# Uvod do semestrální práce:

Během první částí semestrální práce měli jsme udělat program, který bude správně načítat data z PGM souboru, a kreslit mapu vytvořenou z těchto dat. Problemem je to, že v PGM souboru muže být zakódován obrázek libovolné velíkostí ale podle zadání musíme vykreslit mapu s minimální velikostí 800:600 pixelů, se zachovánim poměru stran. Na mapě třeba najít body maximálního a minimálního převýšení a bod maximálního stoupání, vykreslit šipky ukazující na tyto body, každá šipka musí mít podpis. Ani podpis ani šipka v žádném případě nesmí vykreslovat mimo obrázku. Snažil jsem se vyřešit všechny tyto problémy v kódu své semestrální práce.

Během druhe častí semestrální práce měli jsme prodloužít práce s Mapou z první častí. Základními požadavky bylo rozdělít celou mapu na oblastí podle vyšky, a zabarvít každou oblast do jiné barvy. Mezi všemi oblastí měli jsme nakreslit vrstevnice. Při zmačknutí myší na jakoukoliv oblasť, musí zvyrazňovat nejbližší vrstevnice, a na souřadnicích zmačknutí má vypisovat výška daného bodu. V dalších požadavcích bylo vykreslení přehledné legendy, a generace dvou grafů: Histogram převýšení a komplikovaný graf obsahující maximální výšku, minimální výšku, median, a dolní i horní kvartily. Grafy vytvořil jsem pomocí knihovny JfreeChart. Snažil jsem se vyřešit všechny tyto problémy v kódu druhé částí své semestrální práce.

# Hlavní třída: Mapa\_SP2021

Ve třídě Mapa\_SP2021 se nachází metoda main. V metodě „main“ se vytvaruje „okno“ – instance třídy JFrame. Okno tvoří a spravuje okno. Taky v metodě „main“ se vytvaruje „panel“ – instance třídy DrawingPanel. Kterou dále přidáme do našeho okna. Taky v mainu tvoříme tlačítka “Histogram” a “Graf”, které odpovídají za vyvolání metodů, ktere vytvarují okna pro potřebne grafý podle ukolu, a vytvaruje te samotné grafy. Pří zmačknutí jedného ze tlačítek voláme odpovidajicí metodu (histogramGraf() nebo prvniGraf()) Ve třídě DrawingPanel budeme kreslit mapu a šipky.

nacteniPGM(args[0], panel) - Při volání metody nacteniPGM parametrem jmenoSouboru bude args[0], který přijímáme jako parametr v mainu, při sputění programu z příkazové řadky.

okno.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); - Umožňuje zavření okna pomocí kříže.

okno.setLocationRelative(null); - Nastavuje pozice okna na obrazovce.

okno.setVisible(true); - Tvoří viditelné okno.

Metoda „nacteniPGM(String jmenoSouboru, DrawingPanel panel)“. Tato metoda přijímá jmenoSouoru „PGM“ ze kterého bude provádět nactení dat, a instance třidy DrawingPanel – panel, aby pomocí setterů předat BufferedReader, a pole dat do třidy DrawingPanel. Metoda slouží k tomu, aby správně načíst data ze formátu PGM. To znamená, že musí odparsovat komentáře, které jsou označené „#“, prvním řádkem po komentářů je napsáno rozlišení  souboru, třetím řádkem je napsána maximální hodnota celého souboru. Ale problémem je to, že maximální hodnota barvy se rovná 255, to jest, když maximální hodnota ve souboru je více než 255, musíme převest všichní hodnoty ták, aby maximém byl 255. Take v metodě nacteniPGM naplňujeme polé “data[]”, hodnotami ze souboru PGM. Metoda nacteniPGM vráti BufferedImage vytvořeny z dat načtených ze souboru. Ještě v metodě jsou vyvolává spousta setterů ze třídy DrawingPanel settery nastavují hodnoty, které potřebujeme nastavit pouze jednou během práce programu.

