**Dokumentace projektu JavaPaint – Tým 3 Čermák, Jindra, Kapsa**

**Úvod**

JavaPaint je jednoduchá aplikace vytvořená v programovacím jazyce Java s pomocí knihovny JavaFX a aplikace SceneBuilder. Aplikace slouží k jednoduchým úpravám obrázků a tvorbě nových. Momentálně obsahuje funkce jako Load – nahrání obrázku (podporované formáty: .png, .jpg, .jpeg, .bmp), Save – uložení obrázku, malování s výběrem barvy, Invert filtr, černo-bílý filtr, krok zpět a dopředu, historie použitých nástrojů a náhodná generace obrázku.

Na jejím vývoji se podíleli členi týmu The Crackheads – Marek Čermák, Tadeáš Jindra a Adam Kapsa.

Rozdělení rolí bylo následující:

* Tadeáš Jindra – Project manager, Lead function programmer
* Adam Kapsa – GUI designer, Lead GUI programmer
* Marek Čermák – Function programmer, Tester

Rozdělení práce:

* Tadeáš Jindra – Load, Save, Malování, Krok zpět a vpřed, Resizing
* Adam Kapsa – GUI, přiřazení funkcí k GUI, About pop-up
* Marek Čermák – Invert a černo-bílý filtr, náhodná generace obrázku, historie akcí

Celá práce je uložena na GitHub: <https://github.com/TadeJin/JavaPaint>

**Technické parametry**

Aplikace byla vytvořena použitím Javy 21.0.4 2024-07-16 LTS a knihovny JavaFX verze 21.0.4. Zkompilování do .jar souboru bylo realizováno pomocí nástroje pro sestavení Java aplikací Maven. GUI bylo vytvořeno v aplikaci SceneBuilder a následně implementováno do kódu pomocí .fxml souboru.

**Realizace jednotlivých funkcí**

**Load**

Funkce Load načte obrázek vybraný uživatelem na plátno, kde jej lze dále upravit. Obrázek je nejdříve načten z uživatelského výběru a poté vykreslen na plátno. Implementace je realizována následovně:

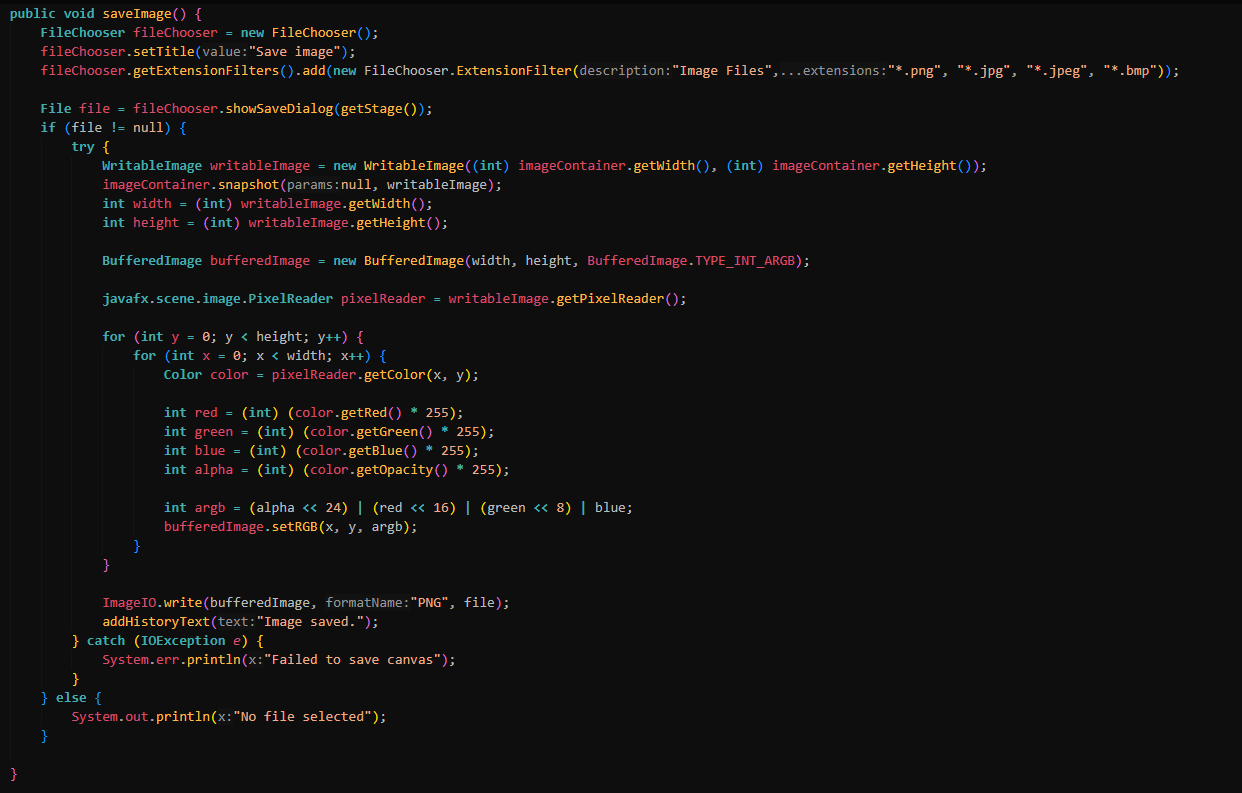
Obsah obrázku text, snímek obrazovky, Písmo

Popis byl vytvořen automaticky

**Save**

Uložení obrázku probíhá tak, že uživatel si nejdříve vybere, kam se má výsledný obrázek uložit a jak se bude jmenovat a poté bude provedeno uložení. Programově je tohle řešeno pomocí proměnné s datovým typem BufferedImage, do které se uloží jednotlivé hodnoty pixelů plátna, následně jsou tyto hodnoty převedeny do souboru a uloženy na předem zvolené místo.

