> „Hrát“ (jedná se o obrazovku, která se zobrazí při zapnutí programu). Ze seznamu her vybereme tu, kterou chceme hrát. Nyní se musíme rozhodnout, zda chceme hru vytvořit nebo se připojit k již existující hře.

Pro vytvoření hry je nutné v kolonce „Počet hráčů“ vybrat celkový počet hráčů a v kolonce „Řízených počítačem“ vybrat počet hráčů, které bude řídit počítač. Dále je nutné vybrat, kterou umělou inteligenci chceme použít. Pokud chceme použít umělou inteligenci, která je speciálně napsaná pro danou hru (musí být vytvořena), potom zaškrtneme políčko „Vlastní umělá inteligence“. Pokud chceme použít zabudovanou umělou inteligenci společnou pro všechny hry, potom políčko necháme nezaškrtnuté. Nyní když už máme vybrány všechny parametry, tak stačí kliknout na tlačítko „Vytvořit“. Ten kdo hru vytvořil, má možnost před připojením všech hráčů hru ukončit kliknutím na tlačítko „Ukončit“. Po připojení všech hráčů se hra spustí.

Pro připojení se k již existující hře se musí vyplnit pole „IP pro připojení“, do kterého se píše IP adresa počítače, ke kterému se chceme připojit. Pokud je necháno prázdné, tak dojde k připojení k tomuto počítači. IP adresa počítače je vidět v kolonce „Tvoje IP“. Nyní už jen musíme kliknout na tlačítko „Připojit“ a počkat, dokud se nepřipojí všichni hráči.

3. Vytvoření nové hry nebo načtení již existující

Novou hru vytvoříme přes „Soubor“ -> „Vytvořit novou“. Nyní se nám zpřístupnila záložka „Návrh“, ve které se může přepínat mezi jednotlivými obrazovkami pro návrh hry. Hru si nyní můžeme uložit a to přes „Soubor“ -> „Uložit jako“. Umístění necháme takové, jaké se nám nabízí, abychom hru později mohli hrát. Pod názvem, pod kterým jsme hru uložili, ji bude možné později najít při vybíraní, kterou hru chceme hrát. Uložené hry mají příponu „.xml“, ta se ale při zadávání názvu nepíše. Nyní když již máme hru uloženou, tak pro další ukládání můžeme místo „Uložit jako“ používat „Uložit“. Dále pokud bychom chtěli pouze upravit již existující hru, tak ji načteme přes „Soubor“ -> „Načíst“.

4. Tvorba balíčku

V každé hře je třeba nastavit balíček karet. Do adresáře „obrazky“ nakopírujeme obrázky karet, které budeme chtít používat. Dále zde můžeme nalézt již hotové balíčky. Velikost obrázků, které se nacházejí v adresáři, je stejná jako velikost obrázků, které se použijí ve hře. Pro hru je vhodné použít karty o velikosti okolo 98x164 pixelů. Pomocí tlačítek „Uložit balíček“ a „Načíst balíček“ můžeme uložit námi vytvořený balíček pro budoucí použití resp. načíst již existující balíček.

Nyní se již přepneme do obrazovky pro tvorbu balíčku a to přes „Návrh“ -> „Karty“.

Při vytváření vlastního balíčku je třeba nejdříve nastavit parametry, které mohou karty mít. Klikneme na tlačítko „Změnit parametry“. Otevře se nám okno, ve kterém můžeme nastavit jaké parametry budou karty mít. Kliknutím na tlačítko plus se parametry přidávají a kliknutím na tlačítko mínus odebírají. Do vzniklých polí se napíšou názvy parametrů, které chceme používat. Parametr karty může být cokoliv, co chceme přiřadit ke každé kartě. Může se jednat například o barvu nebo hodnotu karet. Po kliknutí na tlačítko „OK“ se aplikují změny do balíčku. Po kliknutí na tlačítko „Storno“ dojde ke zrušení všech změn, které jsme provedli.

Každá karta má parametry „Počet“ a „Název“. „Počet“ udává kolikrát se daná karta v balíčku vyskytuje. „Název“ nám udává jednoznačné jméno karty. Do kolonky „Název“ si můžeme vepsat nový název karty a stisknout Enter. V horní části obrazovky v seznamu karet můžeme vidět, že došlo ke změne názvu karty. V tomto seznamu můžeme překlikávat mezi jednotlivými kartami balíčku. Tlačítka „Nová karta“ resp. „Odebrat kartu“ nám přidají resp. odeberou kartu z balíčku. Při kliknutí na tlačítko „Změnit obrázek“ si můžeme vybrat obrázek, který chceme aby karta měla.

