KIV/UPG
2022/23
2. odevzdání **Celkový počet strávený na řešení SP dosud: 80 h**

Obsah

Seznámení s problematikou

1. fáze

Cílem první fáze semestrální práce je vytvořit program, který po spuštění vykreslí v okně o rozměrech **800 × 600 px** šachovnici s figurkami ve výchozích pozicích. Program by měl dále reagovat na podněty od uživatele, jako je například **drag & drop** jednotlivých figurek, nebo přizpůsobení se změnám velikosti okna a velikosti jednotlivých figurek na šachovnici.

Konkrétní body zadání:

- Výchozí okno o velikosti **800 × 600 px**.
- Čtvercová šachovnice zabírající maximální možný prostor okna a vždy umístěná ve středu.
- Vektorově vykreslené figurky ve výchozích pozicích dle pravidel šachu.
- Reakce programu na změny velikosti okna (okno se překreslí tak, aby byl zachován správný vzhled a rozložení).
- **Drag & drop** figurek (dojde-li k přesunu na obsazené pole, původní kámen bude odstraněn).

2. fáze

Cílem druhé fáze semestrální práce je program doladit do takového stavu, aby splňoval veškeré náležitosti šachové hry dle oficiálních pravidel.

Konkrétní body zadání:

- Pohyby figurek (zobrazení možných tahů figurky)
- Braní „mimochodem“ pěšců
- Rošády krále
- Šach
- Informování uživatele o konci hry a zobrazení výsledku (Mat, Pat, remíza)
- Možnost vracení tahů

Implementace

Základní popis tříd projektu

Figurky → abstraktní třída `AFigurka`

Obsahuje základní metody pro práci s figurkou a pro nastavování a získávání klíčových atributů figurky. Implementační metody a atributy potřebné pro konkrétní typy figurek jsou uvedeny ve třídách s názvy jednotlivých figur (např. `Vez`, `Kral`, `Pesec` atd.). Tyto třídy dědí od `AFigurka` a obsahují implementační detaily pro danou figurku – například získání možných tahů (každá figurka má jiné pohyby, jiné tělo apod.).

Řízení hry a šachovnice → třída `Sachovnice`

Obsahuje veškerou logiku pro řízení hry na šachovnici, manipulaci s figurkami a metody a atributy potřebné pro správu stavu šachovnice.

Vykreslení hry do okna a propojení tříd → třída `DrawingPanel`

Zajišťuje propojení a volání veškeré logiky v projektu potřebné pro vytvoření panelu, kreslení do okna, zpracování vstupů uživatele (výběr figurky, umístění figurky, drag & drop, proměna pěšců, nová hra, vracení tahů) a celkové řízení programu. Ve třídě `DrawingPanel` jsou implementovány metody, které určují běh programu. V metodě `paint()` se pomocí metod třídy `Sachovnice` vykreslí šachovnice s figurkami v aktuálním stavu hry.

Vracení tahů a určení speciálního případu remízy → třída `ForsythEdwardsNotation`

Obsahuje logiku pro vytváření řetězců podle standardu **Forsyth–Edwards Notation (FEN)**, které slouží pro vracení tahů ve hře a detekci remízy v případě, že se třikrát opakuje stejný stav šachovnice během šachové partie.

Uchování stavu figurek → třída `StavFigurky`

Slouží k uchování stavů figurky (např. klíčových atributů) v případech, kdy se filtrují možné tahy tak, aby nedošlo k ohrožení vlastního krále (např. při kontrole legality tahu). Figurky se při tomto ověřování interně přesouvají, stav hry se posoudí a následně se vrátí do původního stavu.

Popis řešení

Celá hra se vykresluje do komponenty `DrawingPanel`, která dědí od `JPanel`. Šachovnice je reprezentována dvourozměrným polem `AFigurka[][]`, kde každé pole buď obsahuje instanci figurky, nebo `null`, pokud je prázdné.

Všechny instance figurek jsou navíc uloženy v poli `AFigurka[]` pro efektivnější zpracování. Výběr a přesun figurek (drag & drop) je realizován pomocí posluchačů (`listeners`).

Figurky si generují možné tahy, reprezentované jako pole `Rectangle2D[]`, které popisují základní pohyby figurky (bez kontroly šachu).
Před samotným provedením tahu se tyto tahy filtrují – odstraní se ty, které by vedly k ohrožení vlastního krále. Toto je řešeno tak, že figurka je dočasně přesunuta a stav hry se následně vyhodnotí.

