Univerzita Pardubice

Fakulta elektrotechniky a informatiky

# Kontrola výpočtu plochy křivek naměřených programem Temporary Display

Milan Kacálek

Bakalářské práce

2011

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2010/2011

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

| | |
|---|---|
| Jméno a příjmení: | **Milan KACÁLEK** |
| Osobní číslo: | **I08072** |
| Studijní program: | **B2646 Informační technologie** |
| Studijní obor: | **Informační technologie** |
| Název tématu: | **Kontrola výpočtu plochy křivek nameřených programem Temporary Display** |
| Zadávající katedra: | **Katedra informačních technologií** |

Zásady pro vypracování:

Program Temporary Display načítá průběhy změn teplot krve při vstřiku injektátu do příslušné komory srdce a průběhy všech načtených křivek pacienta ukládá do binárního souboru. Program počítá, zda křivka návratu teploty do původního stavu je křivkou zdravého nebo nemocného pacienta. Kritickou veličinou výpočtu je plocha křivky.
Program studenta pro bakalářskou práci načte křivky z binárního souboru a zobrazí kontrolní vypočtené plochy a srovná s plochami vypočtenými programem Temporary display.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

**Příručka programu Temporary display.**

Vedoucí bakalářské práce: **Ing. Jaroslav Štroch**
Katedra informačních technologií

Datum zadání bakalářské práce: **17. prosince 2010**

Termín odevzdání bakalářské práce: **13. května 2011**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2011

# Prohlášení

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 17. 1. 2009

Milan Kacálek

# Poděkování

Rád bych touto cestou poděkoval panu Ing. Jaroslavovi Štrochovy za ochotnou a trpělivou spolupráci při tvorbě této práce.

# ANOTACE

Tato práce je zaměřena na vývoj aplikace, který zobrazuje průběhy změn teploty krve při vstřiku injektátu do příslušné komory srdce. Dále dopočítává plochu křivky. Z těchto údajů lze určit stav pacientova srdce. Veškeré průběhy a další data jsou uložena v binárním souboru.

# KLÍČOVÁ SLOVA

java, temporary display,

# TITLE

Checking calculation of the area curves measured program Temporary Display

# ANNOTATION

This work is focused on application development, which shows the pattern of change inblood temperature injektátu injection into the chambers of the heart. Furthermore,calculates the area of the curve. From these data can determine the status of the patient's heart. All courses and other data are stored in a binary file.

# KEYWORDS

java, temporary display,

# OBSAH

# Seznam zkratek

| | |
|---|---|
| TP | program Temporary Display. |
| IDE | Integrated Development Environment (integrované vývojové prostředí). |
| POJO | Plain Old Java Object. |
| XML | Extensible Markup Language (rozšiřitelný značkovací jazyk). |
| SQL | Structured Query Language (strukturovaný dotazovací jazyk) |
| JDBC | Java Database Connectivity |
| API | Application Programming Interface |
| BST | Binární soubory dat pro Temporary Display. |
| CO | minutový objem (cardiac output) |
| GUI | Graphical User Interface (grafické uživatelské rozhraní) |

# Seznam obrázků



# Seznam tabulek

# 1 Úvod

Tématem mé bakalářské práce je „Kontrola výpočtu plochy křivek naměřených programem Temporary Display“. Program Temporary Display měří postupné změny teploty krve při vstříknutí studeného injektátu do jedné ze srdečních komor. Touto metodou lze odhalit jisté srdeční vady, například tzv. zkrat neboli proražení srdeční stěny mezi jednotlivými srdečnými stěnami. Program Temporary Display tyto data ukládá do binárního souboru. Tento binární soubor bude zpracovávat mnou vytvořený program.

Vlastní práce je tedy zaměřena na tvorbu aplikace, která bude zobrazovat naměřené hodnoty v přehledném grafu a dopočítávat další potřebné hodnoty k určení pacientovy srdeční vady. Aplikace je implementována prostřednictvím programovacího jazyka Java.

Práce je rozděluje na teoretickou část a na část praktickou. V teoretické části budou postupně představeny technologie a metody potřebné pro vytvoření programu (použité IDE, metody výpočtu integrálu) a v teoretické části bude popsán program samotný spolu s postupem vytváření a některými problémy, které jsem byl nucen při vývoji aplikace řešit.

