# Dokumentace UPG – semestrální práce

Cíl této práce bylo vytvořit aplikaci, která dokáže vizualizovat data poskytnuté knihovnou. Použitým jazykem je Java a grafické frameworky jsou Java Swing a Java AWT, používané knihovny jsou JFreeChart and JFreeSVG.

**Zpracování**

Práce se zpracovává ve dvou etapách a následné bonusové úlohy.

První etapa (Pasivní vizualizace)

* cílem této etapy je vykreslovat síť a vizualizovat její simulaci

Druhá etapa (interaktivní vizualizace)

* cílem této etapy je krom vykreslování vizualizace vodovodní sítě také vizualizace podrobností o průběhu simulace, jako např. grafy zaplnění nádrží nebo průtoku v potrubí, poté také možnost nastavit otevřenost uzávěrů.

**Použití**

**Kompilace aplikace**

Aplikaci zkompilujeme pomocí cmd scriptu Make.cmd, který připraví aplikaci na zpuštění.

**Zpuštění aplikace**

Zpuštění programu provedeme pomocí cmd souboru Run.cmd, který zpustí aplikaci s defaultními parametry.

Zpuštění programu s volitelným parametrem <Glyph size> pomocí příkazu

java -cp water-network Main

Příkaz zpustíme ze složky bin

java -cp water-network Main 150

Tímto příkazem zpustíme aplikaci s vlastním parametrem

**Řešení**

Během zpracovávání vizualizace jsem narazil na mnoho problémů a několik složitějších situací, která mi dali pořádně zabrat. Jedním z nich bylo napojení na knihovnu, i když později se vše zdálo jednoduché, ze začátku bylo obtížnější pochopit jak se na knihovnu napojit, ale během hodinové analýzy jsem většinu pochopil.

**Vykreslování**

Vykreslování je založené na „atomickém“ přístupu tj. každá komponenta je vykreslována vlastním objektem s názvem Draw<název komponenty>, tento přístup umožnuje rychlé rozšíření v případě potřeby, a opravu chyby většinou bez potřeby přepisovat návazné kódy, taktéž to zabrání takzvanému špagetovému kódu.

**Souřadnice reálné a virtuální**

Další velkou překážkou bylo zpracovávání souřadnicového systému, původní přístup byl, že každé vykreslování si bude samo počítat, jak a kde se má vykreslit, ovšem toho se ukázalo jako nevhodné vzhledem k tomu že jsem musel používat redundantní kód, což se neslužuje se zásadou don’t repeat yourself (DRY). Také se tento přístup ukázal, jako nevhodný vzhledem k „atomovému“ modelu. Nakonec jsem zvolil použití Singletonu Translater a jak už název napovídá, tato třída překládá virtuální souřadnice dodané knihovnou na reálné, který využívám pro vykreslování

**Řešení druhé části**

**Opravy chyb z první části**

První věc než jsem začal pracovat na nových funkcionalitách, byla oprava všech chyb které jsem nalezl během procházení kódu, především pak předávání GlyphSize, která byla v kódu z testování nastavena na pevno, dále pak chyba při které se spoje potrubí (nódy) špatně umísťovaly pokud byla hodnota GlyphSize nastavena na nestandardně velkou nebo malou hodnotu. Poté přišel velký refactoring, na špatný návrh a rozložení tříd mě upozornil můj cvičící na UPG Libor Váša a po krátkém vysvětlení kde byl problém jsem měl jasno a předělal systém kreslení, kde jsem sloučil všechny třídy do jedné ucelené třídy DrawNetworkComponents, která obsahuje logiku kreslení s příslušnými třídami.

**Grafy**

Když jsem přemýšlel nad grafy, byli k dispozici dvě možnosti, použít knihovnu nebo si napsat vlastní grafy. Napsáním vlastního grafy bych měl absolutní kontrolu nad kódem a přesně bych věděl, kdyby něco nefungovalo, ovšem z důvodu časového presu a kvalitě dostupných knihoven jsem se rozhodl pro první možnost a to využít možnost, zdě opět vyvstali dvě možnosti a do knihovna JFreeChart a Apache Batik. Po přečtení dokumentace obou knihoven jsem se rozhodl pro JFreeChart jak pro její jednoduchost tak protože je stále ve vývoji.

**Detekce kliknutí**

Detekce kliknutí sama o sobě není problém, ale v případě detekování kliknutí na prvky sítě se jednalo o nelehký úkol. Jako první jsem musel vyřešit jak detekovat kliknutí, pro to jsem zvolil metodu .contains(), která je k dispozici u objektů Ellipse2D a Rectangle 2D, které jsem předával bod z eventu kliknutí .getPoint(). Další otázka byla odkud brát instanci objektu pro porovnání, když jsme tento problém konzultovali s kolegou, přemýšleli jsme nad přímím předáváním, což se nezdálo příliš použitelné nebo uložení v nějaké datové struktuře, s touto myšlen jsem dělal několik pokusů jako například pseudo-strom slovníků nebo jeden ArrayList pro všechny předávané objekty, ale nakonec jsem vytvořil sadu hashMap, kde klíčem je skutečný objekt tj. prvek v síti, a jako hodnota byl objekt vytvořený na základě souřadnic a velikosti vykreslovaných prvků sítě. S tímto návrhem jsem postavil celou detekční proceduru. Třídu uchovávající tyto hashMapy jsem pojmenoval ComponentCatalog.

**Dodatečné funkcionality**

Jako bonusovou funkcionalitu jsem přidal možnost exportu do SVG a PNG, plán byl takový že přidám tuto možnost i pro grafy ale ty samotné už uměli tuto funkcionalitu.