Implementace v kódu:



Obsah obrázku text, snímek obrazovky, software

Popis byl vytvořen automaticky**Malování**

Malování funguje na základě vykreslování čar mezi aktuální a minulou pozicí myši, když uživatel drží levé tlačítko myši a má zapnutou funkci pro malování.

Výběr barvy je realizován pomocí GUI prvku ColorPicker.

Zda je funkce zapnutá udává proměnná isDrawing.

Minulá pozice myši je uložena v proměnných previousX a previousY.

**Invert filtr a černo-bílý filtr**

Invert filtr zneguje barvy obrázku, tj. z černé se stane bílá a naopak. Funguje na principu, kdy se stejně jako u funkce Save načte obsah plátna. Následně je obsah po jednotlivých pixlech upraven, tak aby došlo k negaci tzn. od 255 odečteme hodnotu barevné složky RGB a máme znegovanou složku. Upravený obrázek je poté vykreslen na plátno.

Implementace negace:



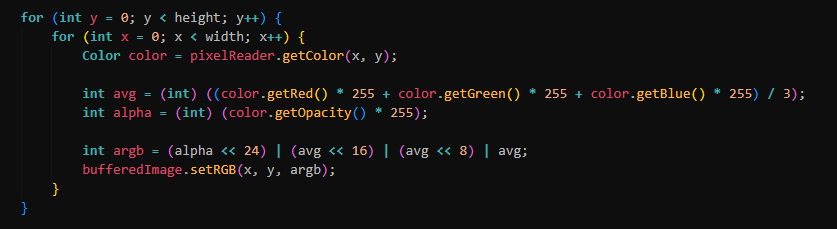
Obsah obrázku krajina, venku, obloha, strom

Popis byl vytvořen automatickyUkázka negace:

Obsah obrázku strom, snímek obrazovky, krajina, zima

Popis byl vytvořen automaticky

Černo-bílý filtr funguje na úplně stejném principu, akorát při úpravě je každá barevná složka nahrazena průměrem součtu všech barevných složek, abychom získali pouze bílou, černou a odstíny šedé. Implementace úpravy:



Obsah obrázku krajina, obloha, venku, strom

Popis byl vytvořen automatickyObsah obrázku krajina, venku, obloha, strom

Popis byl vytvořen automatickyUkázka černo-bílého filtru:

Veškeré filtry se nacházejí v menu v záložce Filters.

**Krok zpět a vpřed**

Tato funkce je klasické Ctrl + Z a Ctrl + Y, těmito tlačítky je také zavolána, ale lze jí také zavolat pomocí GUI prvků. Funguje na základě ArrayListu, který obsahuje posledních 10 změn na plátně. Po každé změně je zavolána funkce saveCanvas, která uloží obsah plátna do daného ArrayListu, pokud už ArrayList obsahuje 10 prvků nejstarší je vymazán, aby se udělalo místo pro nový. Při kroku zpět/dopředu je obsah na požadovaném indexu vykreslen na plátno.

**Historie akcí**

Na pravé straně aplikace se nachází GUI prvek TextArea, do kterého se po každé akci zapíše, co bylo provedeno. Tento prvek se roztahuje při resize. Pokud je textu hodně a nejde vidět zobrazí se slider pro posun a TextArea se dá scrollovat. Funkce pro zápis do tohoto boxu se nazývá addHistoryText a jejím parametrem je string, který chcete zapsat.

**Náhodně generovaný obrázek**

Náhodný obrázek vykreslit pomocí tlačítka „Generate random image“. Tohle tlačítko zavolá funkci generateImage, která na základě náhodně vygenerovaných hodnot a funkcí sinus a cosinus vygeneruje obrázek. Při každém zavolání se složení obrázku mění.

Implementace:

Obsah obrázku text, snímek obrazovky, software

Popis byl vytvořen automaticky

Ukázka náhodného obrázku:

Obsah obrázku Barevnost, kruh, vzor, Fraktální umění

Popis byl vytvořen automatickyObsah obrázku Barevnost, kruh, nachový, vzor

Popis byl vytvořen automaticky

**Realizace GUI**

Veškeré GUI prvky byly vytvořeny v programu SceneBuilder. V tomto programu jim byl přidělen vzhled, velikost, funkce. Následné nastavení bylo exportováno do .fxml souboru, ve kterém se nachází veškeré reference na funkce atd., a načteno programově. S GUI komunikuje třída Controller, která obsahuje veškeré funkce pro práci s programem.

Ukázka GUI aplikace:

Obsah obrázku Barevnost, kruh, snímek obrazovky, duha

Popis byl vytvořen automaticky

Obsah obrázku text, rukopis, snímek obrazovky, Písmo

Popis byl vytvořen automaticky

**About pop-up**

V menu se nachází tlačítko About, které po kliknutí zobrazí vyskakující okno s informacemi o programu. Programově je tohle implementováno jako vytvoření nové scény a zobrazení .fxml souboru pro About, a pozastavení hlavního programu do zavření tohoto okna.

Ukázka:

Obsah obrázku text, snímek obrazovky, software, Počítačová ikona

Popis byl vytvořen automaticky

**Závěr**

Cílem projektu bylo vytvořit funkční aplikaci splňující zadání a myslíme si, že se nám to povedlo. Spolupráce probíhala bez problémů a i přesto, že jsme se potkali s pár problémy, vše jsme vyřešili a práce nás bavila. V budoucnu možná projekt ještě upravíme, jelikož se jedná o poměrně užitečnou aplikaci.

**Použitá literatura**

Java dokumentace

JavaFX dokumentace

SceneBuilder dokumentace