# 2 NetBeans IDE

NetBeans začal jako studentský projekt (původně nazván Xelfi) v České republice v roce 1996. Xelfi byl první Java IDE napsaný v jazyce Java. Dnes je NetBeans úspěšný Open Source projekt s velmi rozsáhlou uživatelskou základnou, rostoucí komunitou vývojářů a téměř 100 partnery po celém světě. Firma Sun Microsystems založila Open Source projekt NetBeans v červnu 2000 a do roku 2010 byl hlavním sponzorem celého projektu, kdy se společnost Sun Microsystems stala dceřinou společností Oracle. Vývojové prostředí NetBeans IDE je nástroj, pomocí kterého programátoři mohou psát, překládat, ladit a distribuovat aplikace. Vývojové prostředí podporuje prakticky jakýkoliv programovací jazyk. Existuje rovněž velké množství modulů, které toto vývojové prostředí rozšiřují. Vývojové prostředí NetBeans je bezplatně šířený produkt a jeho užívání není nijak omezeno. NetBeans IDE je vyvíjeno pod licencí Open Source a je možné je bezplatné používat v komerčním i nekomerčním prostředí. Zdrojový kód je dostupný pod licencí Common Development and Distribution License (CDDL) v1.0 a GNU General Public License (GPL) v2. (1) (2)

# 3 Java

Java je objektově orientovaný programovací jazyk, který vyvinula firma Sun Microsystems a představila 23. května 1995. Podle TIOBE je JAVA nejpopulárnější programovací jazyk. (3) Java je přenositelný programovací jazyk. Díky tomu se používá na různých systémech počínaje čipovými kartami, přes desktopové počítače až po rozsáhlé distribuované systémy. Existuje několik teorii o původu názvu JAVA, podle jedné z nich byla pojmenovaná podle druhu kávy. (4)

Java vychází z jazyka C++, ke kterému má syntakticky nejblíže. Oproti svému předchůdci Java neobsahuje některé konstrukce, které způsobovaly při programování největší potíže, a navíc přidává mnoho užitečných vlastnosti, nad kterými zajásal nejeden programátor. Zde je pár příkladů:

- Přidělování a uvolňování paměti je zde obstaráno automaticky (pomocí garbage collectoru). Objekt se neruší pomoci delete nebo free(), ale pouze se "nabídne" ke zrušení (např. přiřazením neplatné reference null).
- Klasický problém z C/C++, ukazatele, je zde zcela odstraněn, neboť ty zde prostě nejsou (resp. jsou nahrazeny referencemi). Dereferencovaní samozřejmě provádí runtime. Programátor je zde ušetřen notorické chyby zápisu pointerem mimo datovou oblast.
- Je implementován mechanismus vláken (threads) a lze tudíž spouštět více úloh v rámci jednoho programu. Bylo samozřejmě pamatováno i na jejich synchronizaci pomoci tzv. monitorů.
- Je implementován mechanismus výjimek, takže veškeré runtime chyby je možné odchytit a zpracovat. Výjimky jsou samozřejmě objektové, takže lze zachytit i celou hierarchii výjimek v jednom hlídaném bloku.
- Lze provádět reflexi, neboli zjišťování informací o objektu (jaké má proměnné, metody atd.).
- Značným ulehčením pro programátory je obsáhlost standardně dodávaných knihoven, se kterou se nemůže srovnávat asi žádný běžně používaný jazyk. K dispozici jsou knihovny pro tvorbu grafického uživatelského rozhraní (GUI), vstup/výstup, práci s textem, komunikaci s SQL databázemi, práci s komprimovanými soubory a mnoho dalších!
- Java klade značný důraz na bezpečnost. Díky tomu, že je překládaná do bytecodu, a implementovaným bezpečnostním mechanismům (podepisování kódu a přidělování práv pro různé akce), lze zajistit, že program v Javě, který si uživatel stáhne ze sítě, mu nezformátuje disk, nebude komunikovat s žádným jiným počítačem, než ze kterého pochází atd. (5)

## 3.1 Rozdělení

### 3.1.1 Java SE (Standard Edition)

Java Platform, Standard Edition umožňuje vývoj a nasazení Java aplikací na desktopy a servery, stejně jako na dnešní náročné vestavěných zařízení a real-time zařízení. (6)

### 3.1.2 Java EE (Enterprise Edition)

Java Platform, Enterprise Edition (Java EE[1]) je průmyslový standard pro podnikové využívání Java. Vývojáři mají prospěch ze zvyšování produktivity s více anotace, více objektů POJO, zjednodušené balení a méně XML konfigurace. (7)

### 3.1.3 Java ME (Micro Edition)

Java Platform, Micro Edition (Java ME), poskytuje robustní a flexibilní prostředí pro běh aplikací na mobilních telefonech a dalších vestavěných zařízeních: mobilní telefony, osobní digitální asistenty (PDA), TV set-top boxy, a tiskárny. Java ME zahrnuje flexibilní uživatelské rozhraní, robustní zabezpečení, vestavěný síťové protokoly, a podpora pro sítě a offline aplikace, které lze dynamicky stahovat. Aplikace založených na Java ME jsou přenosné přes mnoho zařízení. (8)

### 3.1.4 Java FX

V návaznosti na platformě Java, JavaFX poskytuje přesvědčivou kombinaci všudypřítomnosti, schopnosti, expresivita a výkonu. JavaFX SDK má základní sadu technologií, nástrojů a zdrojů potřebných pro vývojáře a designéry k vytvoření a nasazení výrazných a silných programů přes prohlížeč, desktop, mobil, televize a dalších připojených zařízení. (9)