Metoda histogramGraf() vytvaruje okno pro histogram výšek na mapě. Vytvoření toho okna je velmi podobně vytvoření hlavního okna, ale odstranil jsem  volání metody setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE), protože nepotřebujeme zastavovat celý program při zavření tohoto okna. Taky v teto metodě vytvarujeme panel - instance třídy ChartPanel z knihovny JfreeChart, do které vkládáme vystup metody histogram(). Výsledkem práce cele metody histogramGraf() bude otevřené okno s histogramem převýšení.

Metoda “oknoPrvniGraf()” dělá skoro to samé, co i histogramGraf(), ale v okně vykresluje 1. graf ze zadání, jeho vlastností podrobněji rozepsány v popisu metody prvniGraf().

Metoda “histogram(int[] data)” přijímá pole všech výšek mapy, a pak vrátí instance třídy JFreeChart, konkrétně, která v okně vykresluje jako histogram. V grafu je maximálně 15 sloupcu. Pomocí instance třidy XYPlot - plotu, nastavujeme bilou barvu pozadí. Ten celý histogram ukazuje kolik bodů výšky z nějakym krokem máme.

JFreeChart “prvniGraf()” Vytvaruje 1 graf ze zadání. Tento graf musí obsahovat Maximální a minimální hodnoty mapy, Median převýšení a dolní i horní kvartily. Na začátku vytvaruje DefaultCategoryDataset - dataset do kterého se vkládají všichni této hodnoty. Median, i horní a dolní kvartily vypočítám v oddělených metodach. Pak vytvorujeme JFreeChart - chart, barchart na kterem budou zobrazovat všechna data. Potom pomocí CategoryItemRenderer - render měníme některé nastavení toho grafu.

Metoda “medianPrevyseni(int[] data)” přijímá pole dat načtených ze souboru, a vypočítá median.

Metoda “dolniKvartil(int[] data)” přijímá pole dat načtených ze souboru, a počítá dolní kvartil, to je hodnota, která je větší než 25% hodnot na mapě.

Metoda “horniKvartil(int[] data)” přijímá pole dat načtených ze souboru, a počítá horní kvartil, to je hodnota, která je větší než 75% hodnot na mapě.

Metoda “sort(int[] arr)” přijímá pole dat načtených ze souboru, a vrací tříděné pole dat.

# Třida DrawingPanel

Třída DrawingPanel dědí třídu „Jpanel“.

Ve třídě DrawingPanel mimo žádné metody jsou vytvořené několik proměnných.

BufferedImage bg\_img - nacteniPgm(„název souboru“) – Načtený obrázek metodou nacteniPGM

BufferdImage img – Zpracovány obrázek metodou processImage

BufferdImage imageVrstevnice – Obrázek který bude použit dale pro vykreslení vrstevnic

int[] data – Pole prvků ze souboru PGM převedených do formátu (0 - 255)

int[] nezpracovanaData – Pole prvků ze souboru PGM

int minHodnota; - Minimální hodnota výšky ze souboru PGM, nastavená pomocí setteru

int pocetBarev - Počet barev, potřebných pro vykreslení všech výšek, nastavuje pomoci setteru

double krokVysky - Krok ze kterym barvíme výšky, nastavuje pomocí setteru

Color[] poleBarev - pole do kterého vkládáme vygenerované barvy, potřebné pro barvení obrázku

int startX - souřadnice na které se nachází X začátku obrazku

int startY - souřadnice na které se nachází Y začátku obrazku

int iW - širka obrázku

int iH - vyška obrázku

int zmacknutaVyska - vyška zmačknutého bodu

int thisWidth – širka okna

int thisHeight – výška okna

int vyskaLegendy - vyška legendý mápy

int indexMinimalniBarvy - index minimální barvy, kterou používáme při vykreslení

double scale - koeficient škálování okna

List<VrstenviceSour>[] poleSouradnicVrst - pole listů, do kterých budeme vkládat, souřadnice vrstevnic podle indexů, do každého listu vkládají souřadnice jedné vrstevnice

Všichni tyto proměnné budou použité dál v ruznych metodach. Do proměnné bg\_img pomocí metody bg\_img přidáváme načteny ze souboru BufferedImage.