V každém balíčku musí navíc být karta, která má název „pozadi“ a počet 0. Jedná se o zadní stranu karet.

5. Návrh vzhledu

Dále je třeba navrhnout vzhled výsledné hry. Přepneme se tedy do obrazovky pro tvorbu vzhledu a to přes „Návrh“ -> „Vzhled“. V horní části vybereme minimální a maximální počet hráčů, kteří mohou hru hrát. Maximální počet hráčů nesmí být větší než 4 a minimální nesmí být menší než 1.

Výsledná obrazovka hry je tvořena dvěma částmi. První částí je oblast, kterou má před sebou každý hráč. Druhou částí je oblast, která se nachází uprostřed. Tlačítky plus a mínus můžeme přidávat resp. odebírat jednotlivé části oblastí. U každé části se nastavuje, jak jsou karty uspořádány, pro koho jsou viditelné, umístění na obrazovce a název oblasti.

V následujících odstavcích si ukážeme jak nastavit vzhled pro hru „Prší“.

V našem případě bude mít každý hráč před sebou pouze karty, které má na ruce. Do kolonky vlevo napíšeme název „ruka“. Do vedlejších kolonek vyplníme souřadnice, kde se oblast nachází. Vzhledem k tomu, že se jedná o jedinou oblast u hráče, tak můžeme zvolit souřadnice [0,0]. Karty v této oblast se budou zobrazovat vedle sebe, tedy zaškrtneme políčko „vedle sebe“. Dále karty budou známy jenom hráči, proto zaškrtneme „pro hráče“. Při zaškrtnutí „na sobě“, by z karet byla vidět pouze vrchní karta. Pokud by se zašktlo „pro všechny“, potom by karty na dané pozici viděli všichni. Nakonec pokud by se zaškrtlo „pro nikoho“, pak dané karty neviděl nikdo.

V sekci střed se budou nacházet již dvě oblasti a to lízací a odkládací balíček. Tyto oblasti pojmenujeme „balicek“ a „odkladaci“. U balíčku nastavíme umístění [1,0], karty budou na sobě a nebudou viditelné pro nikoho. U odkladaciho nastavime umisteni [0,0], karty na sobě a viditelné pro všechny.

6. Pravidla

Nyní se dostáváme k vlastnímu psaní pravidel. Přepneme se do obrazovky pro psaní pravidel a to přes „Návrh“ -> „Pravidla“. Pravidla se píšou do horní části obrazovky. Po kliknutí na tlačítko „Kontrola“ dojde ke kontrole syntax. První nalezená chyba se zobrazí ve spodní části obrazovky. V této kapitole si popíšeme všechny konstrukce potřebné k napsání pravidel.

6.1 Struktura pravidel

Na začátku celého textu se deklarují globální proměnné. Poté již následují jednotlivá pravidla. Každé pravidlo se skládá z hlavičky, poté následuje deklarace proměnných a nakonec jsou jsou složené závorky, ve kterých jsou jednotlivé příkazy. Každý příkaz kromě „if“ a „for“ musí být zakončený středníkem.

6.2 Deklarace proměnných a porovnání

Proměnné se nám dělí podle dvou kritérií. První možnost jak se na proměnné můžeme dívat je jako na intové a Stringové. Druhý způsob je se na ně dívat jako na lokální a globální. Každou proměnnou, kterou chceme použít musíme zadeklarovat. Jediný rozdíl mezi globálními a lokálními proměnnými je ten, že globální se deklarují na začátku celého kódu, zatímco lokální se deklarují na začátku každého jednotlivého pravidla a lze je použít pouze v tom daném pravidle. Proměnné se deklarují tak, že napíšeme typ proměnné, potom následuje mezera a následně všechny proměnné daného typu, které chceme zadeklarovat oddělené čárkami. Deklarace proměnných každého typu je zakončena středníkem. Kdybychom chtěli zadeklarovat Stringovou proměnnou „pozdrav“ a intové proměnné „a“ a „b“, tak můžeme použít následující konstrukci (nezáleží na tom jestli nejdříve zadeklarujeme intové a potom Stringové proměnné nebo obráceně):

```
int a,b;
String pozdrav;
```