### 3.1.5 Java DB

Java DB je Sun podporována distribuce open source Apache Derby 100% databázových technologií Java. Je plně transakční, bezpečná, snadno použitelná, dodržující normy SQL, JDBC API, a Java EE. (10)

[1] *Název platformy Java for Business byl zjednodušen. Dříve byla známá jako Java 2 Platform, Enterprise Edition (J2EE).*

### 3.1.6 Java Card Technology

Technologie Java Card poskytuje bezpečné prostředí pro aplikace, které běží na čipových kartách a dalších zařízeních s velmi omezenou pamětí a možnosti zpracovávání. Více aplikací může být nasazen na jedinou kartu, a nové aplikace mohou být přidány i později. Aplikace napsané v programovacím jazyce Java může být bezpečně nasazen na karty od různých výrobců. (11)

## 3.2 Java Virtual Machine (JVM)

JVM je sada počítačových programů a datových struktur, která využívá modul virtuálního stroje ke spuštění dalších počítačových programů a skriptů vytvořených v jazyce Java. Úkolem tohoto modulu je zpracovat pouze tzv. bytecode. Právě díky tomu že je JVM k dispozici na mnoha platformách, stačí program vytvořit pouze jednou a spustit na kterékoli z platforem, pro které bylo vyvinuto JVM. JVM umožňuje automatické zpracování výjimek, díky kterým dokážete určit příčinu chyby nezávisle na zdrojovém kódu. (12)

# 4 JFreeChart

JFreeChart je 100% zdarma Java graf knihovna, která usnadňuje vývojářům zobrazit profesionální a kvalitní grafy ve svých aplikacích. Mezi jeho rozsáhlé vlastnosti patří:

- konzistentní a dobře zdokumentované API,
- podpora celé řady grafů,
- flexibilní design, který se dá snadno rozšířit,
- podpora mnoha typů výstupu, včetně Swing komponent, obrázků (včetně PNG a JPEG) a vektorové grafiky (včetně PDF, EPS a SVG),
- JFreeChart je "open source", nebo, přesněji, svobodný software. Je distribuován pod GNU Lesser General Public Licence (LGPL), který umožňuje použití v proprietárních aplikacích. (13)

## 4.1 Historie

Projekt JFreeChart byla založena před devíti lety, v únoru 2000, Davidem Gilbertem. Dnes je používán přibližně 40.000 až 50.000 vývojáře. Projekt pokračuje pod vedením pana Gilberta, s přispěním pestré komunity vývojářů. (13)

## 4.2 Kde knihovnu získat

Knihovnu si můžete stáhnout na stránkách http://www.jfree.org zde si můžete i nastudovat dokumentaci, nebo se podívat na ukázkové programy. Ve staženém archívu je nejdůležitější složka lib, ve které najdete mnoho souboru s příponou .jar, z nichž jsou nejdůležitější tyto soubory - jcommon-1.0.16 a jfreechart-1.0.13, které se musí naimportovat do vyvojářského prostředí např. Netbeans nebo Eclipse.

# 5 Numerická integrace

Velké množství určitých integrálů nelze vyjádřit prostřednictvím elementárních funkcí. V takovém případě používáme k určení hodnoty integrálu přibližných metod, mezi něž patří tzv. numerická integrace. Při numerické integraci se snažíme nahradit integrál jiným druhem výpočtu, přičemž se snažíme zajistit, aby se získaná hodnota od skutečné hodnoty integrálu lišila co nejméně. Numerické metody výpočtu integrálu jsou velmi vhodné pro použití ve výpočetní technice, neboť umožňují vytvoření relativně jednoduchých algoritmů pro určování hodnot určitých integrálů.

Nejjednodušší metody numerické integrace jsou Riemannovi součty, lichoběžníková metoda. (14)

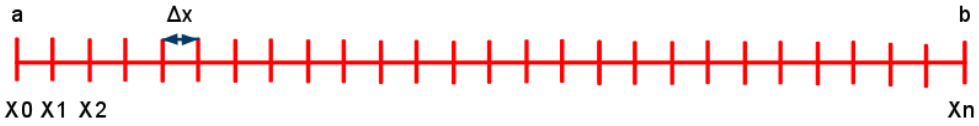
## 5.1 Riemannův integrál

Historickou motivací pro vznik určitého integrálu byl výpočet obsahů ploch. Tento problém řešili již staří Egypťané v souvislosti s určováním velikostí pozemků, jejichž velikost se měnila v důsledku záplav Nilu. Problém řešili tak, že danou plochu rozdělili na trojúhelníky, spočítali jejich obsahy a ty pak sečetli. Tyto metody později rozvinuli staří Řekové. Pojmenován byl po německém matematikovi Bernhardu Riemannovi. (15)

Začneme tak že interval [a, b] integrálu $\int_a^b f(x)dx$ rozdělíme na n stejných částí.