Konstruktor třídy „DrawingPanel“. V konstruktoru pomoci „this.setPreferedSize()“ nastavuje rozlišení obrázku, který bude vykreslen v okně. Taky nastavujeme proměnné thisWidth a thisHeight. Ještě do konstruktoru přidal jsem metodu this.addMouseListener(), která zpracovává zmáčknutí a posuv myší. Konkrétně v přepsané metodě mousePressed(MouseEvent e), zpracováváme zmáčknutí do mapy. v této metodě pomoci getX() a getY() dostáváme souřadnice zmáčknutého bodu, pak odstraňujeme škálování, a odčítáme startX a startY. Takhle přišli jsme k bodu obrázku. Pomocí metod getVyska a indexBarvyVysky zjišťujeme výšku v daném bodu mapy, a index souřadnice která musí zvýraznit při zmáčknutí pak voláme metodu barveniVrstevnice() aby obarvit tuto vrstevnice, na konci v pole otevreneVrst[] označujeme vrstevnice z daným indexem jako true(otevřené), aby při otevření příští vrstevnice věděli jsme jeli otevřená nějaká jiná.

Metoda „paint(Graphics g)“ je přepsána ze třídy JPanel. A slouží k tomu, aby kreslit všechno co potřebujeme na „JPanelu“. „g“ – instance třídy „Graphics“ slouží k jednoduchemu kreslení. Uvnitř metody paint vytvořil jsem „g2“ – instance třídy „Graphics2D“ od  „g“.  „Graphics2D“ je skoro stejné jako „Graphics“, ale novější, a má více možností. Take v metodě paint vyvolávám metodu „drawPlzenImage(g2, this.getWidth(), this.getHeight())“. Do které vkládáme komponent „g2“, aktuální výšku a šířku.

Metoda “barveniVrstevnice (int indexVrstevnice, Color color)” přijímá index vrstevnice, a barvu do které musíme tuto vrstevnice zabarvit. Prochází ArrayList správného indexu z pole poleSouradnicVrst[indexVrstevnice] a mění barvu objektů za správnou.

Metoda drawLegendaMapy(Graphics2D g2, int W, int H, int startX, int startY, double scale) přijímá Graphics2D pro kreslení na panelu, výšku a šířku obrázku, počáteční body kreslení obrázku, a koeficient škálování. drawLegendaMapy slouží k tomu, aby správně kreslit legendu mapy pod samotným obrázkem. Na začátku metoda ověřuje není li minimální hodnota menší než index minimální barvy násobenou 50, dělám to proto že v případě když je menší, tak první barva legendy bude větší než minimální barva na mapě, a musíme ten index zmenšit o jeden. Výška vypočíta nasobenim indexu min. barvy o 50. Protože krokém výšky je 50. Dal uvnitř cyklu procházíme všichni pole barev od minimálního indexu do konce, a vykreslujeme pod obrázkem obdélníky dané barvy, z podpisem. V případě když nadpis nebo obrázek přesahuje velikost okna, zvětšujeme výšku legendy, a přenosíme tuto barvu o řádek dolu. V případě když barev je více než 30, aby mohli jsme tyto barvy korektně vykreslit, zvětšujeme krok výšky tak, aby počet barev nepřesahoval číslo 10.

Metoda „drawPlzenImage(Graphics2D g2, int W, int H)“ slouží k tomu, aby kreslit hlavní obrázek mapy, správně ho škálovat, to znamená, že když měníme velikost okna take mění se i velikost obrázku se zachováním poměru stran. Pod obrázkem je vykreslen černý obdélník, i když poměr stran okna a obrazku se liší, tak nevidíme bílé pruhy, ale vidíme ten obdélník. Aby obrázek se kreslí ve středu okna, zavádíme proměnné: „startX a startY“. Obrázek se nebude kreslit od počátku souřadnicového systému, ale od „startX, startY“. Dale v metodě zavádíme dvourozměrné pole: „pixels[][]”. Do kterého vkládáme správně provedené pomocí metody „arrayToDoubleArray()“ jednorozměrné pole „data[]”. Pole: „pixels[][]”. Teď v „pixels[][]” mamé seznam všech pixelu našeho obrázku.  Dale budeme používat k nalezení souřadnic pro všechny požadované body.  Dalším krokem tvořím instance třídy „souradniceXY“ pro bodý maximálního stoupání, maximálního a minimálního převýšení a vkladám do nich udaje vynalezené pomocí speciální metody, o kterych budeme mluvit dale. Taky v metodě „drawPlzenImage()“ zapínám a provádím bilineární interpolace obrazku a antialiasing fontů pomocí instance třídy „RenderingHints“ z knihovny Java. Mápu kreslíme pomocí standardní metody ze „Graphics2D“ – „drawImage()“. Po vykreslení obrazku, ale před voláním metod vykreslujicích šipky zapínáme antialiasing podobným způsobem jako interpolace, zatím interpolace se vypíná, teď zaplňujeme pozadí “imageVrstevnice” velikostí iW / iH černou barvou, a pomocí metody “makeColorTransparent” děláme toto pozadí průhledným, zatím převádím “Image” do “BufferedImage”. Potom voláním metody “kresleniVrstevnic” vykreslujeme vrstevnice na obrázku “imageVrstevnice”. Pak vykreslujeme tento obrázek do okna. Udělal jsem to aby neinterpolovat ty vrstevnice.