V pravidlech můžeme potřebovat umět porovnat Stringové proměnné. To bychom využili například, pokud bychom si hodnoty karet v prší nepojmenovali čísly, ale stringy a někde v pravidlech bychom potřebovali zjistit, zda je hodnota jedné karty větší než druhá. Pokud používáme porovnání pouze na rovnost, potom není třeba definovat žádné porovnávání. Deklarace porovnání musí být ještě před deklarováním globálních proměnných. Při deklaraci napíšeme „porovnani=“ a následuje seznam stringů oddělených čárkami a seřazených od nejmenšího po největší. Deklarace se ukončuje středníkem. V ukázce máme tři stringy, takové, že a<b<c.

```
porovnani=a,b,c;
```

Ve stringách, které chceme porovnávat, nebo v názvech proměnných by se měli používat pouze písmena anglické abecedy a čísla.

6.3 Umístění karet

V této části si popíšeme, jak se odvolat na určité místo, kartu na tom místě a jak zjistit nějaký její parametr.

Ke všem těmto věcem se využívá tečková notace. V první fázi musíme vybrat hráče, který nás zajímá. Hráči jsou označeni čísly a to popořadě od 0 do n-1, kde n je celkový počet hráčů. 0 má ten hráč, který hru zahájil. Pokud by nás zajímali karty, které jsou na středu, tak toto umístění je označené „stred“.

Když už máme vybraného hráče (střed), tak musíme zvolit oblast u daného hráče. Tu vybereme z možností, které jsme si nadefinovali při tvorbě vzhledu. Toto místo může být například „ruka“.

Pokud by nás zajímalo pouze místo, tak jsme již hotoví. Pokud, ale chceme získat konkrétní kartu, potom musíme pokračovat. Všechny karty v dané oblasti si můžeme představit jako seznam. Můžeme získat první kartu pomocí „first“, poslední pomocí „last“, a i-tou kartu pomocí „get(i)“, kde i může být libovolný výpočet. Pokud máme v nějaké oblast karty na sobě, potom někdy můžeme potřebovat vědět, jaká karta je vrchní. Je to vždy poslední karta v seznamu. Ještě můžeme potřebovat zjistit kolik karet, je na daném místě, což zjistíme pomocí „size“.

Nyní jsme dostali konkrétní kartu, což nám může v mnoha případech stačit, ale někdy potřebujeme zjistit nějaký parametr, který daná karta má. To se udělá napsáním názvu parametru, který nás zajímá. V ukázce vidíme, jak získat hodnotu parametru „barva“, třetí karty, kterou má druhý hráč na ruce (označení hráčů začíná 0, tedy druhý hráč je 1). Na druhém řádku ještě vidíme jak na stejném místě zjistit počet karet.

```
1.ruka.get(3).barva
1.ruka.size
```

6.4 Výpočty

V pravidlech můžeme potřebovat spočítat nějakou hodnotu. To se provede klasickým matematickým výrazem. Avšak jsou zde určitá omezení. Operace, které můžeme provádět jsou +, -, *, / a %. Tyto operace jsou po řadě klasické plus, mínus, krát, děleno a zbytek po dělení. Zbytek po dělení je nejprioritnější operace, potom následuje krát a děleno a nakonec jsou plus a mínus. Při stejné prioritě se postupuje zleva. Další co lze použít jsou závorky a to „(“ a „)“. V matematických výrazech můžeme používat pouze intové proměnné a čísla. Pokud bychom chtěli do výpočtu zahrnout například hodnotu parametru nějaké karty, tak si tuto hodnotu nejdříve musíme uložit do proměnné a potom ji lze teprve použít ve výpočtu. Zde máme ukázku takového matematické výrazu, kde „a“ je intová proměnná.

```
(1+a*2)%3
```

6.5 Přiřazení

Hodnotu do proměnné přiřadíme jednoduše pomocí „=“. Veškeré stringy musí být v uvozovkách. Někdy je třeba do proměnné přiřdit hodnotu, kterou si zvolil

hráč jako například při výběru barvy po zahrání svrška v prší. To se řeší příkazem „show“, který jako parametry bere hodnoty, ze kterých lze vybírat. Při jeho použití se hráči zobrazí nabídka s možnostmi a do proměnné se uloží ta, kterou si vybral. V prší bysme tedy pro výběr barvy použili tuto konstrukci.

```
b=show("cervena","kule","zelena","zaludy");
```

6.6 Podmínky

Důležitou částí všech pravidel jsou podmínky. Podmínky si rozdělíme na jednoduché a složené.