$\Delta x = (b - a)/n$

$x_k = a + k * \Delta x$

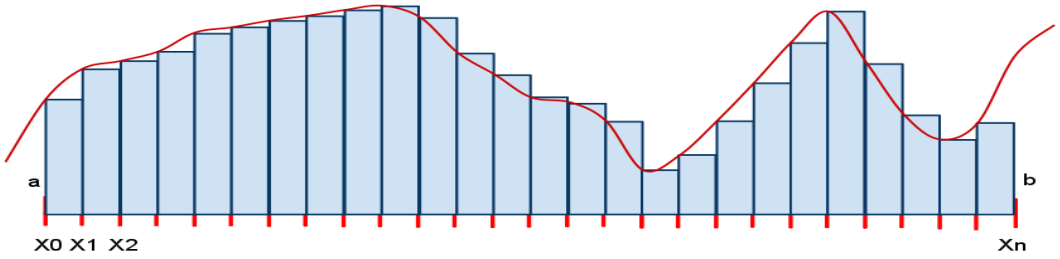



*Obrázek 1 – osa intervalu integrálu.*

### 5.1.1 Levý Riemannuv součet

Při této metodě se obsah obdélníku spočítá jako hodnota funkce v bodě $x_k$ vynásobená hodnotou $\Delta x$. Hodnota $k$ začíná na hodnotě 0. Poté se sečtou obsahy všech takto vzniklých obdélníků. Všimněte si, že levá strana obdélníku odpovídá hodnotě grafu. Proto název levý Riemannuv součet.

$$\int_a^b f(x)dx = \sum_{k=0}^{n-1} f(x_k) * \Delta x$$

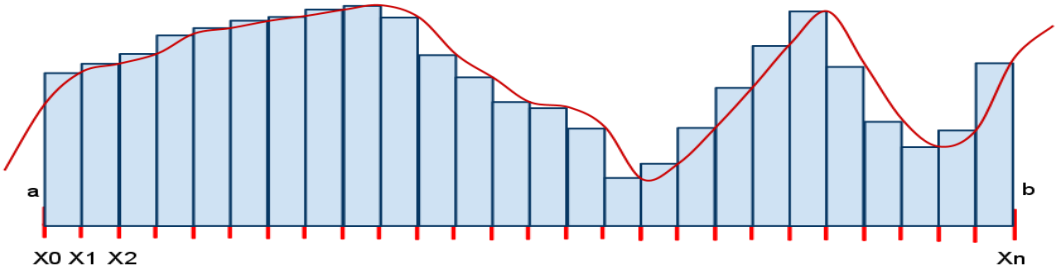



*Obrázek 2 – Levý Riemanuv součet*

### 5.1.2 Pravý Riemannuv součet

Při této metodě se obsah obdélníku spočítá jako hodnota funkce v bodě $x_k$ vynásobená hodnotou $\Delta x$. Hodnota $k$ začíná na hodnotě 1. Poté se sečtou obsahy všech takto vzniklých obdélníků. Všimněte si, že pravá strana obdélníku odpovídá hodnotě grafu. Proto název pravý Riemannuv součet.

$$\int_a^b f(x)dx = \sum_{k=1}^{n} f(x_k) * \Delta x$$




*Obrázek 3 – Pravý Riemannuv součet*

## 5.2 Lichoběžníkový metoda

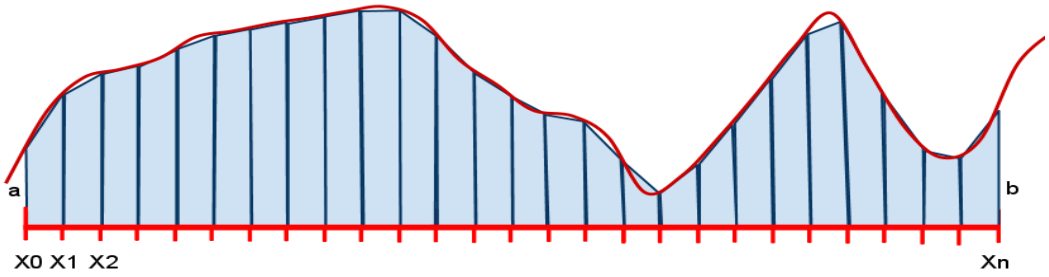
Při této metodě se používají lichoběžníky. Opět se spočítá obsah jednotlivých lichoběžníků a poté se obsahy sečtou. Opět začneme tak že interval [a, b] integrálu $\int_a^b f(x)dx$ rozdělíme na n stejných částí.

$$\Delta x = (b - a)/n$$

$$x_k = a + k * \Delta x$$

Počet opakování se spočítá jako n vydělené celočíselně dvěma plus 1.

$$\int_a^b f(x)dx = \sum_{k=0}^{(n\ div\ 2)+1} \frac{1}{2}\left(f(x_k) + f(x_{k+1})\right) * \Delta x$$




*Obrázek 4 – Lichoběžníková metoda*

# 6 Formát zápisu BST

BST je binární soubor vytvořený programem napsaným v programovacím jazyce C++. Proto jsem musel udělat jisté úpravy tabulky pro použití v jazyce JAVA.