Pro vracení tahů a detekci remízy je využíván algoritmus **Forsyth–Edwards Notation (FEN)**, který pomocí textových řetězců efektivně uchovává jednotlivé stavy hry. Tento přístup je časově méně náročný než ukládání celé šachovnice jako trojrozměrného pole.

Využité datové struktury

Nepoužívají se žádné speciální datové struktury. Používány jsou pouze základní typy, jako jsou pole a `ArrayList` – například pro uchovávání FEN řetězců.

Využité algoritmy

Forsyth–Edwards Notation (FEN):

Uchovává stav šachovnice během hry. Každý FEN řetězec reprezentuje stav šachovnice po jednom tahu. Figurky jsou reprezentovány znaky:

- **Q/q** – dáma
- **K/k** – král
- **P/p** – pěšec
- **N/n** – kůň
- **B/b** – střelec
- **R/r** – věž
- **číslice** – počet po sobě jdoucích prázdných polí v řádku

Řádky šachovnice jsou odděleny lomítky (`/`). V poslední části řetězce je uvedeno:

- kdo je na tahu,
- jaké rošády jsou povoleny pro oba hráče
- počet tahů bez braní či pohybu pěšcem (pro pravidlo 50 tahů → případ remízy).

Příklady FEN řetězců:

rnbqkbnr/pppppppp/8/8/4P3/8/PPPP1PPP/RNBQKBNR/ h2 KQ kq 0

r4knr/pp3ppp/2n5/2pP4/QbP1p2q/NP5b/PB1P1PP1/2KR1BN1/ h2 -- -- 4

rnb1k1nr/ppp2ppp/8/3pp3/QbP1P2q/7R/PP1P1PP1/RNB1KBN1/ h2 -Q kq 1

r4knr/pp3ppp/2n5/2pP4/QbP1p2q/NP5b/PB1P1PP1/R3KBN1/ h1 -Q -- 2

Popis vytvoření, instalace a spuštění aplikace

Uživatel si stáhne zazipovaný soubor, který následně rozbalí ve svém souborovém systému. Spouštěcí soubor `Run` se nachází v kořenovém adresáři `Sachy`. V místě, kde se tento soubor nachází, si uživatel otevře terminál a zadá příkaz:

- `bash Run.sh` (Linux)
- `Run.cmd` (Windows Terminal)
- `.\Run.cmd` (Windows PowerShell)

Tímto se spustí celá aplikace.

Pro úspěšné spuštění programu je třeba mít nainstalovanou Javu verze 21 nebo vyšší.

`Run.sh` → Skript nejprve spustí build projektu. Pokud se build nezdaří, vypíše chybovou hlášku a ukončí běh. Následně spustí skript pro vytvoření dokumentace. Pokud při generování dokumentace dojde k chybě, vypíše varování, ale pokračuje dál. Nakonec spustí příkaz `java -cp ./bin Main "$@"`, který spustí program. Tímto příkazem se využijí přeložené class soubory uložené ve složce *bin* (zakódované v UTF-8). Program se spustí od metody `main` ve třídě `Main`.

`Run.cmd` → Funguje na stejném principu jako `Run.sh`, pouze je přizpůsoben pro prostředí Windows.

`Build.sh` → Skript nejprve vytvoří složku `bin`, pokud ještě neexistuje. Poté přeloží všechny `.java` soubory ze složky `src` do složky `bin` pomocí příkazu `javac -cp ./src -encoding UTF-8 -d ./bin ./src/*.java`.

`Makedoc.sh` → Skript nejprve vytvoří složku `doc/javadoc`, pokud ještě neexistuje. Poté vygeneruje dokumentaci projektu pomocí Javadoc příkazu `javadoc -encoding UTF-8 -sourcepath ./src -cp ./src -d ./doc/javadoc -version -author ./src/*.java`. Dokumentace je uložena ve složce `doc/javadoc` a obsahuje informace o třídách, metodách a atributech projektu.

`Build.cmd` → Funguje na stejném principu jako Build.sh, pouze je přizpůsoben pro prostředí Windows.

`Makedoc.cmd` → Funguje na stejném principu jako Makedoc.sh, pouze je přizpůsoben pro prostředí Windows.