Nejdříve se podíváme na jednoduché. V nich můžeme buď porovnávat dvě hodnoty nebo zjišťovat, zda na nějakém místě je určitá karta. Porovnávat můžeme jednoduše pomocí operací „=“, „<“ a „>“. Zjišťování zda je na daném místě karta s určitými vlastnostmi se dělá pomocí tečkové notace podobně jako při určování místa. Nejdříve určíme místo, které nás zajímá a potom použijeme slovíčko „contains()“, které do zavorky dostane podmínku určující a jakou kartu nám jde. Následující podmínka nám ukazuje, jak zjistit, že hráč 0 má na ruce kartu, která má parametr „barva“ nastaven na „cervena“.

```
0.ruka.contains(barva="cervena")
```

Složené podmínky je několik jednoduchých podmínek složených dohromady pomocí spojek „&&“ a „||“. Narozdíl od jiných programovacích jazyků, ale mají zde tyto spojky trochu jiný význam. Pokud podmínky před libovolným „&&“ jsou vyhodnoceny jako nepravdivé, potom je celá podmínka vyhodnocena jako nepravdivá. Pokud v podmínkách, které jsou oddělené „||“, je alespoň jedna pravdivá, potom se celá tato skupina bere jako pravdivá. Ukázka je pravdivá pokud se proměnná „a“ rovná 0 a následně platí, že proměnná „b“ je menší než 8 nebo je „b“ větší než 16.

```
a=0&&b<8||b>16
```

6.7 Pravidlo start

Nyní již máme dostatek znalostí, abychom se mohli podívat na celé pravidlo. Jednotlivá pravidla nám definují co se má stát po určité akci hráče nebo v určité chvíli. Pravidlo „start“ definuje, co se má stát po zahájení hry. Jedná se o jedno ze dvou pravidel, které musí mít každá hra. Akce, která muže nastat je zamíchání a rozdání karet. Hlavička tohoto pravidla začíná slovem „start“ následováno prázdnými závorkami. V ukázce máme pravidlo „start“, ve kterém budeme potřebovat intovou proměnnou „a“.

```
start()
int a;
{
...
}
```

6.8 Pravidlo konec

Jedná se o druhé pravidlo, které musí mít každá hra. Toto pravidlo se dělí na dvě části. První část nám určuje, zda nastal konec hry a jak na tom jsou jednotlivý hráči. Druhá část definuje co se má stát, pokud nastal konec hry. Typicky to asi bude ukázání výsledků a zahájení další hry. Každá z těchto dvou částí je ve vlastních složených závorkách. Hlavička tohoto pravidla začíná slovem „start“ následováno prázdnými závorkami. V tomto pravidle musíme zadeklarovat několik proměnných. První je Stringová proměnná „konec“, do které uložíme hodnotu „true“, pokud vyhodnotíme, že nastal konec hry. Další proměnné, které musíme zadeklarovat jsou intové proměnné „h0“ až „hn“, kde „n“ je počet hráčů mínus jedna. tyto proměnné slouží umělé inteligenci k vyhodnocování hry. Proměnná „hi“ odpovídá i-tému hráči. Čím je tato proměnná vyšší, tím je na tom daný hráč lépe. Správné nastavení těchto proměnných je důležité ke správnému fungování umělé inteligence. V ukázce vidíme toto pravidlo pro 2 hráče.

```
konec()
String konec;
int h0,h1;
{
...
}
{
...
}
```

6.9 Pravidlo klik

Další pravidlo je „klik“, které reaguje na to, že hráč kliknul na nějakou kartu. To většinou znamená zahrání nějaké karty nebo si líznutí další karty z balíčku atd. O tento typ akcí se nám starají pravidla typu klik. Hlavička takového pravidla začíná slovem „klik“ a následuje seznam parametrů v závorkách oddělených čárkami. Pro toto pravidlo máme celkem máme čtyři parametry. První určuje, u kterého hráče došlo k akci. Zde jsou čtyři možnosti, co napsat. První je „this“, což nám určuje, že ke kliknutí na kartu došlo u hráče, který je zrovna na řadě. Druhá možnost je „other“, tedy že došlo ke kliknutí u hráče, který není na řadě. Potom můžeme použít „all“, což nám určuje libovolného hráče. Poslední možností je „stred“, tedy kliknutí na kartu na středu.