## 6.1 Základní data s určením počtu měření (křivek)

Následující data se v každém měření vyskytují pouze jednou. Řetězec je zde uložen tak že nejdříve je uložena délka řetězce a poté jsou uložené jednotlivé znaky řetězce.

| Načíst typem[2] | | Délka | Poznámka | Popis |
|---|---|---|---|---|
| **int** | | 4 | | Verze BST |
| **řetězec** | **int** | 4 | Délka řetězce včetně \0 | Validace |
| | **String** | Délka | Data řetězce | |
| **int** | | 4 | | Číslo validace |
| **řetězec** | **int** | 4 | Délka řetězce včetně \0 | Číslo katetrizace |
| | **String** | Dálka | Data řetězce | |
| **řetězec** | **int** | 4 | Délka řetězce včetně \0 | Jméno operátora |
| | **String** | Dálka | Data řetězce | |
| **řetězec** | **int** | 4 | Délka řetězce včetně \0 | Příjmení operátora |
| | **String** | Dálka | Data řetězce | |
| **řetězec** | **int** | 4 | Délka řetězce včetně \0 | Jméno technika |
| | **String** | Dálka | Data řetězce | |
| **řetězec** | **int** | 4 | Délka řetězce včetně \0 | Příjmení operátora |
| | **String** | Dálka | Data řetězce | |
| **řetězec** | **int** | 4 | Délka řetězce včetně \0 | Seznam diagnóz |

[2] Datové typy jsou určené pro programovací jazyk Java

| | | | | |
|---|---|---|---|---|
| | **String** | Dálka | Data řetězce | |
| **int** | | 4 | | Počet měření |
| **řetězec** | **int** | 4 | Délka řetězce včetně \0 | Jméno pacienta |
| | **String** | Dálka | Data řetězce | |
| **řetězec** | **int** | 4 | Délka řetězce včetně \0 | Příjmení pacienta |
| | **String** | Dálka | Data řetězce | |
| **řetězec** | **int** | 4 | Délka řetězce včetně \0 | Rodné číslo |
| | **String** | Dálka | Data řetězce | |
| **float** | | 4 | | Váha pacienta |
| **float** | | 4 | | Výška pacienta |

*Tabulka 1 – Základní data měření*

## 6.2 Data jednoho měření (křivky)

Tyto data se opakují podle hodnoty počtu měření z předcházejících dat.

### 6.2.1 Základní data jednoho měření (křivky)

| Načíst typem[3] | | Délka | Poznámka | Popis |
|---|---|---|---|---|
| **int** | | 4 | | Pořadové číslo měření |
| **int** | | 4 | | UTC čas měření |
| **int** | | 4 | | nevýznamná hodnota |
| **int** | | 4 | | index teploty vstřiku injektátu |
| **řetězec** | **int** | 4 | Délka řetězce včetně \0 | Definice kódu condition |
| | **String** | Dálka | Data řetězce | |
| **řetězec** | **int** | 4 | Délka řetězce včetně \0 | Definice kódu vstřiku |

[3] Datové typy jsou určené pro programovací jazyk Java

| | | | | |
|---|---|---|---|---|
| | **String** | Dálka | Data řetězce | a detekce |
| **float** | | 4 | | Teplota krve |
| **float** | | 4 | | Teplota injektátu |
| **float** | | 4 | | Objem injektátu |
| **float** | | 4 | | Konstanta pro výpočet TDCO |
| **int** | | 4 | | indexy rozmezí dat pro určení GAMAFUNKCE (od) |
| **int** | | 4 | | indexy rozmezí dat pro určení GAMAFUNKCE (do) |
| **int** | | 4 | | Počet naměřených hodnot za sekundu |
| **int** | | 4 | | Počet naměřených dat teploty krve (NT) |
| **int** | | 4 | | Počet naměřených dat teploty injektátu (NI) |
| **int** | | 4 | | Zakódovaný stav |

*Tabulka 2 – Data jedho měření*

### 6.2.2 Naměřené teploty krve

Tyto data se opakují podle předchozí hodnoty „Počet naměřených dat teploty krve (NT)“.

| Načíst typem[4] | Délka | Poznámka | Popis |
|---|---|---|---|
| **float** | 4 | | teplota krve [°C] |

*Tabulka 3 – Naměřené teploty krve*

### 6.2.3 Naměřené teploty injektátu

Tyto data se opakují podle předchozí hodnoty „Počet naměřených dat teploty injektátu (NI)“.

| Načíst typem[4] | Délka | Poznámka | Popis |
|---|---|---|---|
| **float** | 4 | | teplota injektátu [°C] |

