Popis problému

Úlohou je vytvoření šachového pole s vyskládanými figurkami dle pravidel šachu. Celé plátno je variabilní ke změně velikosti okna. Šachovnice navíc vždy zabírá maximální velikost okna při zachování jejího čtvercového vzhledu. S figurkami je možné libovolně hýbat. Pokud se dvě figurky nachází na stejném poli, starší figurka bude odstraněna.

V programu bude napříště možné hýbat figurkami, a to pouze dle pravidel šachu.

Implementace

Program je napsán jazykem Java s využití knihoven Java2D pro grafickou část a MouseListeneru pro rozpohybování figurek se vstupem z myši.

Program obsahuje celkem 11 tříd. Je dodrženo základní rozdělení ze cvičení na třídu BasicDrawing a DrawingPanel. K tomu je přidána třída Figurka, která je společným rodičem pro třídy Pesak, Kral, Kralovna, Vez, Strelec a Kun, každá z těchto tříd obsahuje svoji grafickou implementaci a potřebné atributy. Program se tak snaží dodržovat základní pravidla OOP.

Šachovnice je tvořena z 64 instancí třídy Policko. Program se spouští ze třídy Chess_SP_2023, jejíž jedinou funkcionalitou je volání třídy BasicDrawing, která vytvoří plátno a spustí program. Srdce programu se nachází ve třídě DrawingPanel, která se stará o korektní vykreslení všech komponent.

Každá figurka má přiřazenou referenci na políčko, na kterém se aktuálně nachází, reference na políčko je využita ke správnému překreslování dle toho, jak je zvětšováno okno a s tím i herní pole. Pomocí této reference pak jsme schopni i určit, jestli se dvě figurky nachází na stejném políčku, tedy jestli dvě figurky odkazují na stejné políčko.

Všechny metody pro pohyb figurek jsou uloženy ve třídě DrawingPanel. Metoda paint je rozdělena na dvě části, kdy jedna část je vykonána pouze v prvním spuštění – vytvoření instancí tříd, druhá část, která překresluje šachovnici s figurkami, je pak vykonána při každém volání metody.

Spuštění

Program je spouštěn pomocí sciptu Run.cmd, jehož obsahem je: `java -cp .\bin Chess_SP_2023 %* .`

Příprava k další implementaci

Od třídy Policko a reference figurky na její příslušnou instanci si slibuji především jednodušší implementaci šachových pravidel.

Rozšíření o druhou část

Změny v implementaci

Struktura kódu byla dostatečně vhodně vytvořena, a proto nebylo třeba žádnou její část mazat ani ji přepisovat.

Byla přidána třída Graf, která využívá knihovnu *JFreeChart.* Tato třída se stará o vykreslování grafu. Taktéž byla využita knihovna *JFreeSVG* pro tvorbu a export SVG.

Do kódu bylo přidáno značný množství metod starajících se o validaci tahů dle pravidel šachu. Tah, který by byl v rozporu s těmito pravidly, nebude proveden, to znamená, že nebude spuštěna animace takového tahu. Hráč je o validních tazích informován již při uchycení figurky, kdy se mu na šachovnici zeleně vybarví validní políčka. Každá třída reprezentující jeden druh figurek tak má vlastní implementaci metody *zobrazValidni()*. Oranžovou barvou jsou vybarveny počáteční a cílové políčka předcházejícího tahu protihráče. Modře se pak vybarví počáteční políčko, pokud zamýšlený tah nebude proveden, třeba z důvodu nevalidního tahu či změnu úmyslu

hráče. Nápověda nefunguje správně, pokud je hráčův král v šachu. Množina skutečně validních tahů je vždy menší než množina validních tahů, která je hráči zobrazována.

Animaci obstarává především metoda *animace()* v třídě DrawingPanel. Při provádění drag&drop hráč táhne figurkou na požadované políčko. Při tahu je se pozice figurky mění spolu s pozicí myši. Pokud je tento tah validní, figurka se okamžitě vrátí do výchozí pozice a bude spuštěna animace do požadovaného políčka. Jelikož tah uživatele nemusí vždy přesně směřovat doprostřed políčka, bude figurka vždy po skončení animace vycentrována na středu.

Mnoho metod pak obstarává komplexnější šachová pravidla. Braní mimochodem a proměna je výsostná záležitost pěšáků, kde je taky implementace z větší části umístěná. Tyto metody pak jsou volány z metody *vycentruj(),* která během centrování figurky na střed kontroluje i další možné události, které je třeba zároveň vykonat.

Proměna pěšáka je možná v dámu, koně, věž i střelce. Provádí to metoda *promena()* ve třídě DrawingPanel.

Rošáda a patová situace jsou šachové situace, která vyžadují kontrolu většího množství parametrů. Proto je jejich implementace v kódu značně rozptýlená. Jejich části však najdeme v metodách revizeSach(), *rosada()* a *pat(), mat() a* vlastni_sach() ve třídě DrawingPanel.

O výsledku hry je uživatel informován pomocí dialogového okna, kde má možnost hru restartovat.

Aplikace obsahuje i menu, s nabídkou rastrového a vektorového exportu. Je zde také možnost zobrazit graf.