Aleksandr Kartavykh

Dokumentace k semestrální práci z předmětu **A0B36PRI**

Obsah dokumentu

Úvod

Ve své semestrální práci jsem měl naprogramovat jakoukoliv hru s možností uložení rozehrané hry. Rozhodl jsem si pro piškvorky, vždyť tuhle na první pohled jednoduchou hru zná a miluje z dětství skoro každý. Navíc zde je velký potenciál pro modifikaci a zlepšení kódu, který by se dalo využit v budoucnu, kdyby bylo třeba.

Moje hra má několik režimu, jak standardní Player vs Player, tak i solo režim pro hru proti počítači, který má tři úrovně obtížnosti. Také po konci každé hry výsledné herní pole je uloženo do externího .txt souboru s finálním rozložením znáků a s poznámkou o vítěze hry.

Popis metod

1. **public static void main**

   Zde je volána metoda hlavního menu pro začatek hry.

2. **public static void mainMenu()**

   V této metodě je pár operátoru typu **switch** pro volbu režimu hry a cykly **do while** pro opatření chybných voleb a špatných zadání dat. Také zde lze nalézt operátor **try-catch** pro případ zadání textu (vždyť object třídy Scanner chce vidět číslo na vstupu, jinak by házel výjimku **InputMismatchException**).

3. **public static void modeTwoPlayers()**

   Tato metoda je zodpovědna za režim pro dva hrače. Inicializaci pole následuje nekonečný cyklus **while (true)** obsahující volání metod pro kroky hračů(**userShot**, **aiShot**) a který je ukončen buď vítězstvím jednoho z hračů anebo dosažením proměnnou **count** hodnoty 9 (původně je nastavena na 0). Pro pole 3x3 je možné udělat maximálně 9 herních kol, a po každém kolu se proměnná **count** zvětšuje o 1 a pomocí operátoru **if** se ověřuje podmínka vítězství (metoda **checkWin**). Tj. v případě když nevítězí nikdo, hra končí a je nahlašena remíza. Bez ohledu na výsledek hry je finální herní pole uloženo do .txt souboru(metoda **WriteFile**).

4. **public static void modeAgainstAI() + public static void modeAgainstHardAI()**

   Tyhle metody jsou docela podobné minulé metodě, akorát v nich funkci druhého hrače vykonává počítač. Jediným rozdílem mezi nimi dvěma je, že v té druhé počítač začíná hru jako první.

5. **public static void initField()**

   Zde probíhá inicializace pole v tom slovasmyslu, že se všem hodnotám pole přiřádí proměnná typu **String**, která označuje neobsazené buňky.

6. **public static void printField()**

   Tohle je jednoduchá metoda pro výpis aktuálního stavu herního pole na obrazovku. Volá se na začátku hry a na začátku každého herního kola podle potřeby.

7. **public static void userShot (String sign)**

Tady je schován algoritmus pro kolo hrače. Pomocí cyklu **do while** a operátoru **if** jsou provedena opatření proti špatným zadaním dat a volbám již obsazených buněk, jinak se tam jednoduše pomocí objektu **Scanner** získávají souřadnice buňky, kam se potom přiřazuje proměnná typu **String** se znakem odpovídajícího hrače.

Jako vstupní proměnnou používá odpovídající znak hrače.

8. **public static void aiShot()**

Tohle je metoda, kde jsou realizovány začatky umělé inteligence. Jak jsem už zmínil v úvodu, jsou tři úrovně obtížnosti. Nejjednodušší úroveň reprezentuje konec metody, kde jsou souřadnice kroku počítače generovány náhodně pomocí objektu třídy **Random**. Pro druhou úroveň obtížnosti je dodáno zabranění hrači ve vítězství. Program analyzuje každou volnou buňku pole tak, že se tam dosadí znak hrače a ověřuje se podmínka vítezství(metoda **checkWin**), potom se buňce zpatky přiřádí znak neobsazenosti. Pokud je vítězná buňka nalezena, pak se souřadnice ukládají, a počítač označuje tu buňku svým znakem.

Pro třetí úroveň počítač stejným způsobem analyzuje podmínku svého vítězství, a když takovou možnost najde, pak se nebude váhat a využije ji.

9. **public static boolean isCellBusy (int x, int y)**

Zde se ověřuje, zda je buňka volná. Jako vstupní proměnné se berou souřadnice zvolené buď počítačem, nebo hračem.

10. **public static boolean checkLine(int start_x, int start_y, int dx, int dy, String sign)**

Tato metoda je založena na principu rovnice přímky a za správně zvolených vstupních proměnných prochází celé herní pole. Při nalezení tří za sebou jdoucích stejných znaků vrací hodnotu **true**.

11. **public static boolean checkWin(String sign)**

V této metodě, která za vstupní proměnnou má znak hrače a která je volána na konci každého herního kola, je postupně a s určitými proměnnými volána

předchozí metoda. Pokud se vratí hodnota **true**, pak tahle metoda také vrací **true** což znamená konec hry. Jinak vrací **false** a hra se pokračuje.

12.public static void WriteFile(int a)

Tohle je poslední metoda, která zodpovídá za uložení výsledku hry do .txt souboru. Nejprvé je definován ukazatel na objekt třídy **FileWriter**, potom se deklaruje proměnná **StringBuilder**, do níž je pak pomocí operátoru **for** postupně zapsáno finální herní pole a pomocí operátoru **switch** je zvolena poznámka o vítěze dané hry. Nakonec probíha konvertace **StringBuilder toString** a výsledek je uložen.

Na vstupu má proměnnou **WINNER**, která může nabývat hodnoty v rozmezí 0 až 4, přiřazení probíhá na konci hry a je závislé na způsobu konce hry. Např. v případě remízy je ji přiřazena hodnota 0 a na konec souboru jde odpovídající poznámka.

Navíc tahle metoda má ve hlavičce **throws IOException**, stejně jako všechny metody, které ji volají. Tohle je nezbytné nahlašení výjimky IOException, která může vzniknout kvůli chybě při ukládání dat do souboru. Další opatření výjimky jsem neprováděl (ve srovnání s opatřením výjimky **InputMismatchException** pro objekt typu **Scanner**).

Závěr

V této semestrální práci jsem sepsal metody potřebné pro několik režimů hry Piškvorky. Pro řešení dané úlohy jsem použil procedurální přístup, i když ten objektový je žadoucí pro složitější programy a je celkem univerzálnější.

Také se mi podářilo stvořit dost chytrou umělou inteligenci pro režim jednoho hrače. Na nejtěžší úrovni hra proti počítači zpravidla končí remízou, ovšem možnost vítězství tam je, jenom až se uživateli podáří vyvinout správný postup, který však nezaručí 100% úspěch (průměrně se dá zvítězit v každé páté hře).