*Tabulka 4 – Naměřené teploty injektátu*

# 7 Výpočet CO

Cílem regulace krevního oběhu je udržet zásobení orgánů okysličenou krví adekvátní perfuzí tkání. Množství krve vypuzené srdcem do systémového oběhu za minutu vyjadřujeme jako srdeční výdej – cardiac out (CO). Jeho velikost je řízena požadavky jednotlivých orgánů a ne srdcem samým. Je jedním z velmi důležitých hemodynamických parametrů, protože bez jeho znalostí není možné správně hodnotit intrakardinální tlaky, které jsou závislé na velikosti průtoku. (16)

Následující stránky naznačují jak se CO vypočítá. Pro výpočet CO jsou potřeba 3 kroky. Nejprve se musí určit parametry pro výpočet gama funkce, dále se za pomocí parametrů určí gama funkce a nakonec se vypočítá hodnota TDCO.

[4] Datové typy jsou určené pro programovací jazyk Java

## 7.1 Výpočet parametrů exponenciály

Jako první se musí určitě parametry exponenciály.

Obecný výpočet pro exponenciálu je $y = A * t^B * e^{C*t}$ kde A, B, C jsou námi hledané koeficienty.

- A určuje amplitudu,
- B určuje rychlost nárůstu,
- C určuje rychlost sestupu.

Pro výpočet koeficientů se rovnice zlogaritmuje na $\ln y = \ln A + \mathrm{B} * \ln t + C * t$. Všechny změřené teploty musí ležet na hledané exponenciále (Obrázek 5). Ke koeficientům A, B, C se dostaneme maticovým počtem. Kde:

- y je okamžitá hodnota exponenciály,
- n je počet vzorků teploty,
- t je čas.

Matice $Y = X * \beta$

$$\begin{bmatrix} \ln y_1 \\ \ln y_2 \\ \vdots \\ \ln y_n \end{bmatrix} = \begin{bmatrix} 1 & \ln t_1 & t_1 \\ 1 & \ln t_2 & t_2 \\ \vdots & \vdots & \vdots \\ 1 & \ln t_n & t_n \end{bmatrix} * \begin{bmatrix} \ln A \\ B \\ C \end{bmatrix}$$

Cramérovo pravidlo

$$\left[\begin{array}{ccc|c} n & \sum_{i=1}^{n} \ln t_i & \sum_{i=1}^{n} t_i & \sum_{i=1}^{n} \ln y_i \\ \sum_{i=1}^{n} \ln t_i & \sum_{i=1}^{n} (\ln t_i)^2 & \sum_{i=1}^{n} \mathrm{t_i} * \ln t_i & \sum_{i=1}^{n} \ln t_i * \ln y_i \\ \sum_{i=1}^{n} t_i & \sum_{i=1}^{n} \mathrm{t_i} * \ln t_i & \sum_{i=1}^{n} {t_i}^2 & \sum_{i=1}^{n} t_i * \ln y_i \end{array}\right]$$

Pro výpočet determinantů zavedeme

$$a_{11} = n \qquad a_{33} = \sum_{i=1}^{n} t_i^2 \qquad b_1 = \sum_{i=1}^{n} \ln y_i$$

$$a_{12} = \sum_{i=1}^{n} \ln t_i = a_{21} \qquad a_{22} = \sum_{i=1}^{n} (\ln t_i)^2 \qquad b_2 = \sum_{i=1}^{n} \ln t_i * \ln y_i$$

