# 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.

# 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. Ve třídě Drawing Panel budeme kreslit mapu a šipky.

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.

# 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

int[] data – Pole prvků ze souboru PGM

int windowsWidth – širka okna

int windowsHeight – výška okna

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.

Metoda „nacteniPGM(String jmenoSouboru)“. Tato metoda slouží k tomu, aby správně načíst data ze formátu PGM. To znamená, že musím 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.

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. Pak, pomocí metody setRGB vkladáme pole do proměnné BufferedImage image, kterou vytvořili mimo metod na začátku třídy.

Metoda „paint(Graphics g)“ je přepsána ze třidy JPanel. A slouži k tomu, aby kreslít všechno co potřebujeme na „JPanelu“. „g“ – instance třidý „Graphics“ slouží k jednoduchemu kreslení. Uvnítř metody paint vytvořil jsem „g2“ – instance třidy „Graphics2D“ od „g“. „Graphics2D“ je skoro stejné jako „Graphics“, ale novějsí, a má více možností. Take v metodě paint vyvolavám metodu „drawPlzenImage(g2, this.getWidth(), this.getHeight())“. Do které vkladáme komponent „g2“, aktualní vyšku a šírku.

Metoda „drawPlzenImage(Graphics2D g2, int W, int H)“ slouží k tomu, aby kreslít hlávní obrázek mápy, spravně ho škalovat, to znamená, že když měníme velíkost okna take mění se i velíkost obrazku se zachováním poměru strán. Pod obrázkem je vykreslen černý obdélník, i kdýž poměr strán okna a obrazku se liší, tak nevidíme bílé pruhy, ale vidíme ten obdelník. Aby obrázek se křeslíl ve stedu okna, zavádíme proměnné: „startX a startY“. Obrázek se nebudé kreslít od začátku souřadnicovýho systému, ale od „startX, startY“. Dale v metodě zavádimé dvourozměrné pole: „pixels[][]”. Do kterého vkladáme spravně peřevedené pomocí metody „arrayToDoubleArray()“ jednorozměrné pole „data[]”. Pole: „pixels[][]”. Teď v „pixels[][]” mamé seznam vešech pixelu našécho obrázku. Dale budemé použivát k nalezení souřadnic pro všechny požadované body. Dalším krokém tvořím instance třídy „souradniceXY“ pro bodý maximalního sloupaní, maximalního a minimalního převyšení a vkladám do nich udaje vynalezené pomocí speciální metody, o ktérych budeme mluvit dale. Taky v metodě „drawPlzenImage()“ zapínam a provádím bilinearni interpolace obrazku a antialiasing fontů pomocí instance třidy „RenderingHints“ z knihovny Java. Mápu kreslíme pomocí standartní metody ze „Graphics2D“ – „drawImage()“. Po vykresléní obrazku, ale před voláním metod vykreslujicích šipky nastvujemé hodnoty velíkostí okna s obrázkem do proměnných „windowsWidth“ a „windowsHeight“, které byly vytvořeny mimo metod na začátku třidý. Dále volamé všichní třidy vykreslujicí šipky a podpisy k ním.

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 „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 „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 „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()“.

# 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.