# Zadání

## Část 1: Pasivní vizualizace

Zadáním semestrální práce bylo vytvořit program, který zobrazí přehlednou mapu převýšení. Program musí být spustitelný z příkazové řádky s parametrem datového obrázku ve formátu PGM (P2). Mapa převýšení nezávisle na vstupním obrázku maximálně vyplní okno vizualizace o minimální velikosti 800px na 600px se zachováním poměru stran obrázku za použití bilineární interpolace obrázku.

Ve vizualizaci budou zobrazeny šipkou body maximálního a minimálního převýšení a dále bod maximálního stoupání. Šipky budou stejně dlouhé a opatřené popisky. Ani šipka ani popisek ani jejich část se nesmí za žádných okolností vykreslit mimo zobrazenou mapu.

## Část 2: Interaktivní vizualizace

Základní funkční požadavky:

- Program musí splňovat všechny funkční požadavky z první části zadání
- Doplňte vizualizaci o vrstevnice. Vrstevnice budou vykresleny po každých 50 metrech převýšení (např. 350, 400, 450 atd.), po kliknutí do mapy se zobrazí převýšení v daném bodě a tak´e se zvýrazní vrstevnice, která je nejblíže svojí hodnotou nadmořské výšce zvolen´e pozice na mapě.
- Bude zvolena adekvátní barevná škála pro vykreslen´ı nadmořské výšky.
- Součástí odevzdání bude minimální dokumentace popisující základní myšlenky vedoucí k řešení problému.

Další požadavky:

- Pro celou oblast zobrazte ve vizualizaci
    - 1. Minimální a maximální převýšení, průměrné převýšení, medián a kvartily převýšení.
    - 2. Histogram převýšení
- Barevnou škálu ze základních funkčních požadavků doplňte o přehlednou legendu.
- Přiložte plnohodnotnou dokumentaci v předepsaném formátu.

# Implementace

Program jsem vytvořil v programovacím jazyce Java. Program se skládá ze tří tříd Main, DrawingPlanel a DataLoader.

## Main

Main je hlavní spustitelná třída, která se stará o spuštění programu. V Main se nejprve otestuje, jestli byl zadán vstupní parametr. Pokud žádný vstupní parametr zadán nebyl, tak se nastaví defaultní scénář plzen. Třída Main dále má na starost vytvoření funkčního okna.

### addWindowComponents

Metoda vytvoří jednotlivé komponenty a poté je přidá do okna. Do středu okna přidá plátno, doprava legendu a dolu ovládání programu. Pro ovládání si metoda vytvoří nový JPanel, do kterého si přidá jednotlivá tlačítka.

### move

Metoda si vytvoří nový okno s tlačítky pro posun, zvětšení a zmenšení obrazu. Stejně jako u metody addWindowComponents tato metoda si vytvoří okno a do ní vloží JPanel s tlačítky.

### printPNG

Exportujte data o převýšení do formátu PNG o velikosti zadané uživatelem pomocí dialogového okna. Dialogové okno se vytvoří pomocí JOptionPane a do ní se pak vloží JPanel s komponenty. Po vložení velikosti a stisknutí

tlačítka OK se přetypuje vstup na int pomocí Integer.parseInt. Pokud se přetypování nepovede, tak se vypíše chybová hláška. V opačném případě se vytvoří PNG obrázek o dané velikosti. Výsledný obrázek je upraven tak aby se vešel do okna, a přitom zachoval poměr stran.

### printSVG

Exportujte vrstevnice, šipky a další vektorovou grafiku do formátu SVG. Pro implementaci se požila knihovna SVGGraphics2D, která se stará o výpis SVG komponent. Velikost SVG obrázku je nastavená na 1920x1080. Dále se dá měnit velikost obrázku pomocí velikosti okna.

### print

Umožňuje tisk mapy na tiskárně připojené k počítači. Výtisk se implementoval pomocí PrinterJob. Mapa je upravena tak, aby se vešla na A4. Pro změnu velikosti se musí nastavit paperW a paperH v metodě print.

### showBox a showHis

Pro celou oblast zobrazí histogram převýšení, minimální a maximální převýšení, průměrné převýšení, medián a kvartily převýšení. Pro zobrazení se použila knihovna jfreechart. V obou případech se nejdřív vytvoří okno, do kterého se následně přidá samotný graf.