Metoda “imageToBufferedImage(Image image)” převádí Image do BufferedImage.

Metoda “makeColorTransparent(BufferedImage im, final Color color)” přijímá BufferedImage a Color, ktery třeba udělat průhlednou.

 Metoda  „processImage()“ Tato metoda pomocí metody „getRGB“ načítá data z „BufferedImage bg\_img“ do pole „pixels[][]“, problém je v tom, že data jsou připravené k vykreslení barevného obrazku, ale hodnoty z PGM jsou černobíle. Proto musíme ta data připravit k zobrazení v černobílém obrázku. Metoda Prochází v cyklu pole „pixels[][]“, a zpracovává data tak, aby bylo vhodné k vytvoření černobílého obrázku. Potom tvoří Color s rgb hodnotou kterou jsme dostali, a z toho koloru beré libovolnou ze třech barev (všichni jsou stejné), a v souvislostí na výšku tohoto bodu, barvíme ho podle indexu barvy zjištěnému z pomocí metody indexBarvy pixelu z pole nezpracovaných dat nacházejícího ve stejném místě. Pak, pomocí metody setRGB vkladáme pole do proměnné BufferedImage image, kterou vytvořili mimo metod na začátku třídy.

Metoda “getBarvy(int pocetBarev)” přijímá počet barev, které se potřebují pro vykreslení obrazku, a tvoří pole těchto barev. Barvy prochází v cyklu, první barva je maximálně zelená, a minimálně červená, každý krok cyklu o krokBarvy (se rovná 255/pocetBarev) zvětšuje červenou barvu, a zmenšuje zelenou. Modrá na začátku je 80, a mění se v souvislostí na zelenou a červenou.

Metoda “kresleniVrstevnic(BufferedImage imageVrstevnic)” přijímá obrázek, do kterého třeba nakreslit vrstevnice. Pak prochází pole Arraylistů - “poleSouradnicVrst” v těchto Listech jsou zapsané objekty třídy “VrstevniceSour” které obsahují souřadnice a barvu každého pixelu vrstevnic. Index pole se odpovídá indexu vrstevnice. Metoda vratí “imageVrstevnic” s nakreslenyma vrstevnicamá.

Metoda prvniZaplneniSour() na začátku metoda “arrayToDoubleArray” převádí pole “nezpracovanaData” do dvourozměrného pole, prochází ho a hledá pixely vrstevnic. Metoda bere index barvy výšky každého pixelu, a jeho sousedů, pak jich porovnáva, a v případě když hlavní pixel větší než jakýkoliv ze sousedů, tak to je vrstevnice nalezený pixel zapisuje do správného indexů v poleSouradnicVrst[ ].

Metoda indexBarvyVysky(int vyska) přijímá výšku, a spočítá pro ní index barvy, dělením výšky o krok (50), pak zaokrouhlíme to nahoru.