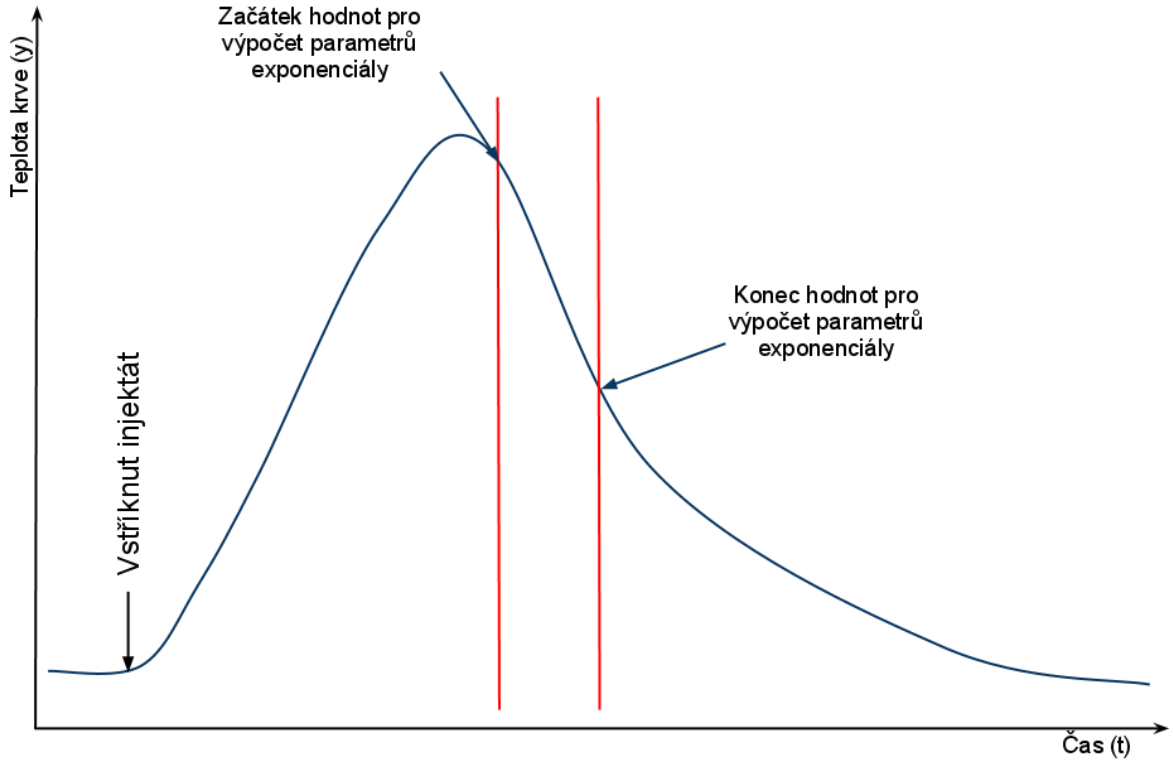
$$a_{13} = \sum_{i=1}^{n} t_i = a_{31} \qquad a_{23} = \sum_{i=1}^{n} t_i * \ln t_i = a_{32} \qquad b_3 = \sum_{i=1}^{n} t_i * \ln y_i$$

Výpočet determinantu

$$D = \begin{vmatrix} a_{11} a_{12} a_{13} \\ a_{21} a_{22} a_{23} \\ a_{31} a_{32} a_{33} \end{vmatrix} \quad D_{\ln A} = \begin{vmatrix} b_1 a_{12} a_{13} \\ b_2 a_{22} a_{23} \\ b_3 a_{32} a_{33} \end{vmatrix} \quad D_B = \begin{vmatrix} a_{11} b_1 a_{13} \\ a_{21} b_2 a_{23} \\ a_{31} b_3 a_{33} \end{vmatrix} \quad D_C = \begin{vmatrix} a_{11} a_{12} b_1 \\ a_{21} a_{22} b_2 \\ a_{31} a_{32} b_3 \end{vmatrix}$$

Výpočet koeficientů exponenciály

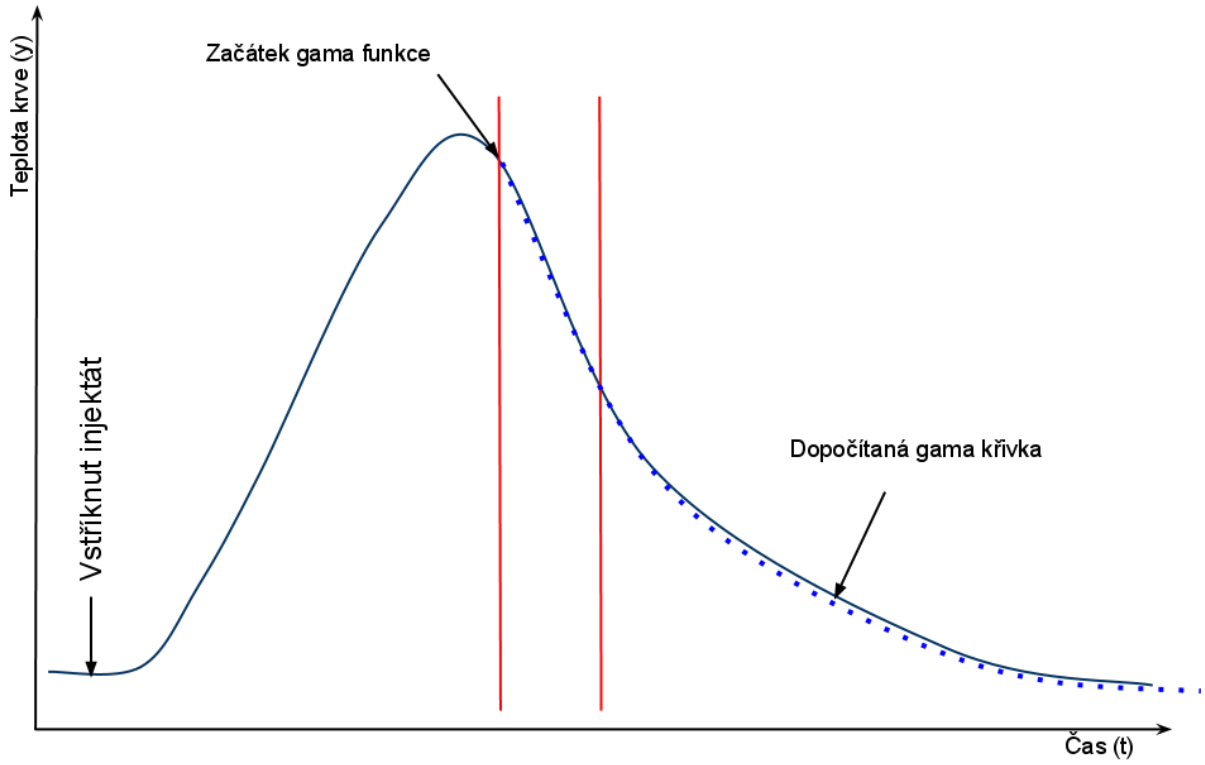
$$\ln A = \frac{D_{\ln A}}{D} \qquad B = \frac{D_B}{D} \qquad C = \frac{D_C}{D}$$