### DrawingPanel

Třída má na starost vykreslení mapy. Ve třídě se nastaví bilineární interpolace pomocí RenderingHints. Poté se zavolá metoda scale, která vypočte hodnotu pro škálování obrázku. Mapa je vykreslena tak, že se vždy vejde do okna a zároveň zachovává poměr stran obrázku. Dále se zavolá metoda processImage, která ze vstupních dat vytvoří obrázek. Barva obrázku se dá změnit pomocí `(g << 16) | (g << 8) | g).` Obrázek je pak uložen jako BufferedImage typu TYPE_3BYTE_BGR. V metodě drawImage se nastaví pozadí na černou barvu a poté se vykreslí obrázek na plátno. Pozice a velikost obrázku se dá změnit v g2.drawImage(). Po vykreslení obrázku se spočítá body maximálního, minimálního převýšení a bod maximálního stoupání. Metoda findMinVertical() a findMaxVertical() fungují na stejném principu a to tak, že v datech najdou nejvyšší nebo nejnižší hondnotu a uloží si jejich x a y souřadnice. Pro výpočet stoupání jsem použil vzorec:

$$m = \frac{\Delta y}{\Delta x}$$

Pro sousední body je Δx = 1 a $\sqrt{2}$~ 1,4 u diagonálních bodů. Po spočtení jedlových bodů se zavolá metoda setUpArrow, která si vypočte počátek šipky. Každá šipka ukazuje od středu obrázku na vypočtený bod. Velikost šipky je možné změnit v arrowLength. Poté co se vykreslí šipka, tak se vypíše popisek dané šipky. Popisek je vždy vycentrován za šipkou.

### processContourLine a drawConLine

Vypočítá výšku všech vrstevnic. Dále pro každou výšku vykreslí vrstevnici.

### createASCII

Umožní uložení mapy do textového ASCII art formátu. Metoda si nejprve vytvoří HashMap pro každou barvu z palety. Poté každý barvě uloží znak. Nakonec metoda projde každý pixel obrázku a pro každý pixel zjistí jeho znak z HashMap a vypíše ho do textového souboru. Vytvořený ASCII art je nakonci stejně velký jako vstupní soubor.

### zvětšení a posun

Umožní uživateli pohyb ve vizualizaci. Uživatel má možnost si mapu posunout a přiblížit/oddálit a taky mapu resetovat. Pohyb se vyřešil pomocí translate a přiblížení/oddálení pomocí scale. Jednotlivé metody mění hodnoty scale nebo translate. !!! Po posunutí obrazu nefunguje kliknutí do obrazu pro zvýraznění vrstevnice. !!!

### DataLoader

Třída DataLoader má za úkol načíst data ze vstupního souboru. DataLoader si nejprve načte vstupní soubor (java.io.File) a poté si čte jednotlivé řádky pomocí Scanner. Každá řádka se testuje, jestli nezačíná # nebo není prázdná. Pokud obsahuje tyto znaky, tak se řádka jednoduše přeskočí a testuje se další řádka. Jednotlivé řádky jsou pak dále rozděleni na jednotlivá data. Data jsou pak rozřazeny pomocí if else a nakonec jsou uloženy do int nebo do pole int. Pokud třída DataLoader nenajde vstupní soubor, vypíše do konzole chybovou hlášku a program se ukončí.

### Chart

Třída Chart má za úkol vytvořit grafy pro vizualizaci. Metoda createHistogram vytvoří histogram a createTukeyBox vytvoří tukey box. Obě metody využívají knihovny jfreechart pro vytvoření grafu. Obě metody taky fungují podobným způsobem. Nejdřív si z dvoudimenzionální pole data udělá jednodimenzionální pole. Poté si data uloží do datasetu a nakonec se pomocí ChartFactory vytvoří graf.

### Legend

Třída legend má za úkol vytvořit legendu pro vizualizaci. Legenda je umístěna na pravé části okna. Třída využívá metody drawLegend pro vytvoření legendy. Výška legendy se nijak netestuje tudíž se může stát, že mapa bude obsahovat příliš mnoho barev a to způsobí, že se legenda nevejde do okna. Dále se zavolá metoda getBlending, která vytvoří pole tak, že první barvě přiřadí 0 a poslední barvě (počet barev – 1) a podle počtu barev rozdělí tuto úsečku. Z vytvořeného pole se pak vypočte výsledná barva pomocí lineární interpolace.

### ColorPalette

Třída má za úkol vytvořit barevnou paletu pro mapu. Třída má pole barev, kde jsou uloženy barvy pro lineární interpolaci. Metoda createPalette je hlavní metoda pro vytvoření palety. Metoda nejprve volá metodu getNumOfColors, která si vypočte kolik barev je potřeba pro legendu.

## Uživatelská příručka

Program se nejprve musí přeložit pomocí Build.cmd. Poté stačí spustit program pomocí Run.cmd s příslušným parametrem. Pokud nebude vložen žádný parametr spustí se se defaultní scénář plzen (Run.cmd plzen.pgm). V souboru bin/data jsou všechny scénáře, který program může spustit. Po spuštění programu se uživateli zobrazí okno o velikosti 800px na 600px. Dole jsou tlačítka pro ovládání vizualizace. Pro ukončení programu stačí okno zavřít.

## Závěr

Program funguje, ale k ideálnímu fungování je ještě daleko. Pro usnadnění jsem neimplementoval centrování obrázku, proto se obrázek vždy vykresluje vlevo nahoře. Metoda na vykreslení šipky a jeho popisku by se dala taky zlepšit, protože pokud šipka ukazuje doprava nebo doleva, tak se může šipka překrýt s popiskem. Legenda a grafy by se rozhodně taky daly vylepšit. Co se týče splnění zadání, tak si myslím že jsem splnil všechny základní funkční požadavky..