Popis ovládání aplikace

Ovládání aplikace je velmi intuitivní:

- **Výběr figurky**: Uživatel najede kurzorem na tělo figurky a stiskne libovolné tlačítko myši.
- **Přesun figurky**: Myší figurku přetáhne na cílové pole na šachovnici. Po celou dobu tahu je třeba mít tlačítko myši stisknuté.
- **Umístění figurky**: Provedením tahu do cílového pole dojde k puštění tlačítka myši.

Další funkce:

- **Ukončení programu**: kliknutím na křížek v pravém horním rohu okna.
- **Zahájení nové hry**: stisknutím klávesy `n`.
- **Vrácení posledního tahu**: stisknutím klávesy `z`.

Popis dosud neopravených nedostatků a možného rozšíření do budoucna

Vylepšení

Třída `Sachovnice`

Jak je zřejmé, projekt by šel navrhnout mnohem více objektově, čímž by se výrazně snížila složitost některých tříd – například třída `Sachovnice` má přibližně 1500 řádků. Spojit logiku hry, vykreslování a šablonu šachovnice do jedné třídy nebyl nejšťastnější návrh. Lepším řešením by bylo vytvořit:

- **Třídu `KontrolerHry`:** která by obsahovala logiku pro řízení hry (filtrování možných tahů figurek, přesuny na šachovnici, braní figurek, určování šachu, kontrolu braní mimochodem, rošád a proměn, určení, kdo je na tahu, a kontrolu koncových stavů hry).
- **Třídu `Vykreslovač`:** která by se starala o vykreslování šachovnice, figurek a různých upozorňovacích vyskakovacích hlášek o průběhu hry.
- **Třídu `Sachovnice`:** která by reprezentovala samostatný šachový model – tím by došlo k výraznému zjednodušení a zpřehlednění kódu.

Vracení tahů

Použití algoritmu **Forsyth-Edwards Notation** pro vracení tahů nebylo nejšťastnější volbou. Původně jsem se spokojil se složitostí $O(n^2)$, ale opomněl jsem, že algoritmus nedokáže zachytit konkrétní instance figurek. To komplikuje práci při návratu tahů u speciálních situací jako braní mimochodem nebo rošády, kdy je potřeba měnit hodnoty specifických atributů. Lepším řešením by bylo použití **zásobníku** (`stack`), který by byl propojen s třídou `StavFigurky` a do něhož by se ukládaly jednotlivé provedené tahy včetně instancí figurek. Vracení tahů by pak bylo mnohem jednodušší, protože by bylo možné přímo manipulovat s konkrétními figurkami.

Možná rozšíření

AI protivník

Implementace protivníka. Pravděpodobně s využitím algoritmu **MiniMax s ořezáváním** a vyhodnocováním stavů hry na základě hodnoty figurek a celkové pozice směřující k šachu nebo matu.

Statistiky a grafy ke hře a figurkám

Pro tuto funkci by bylo vhodné zavést třídu, která by analyzovala průběh hry. Pravděpodobně by využívala zásobník (pro přístup k instancím figurek) a algoritmus Forsyth-Edwards Notation (pro uchovávání stavů šachovnice). Na konci hry by pak poskytovala statistiky o jednotlivých figurkách a průběhu celé partie.

Přívětivější grafické rozhraní

Zahrnovalo by například interaktivní menu. K tomu by byla vhodná samostatná třída, která by se starala o vykreslování jednotlivých panelů a komponent a zároveň by validovala uživatelské vstupy.

Možná další rozšíření:

- Animace pohybu figurek
- Časovač pro měření času hráčů
- Možnost ukládání a načítání rozehraných her
- atd.

Zavěr

Na závěr bych dodal, že program má místa, která by šla rozhodně ještě vyladit - viz. `Sachovnice` a Vracení tahů. Problém je určitě také v základním algoritmu. Například vykreslení šachovnice s figurkami trvá příliš dlouho. Proto na určitých místech dochází k nepříjemnému trhání obrazu, zejména, když uživatel mění rozměry okna. Kód mi přijde také na několika místech zbytečně zdlouhavý a složitý, což zhoršuje jeho přehlednost. To mě přivádí k myšlence, že pro příště bych měl věnovat více času promyšlení a návrhu projektu.