Druhý parametr je oblast, ve které se karta nachází. V případě, že se kliklo na kartu u hráče, tak oblast může být třeba „ruka“.

Třetí parametr parametr nyní přeskočíme a podíváme se rovnou na čtvrtý. V tom se jedná o podmínku. Dané pravidlo se použije pouze pokud je splněna zadaná podmínka. Může se jednat o naprosto libovolnou podmínku. Například se může jednat o podmínku, která zkontroluje, že karta, na kterou se kliklo, je eso. Pokud chceme, aby podmínka byla vždy splněna, potom místo ní napíšeme „all“.

Nyní se vrátíme ke třetímu parametru. Jedná se o prioritu, se kterou se dané pravidlo použije. Pokud by šlo v daný okamžik použít dvě pravidla, potom se použije to, které má jako prioritu nasteveno větší číslo.

V tomto pravidle můžeme chtít používat kartu, na kterou se kliklo. Pokud chceme zjistit hodnotu nějaké parametru té karty, tak stačí napsat pouze název toho parametru. Pokud chceme použít celou kartu, tak použijeme slovo „this“.

V ukázce máme pravidlo, které se použije při kliknutí na balíček karet na středu, když je barva karty, na kterou se kliklo, zelená. Toto pravidlo budeme chtít využít vždy když není nalezeno žádné jiné pravidlo, tak prioritu nastavíme na 0. Nebudeme deklarovat žádné proměnné.

```
klik(stred,balicek,0,barva="zelena")
{
...
}
```

6.10 Pravidlo button

Jedná se o poslední typ pravidel. Funguje podobně jako pravidlo „klik“, ale místo aby reagovalo na kliknutí na kartu, tak reaguje na kliknutí na tlačítko. Hlavička začíná slovem „button“ a následují parametry v závorkách. Parametry jsou dva a jsou stejné jako u pravidla „klik“. První je určení hráče a druhý určení oblasti. Totopravidlo se použije, pokud někdo kliknul na tlačítko, které se nachází na dané pozici. Pravidlo z ukázky reaguje na kliknutí na tlačítko u hráče, který je na řadě, a tlačítko je v oblasti, která se jmenuje „oblast_pro_tlacitko“.

```
button(this,oblast_pro_tlacitko)
{
...
}
```

6.11 Příkaz if

Dovnitř do pravidel se píší jednotlivé příkazy. První z nich je „if“. Tento příkaz vyhodnotí zadanou podmínku, a pokud platí, tak se provedou příkazy, které následují ve složených závorkách. Potom může následovat slovo „else“, po kterém následují další příkazy ve složených závorkách, které se vyhodnotí, pokud podmínka neplatí. V ukázce je příkaz „if“ s jednoduchou podmínkou, která zjistí jestli v intové proměnné „a“ je 0.

```
if(a=0)
{
...
}
else{
...
}
```

6.12 Příkaz for

Další příkaz je „for“. Ten slouží k cyklickému opakování příkazů, které dostane ve složených závorkách. Cyklus se provede tolikrát, kolikrát má zadáno v parametrech. Má tři parametry. První je proměnná, do které se ukládá hodnota, která momentálně v cyklu je. Další dva parametry jsou výpočty určující, od které do které hodnoty má cyklus jít. V ukázce je cyklus, který ukládá do proměnné i a jde od 0 do 4.

```
for(i,0,4)
{
...
}
```

6.13 Příkaz move

Příkaz „move“ slouží k přesouvání karet. Bere dva parametry. První je jakou kartu chci přesunout a druhý kam jí chci přesunout. Tímto příkazem mohu přesovat buď jednotlivé karty nebo i celé oblasti. V ukázce první „move“ přesune poslední kartu z ruky nultého hráče na odkládací balíček, který je na středu. Druhý „move“ přesune všechny karty.

```
move(0.ruka.last,stred.odkladaci);
move(0.ruka,stred.odkladaci);
```