Metoda „maxSloupani(int pixels [][])“ přijíma pole „pixels [][]“, prochází toto pole vnořeným cyklém for a porovnává hodnotu každého pixelu s hodnotami čtyř vedlejších pixelů. Dolo tady k problému toho, že píxely na okrají obrazku nemají souseda na jedné ze stran, kvůlí tomu dochazí k chybě „IdexOfBoundsException“. Aby tento problém vyřešil jsem napsáním bloku „try – catch“. Metoda se vrací instance třidy maxSloupaniXY, do které vkladá souřadnicé x, y bodu maximalního sloupaní na mapě.

Metoda „maxPrevyseni (int pixels [][])“ přijímá a prochází pole „pixels [][]“ moc podobně předchozí metodě, ale hledá bod maximálního převýšení mápy, to jest když hodnota aktuálního pixelu je větší než hodnota proměnné „maxVyška“ tak metoda vkládá hodnotu pixelu do proměnné, a ukládá jeho souřadnice.

Metoda „minPrevyseni(int pixels [][])“ funguje asi stejně jako maxPrevyseni, ale hleda nejmenší hodnotu.

Metoda „drawDesc(int x2, int y2, String nadpis, Graphics2D g2, double scale, int startX, int startY)“ přijímá všechno jako předchozí a ještě navíc nápis, který se má zobrazovat. Jako v předchozí metodě spočítat počáteční souřadnice šipky. Dále na základě poloh šipky, a velikosti fontu spočítám souřadnice počátečního bodu souřadnice. Pak kontrolujeme, že souřadnice dolního a horního bodu nadpisu jsou na obrázku a že čárka mapy více než délka podpisu. Když nadpis stoupá za obraz vpravo nebo vlevo, tak posouváme ten nadpis opačným směrem.

Metoda arrayToDoubleArray(int[] data) přijímá jednorozměrné pole, a převádí ho do dvourozměrného.

Metoda doubleArrayToArray(int[][] data) přijímá dvourozměrné pole, a převádí ho do jednorozměrného.

Metoda „drawArrow(int x2, int y2, Graphics2D g2, double scale, int startX, int startY)“ přijímá souřadnice koncového bodu šipky „x2 y2“, instance třídy „Graphics2D“ – „g2“, hodnotu škálování – scale a souřadnice začátku obrazku –„startX“, „startY“. Pomocí metody „vypocetSouradniceX1Y1()“ spočítám souřadnice začátku šipky. Dále spočítáme směr a délku šipky, a souřadnice hrotů šipky. Pak, v případě když bod začátku šipky se nachází na mapě, kreslíme šipku ze třech linek pomocí metody „draw“ ze třídy „g2“.

Metoda „vypocetSouradniceX1Y1(int x2, int y2, int startX, int startY, double scale)“ kontroluje, zda se koncový bod šipky nachází blízko okraje mapy, a v případě, když ano, spočítat souřadnice počátečního bodu šipky tak, aby šipka kreslila směrem k okraji obrazu. Metoda vrátí souřadnice pomocí instance třídy „souradniceX1Y1()“.

Metoda getVyska(int x, int y) přijímá souřadnice x a y, a vrací výšku, ktera se nachází na těchto souřadnicích na obrazku.

Metoda getZpracovanaVyska(int x, int y) přijímá souřadnice x a y, a vrací zpracovanou (0-255) výšku, ktera se nachází na těchto souřadnicích na obrazku.

Dal jsou gettery, ve kterych nastavují všechny potřebné hodnoty.

# Třída Souřadnice

Třída usnadňuje práci se souřadnicemi x, y. Když potřebujeme vrátit souřadnice x a y z nějaké metody můžeme udělat v metodě instance této třídy a vrátit ji.

Konstruktor „SouradniceXY(int x, int y)“ přijímá x a y, a nastavuje proměnné třídy.

Metoda „getX()“ vrací proměnnou x.

Metoda „getY()“ vratí proměnnou y.

# Třída VrstenviceSour

Tuto třídu vytvořil jsem pro chránění souřadnic a barev pixelů vrstevnice. Pomocí getterů a setterů můžeme nastavit x, y, a barvu. Na začátku barva je světle modrá, jako zmáčknuté vrstevnice

# Zavěr

Během dvou častí semestralní práce pokusíl jsem vyřešít všichní požadavký které jsme mělí v zadání. Toto je největší program který jsem dosud psál, během ho realizací naučíl jsem spoustu novych věcí.