6.14 Příkaz next

Příkaz „next“ slouží k určování, který hráč je na řadě. Jiný hráč, než který je na řadě, nemá možnost provést jakoukoliv akci. Pokud chceme, aby nějaký hráč odpověděl na akci hráče, který je zrovna na řadě, tak ho musíme pomocí příkazu „next“ přesunout na tah. „next“ bere jako parametr výpočet, který určuje, který hráč je na řadě, relativně od aktuálního hráče. Pokud bysme chtěli tah předat konkrétnímu hráči, potom tento výpočet musíme dát do absolutní hodnoty. Ukázka dá na tah hráče, který je před aktuálním hráčem.

```
next(-1);
```

6.15 Příkaz showinf

Tento příkaz použijeme, pokud budeme chtít hráčům ukázat nějakou informaci. Jako parametr bere text, který se má zobrazit. Tento text se zobrazí všem hráčům. Pokud bychom chtěli všem hráčům říct, jaká hodnota je v proměnné „b“, tak použijeme následující konstrukci.

```
showinf(b);
```

6.16 Příkazy label a button

Příkazy „lablel“ a „button“ se používají stejně, proto si zde popíšeme pouze příkaz „label“. Slouží k zobrazování textu na zadaném místě. Má celkem dvě konstrukce. První konstrukce bere dva parametry. Prvním parametrem je místo a druhým je text, který se má zobrazit. Druhá konstrukce má pouze jeden parametr a tím je místo. Druhá konstrukce slouží k odebrání textu ze zadaného místa. Na každém místě by měli buď karty,tlačítko nebo text, ale ne kombinace popřípadně více než jedno tlačítko nebo text. „button“ slouží úplně stejně, akorát na dané místo přidává nebo odebírá tlačítko, na které může potom hráč klikat. V následující konstrukci odebereme z místa „pro_tlacitka“ na středu text a přidáme tlačítko s textem „Ukoncit“.

```
label(stred.pro_tlacitka);
button(stred.pro_tlacitka,"Ukoncit");
```

6.17 Příkaz shuffle

Posledním příkazem je příkaz „shuffle“. Ten slouží k zamíchání karet. Bere dva parametry. Prvním parametrem je jaká karta nebo oblast se má zamíchat. Místo karty nebo oblasti můžeme zadat „all“. V takovém případě se zamíchají úplně všechny karty. Druhý parametr je místo kam se mají karty zamíchat. V ukázce se zamíchají všechny karty z ruky hráče „1“ do balíčku na středu.

```
shuffle(1.ruka,stred.balicek);
```

7. Umělá inteligence

Pokud bysme nechtěli používat vestavěnou umělou inteligenci a chtěli si napsat vlastní, tak se můžeme přepnout do obrazovky pro psaní umělé inteligence („Návrh“ -> „UI“). Je stejná jako obrazovka pro psaní pravidel. Tedy do horní části obrazovky píšeme naší umělou inteligenci a po stisknutí tlačítka „Kontrola“ dojde ke kontrole syntaxe a vypsání první nalezené chyby do spodní části obrazovky.

7.1 Struktura umělé inteligence

Umělá inteligence začíná deklarací proměnných a potom následují ve složených závorkách jednotlivé příkazy, které definují co se má stát, pokud se dostane počítač na řadu. Lze používat všechny globální proměnné zadeklarovaných v pravidlech. Avšak tyto proměnné se nesmí měnit. Některé části jsou stejné jako v pravidlech. V umělé inteligenci se smí používat příkazy „if“ a „for“, které se používají stejně jako v pravidlech. Dále se stejně přiřazuje do proměnných, ale nelze přiřadit pomocí „show“.

7.2 Příkaz klik a button

V umělé inteligenci se nachází i několik nových příkazů. První z nich jsou příkazy „klik“ a „button“. Ten simuluje kliknutí na kartu resp. na tlačítko. Oba berou jediný parametr, a tím je karta resp. tlačítko, na které se má kliknout. Následující konstrukce simuluje kliknutí na i-tou kartu, kterou má hráč, který je zrovna na řadě, na ruce.

```
klik(tah.ruka.get(i));
```

7.3 Příkaz show

Příkaz „show“ simuluje vybrání hodnoty, kterou nabízí stejnojmený příkaz z pravidel. Jako parametr bere vybranou hodnotu. Pokud bychom tedy chtěli u prší zvolit barvu po zahrání svrška, a tou barvou by byli žaludy, potom bysme použili následující konstrukci.

```
show("zaludy");
```