*Obrázek 5 – Výpočet parametrů exponenciály*

## 7.2 Dopočítání gama funkce

Jako další krok musíme dopočítat hodnoty gama funkce. K tomu použijeme vzorec $y = A * t^B * e^{C*t}$. Výpočet parametrů A, B, C je popsán v předchozí kapitole. Začátek gama funkce je znázorněn na obrázku 6. Gama funkce se počítá do té doby, než se dosáhne konstantních hodnot.




*Obrázek 6 – Dopočítaná gama funkce*

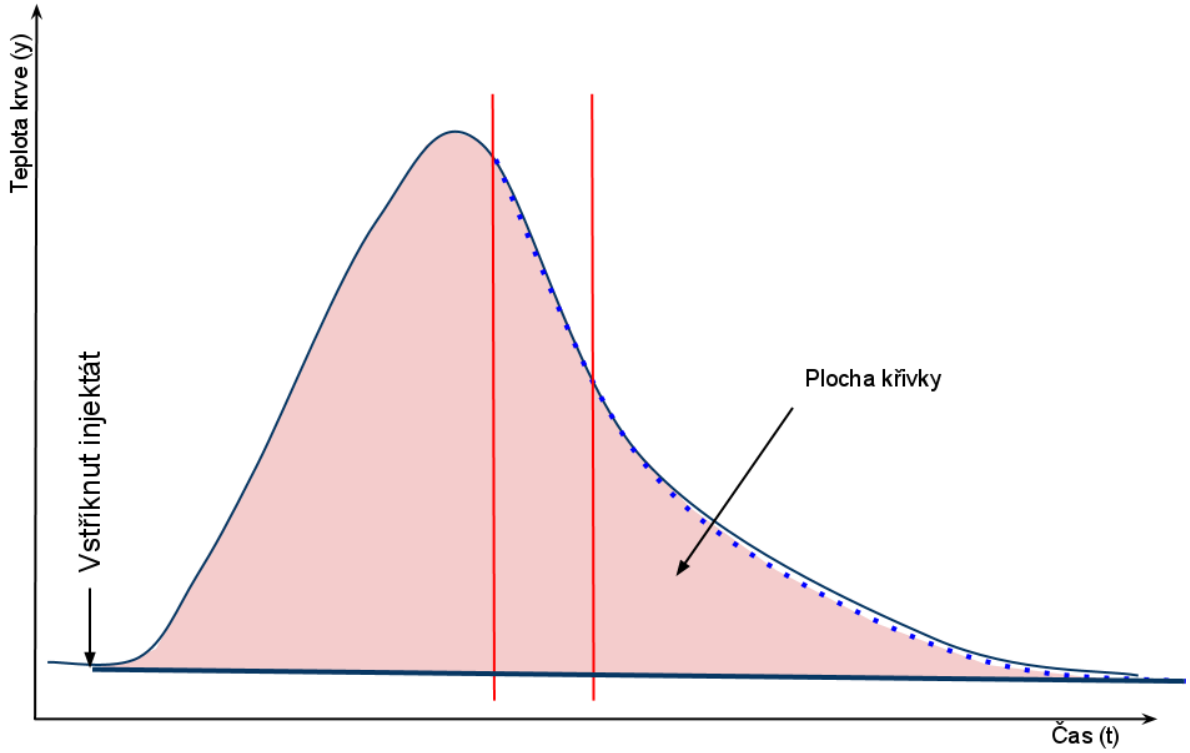
## 7.3 Konečný výpočet TDCO

Pro výpočet TDCO se používá následující vzorec,

$$TDCO = \frac{V_i * 60 * (T_k - T_i)}{S} * C$$

Kde:

- $V_i$ je objem injektátu,
- $T_k$ je teplota krve,
- $T_i$ je teplota injektátu,
- $S$ je plocha pod křivkou (integrál, obrázek 7),
- C je konstanta regulující chybu hardwaru.



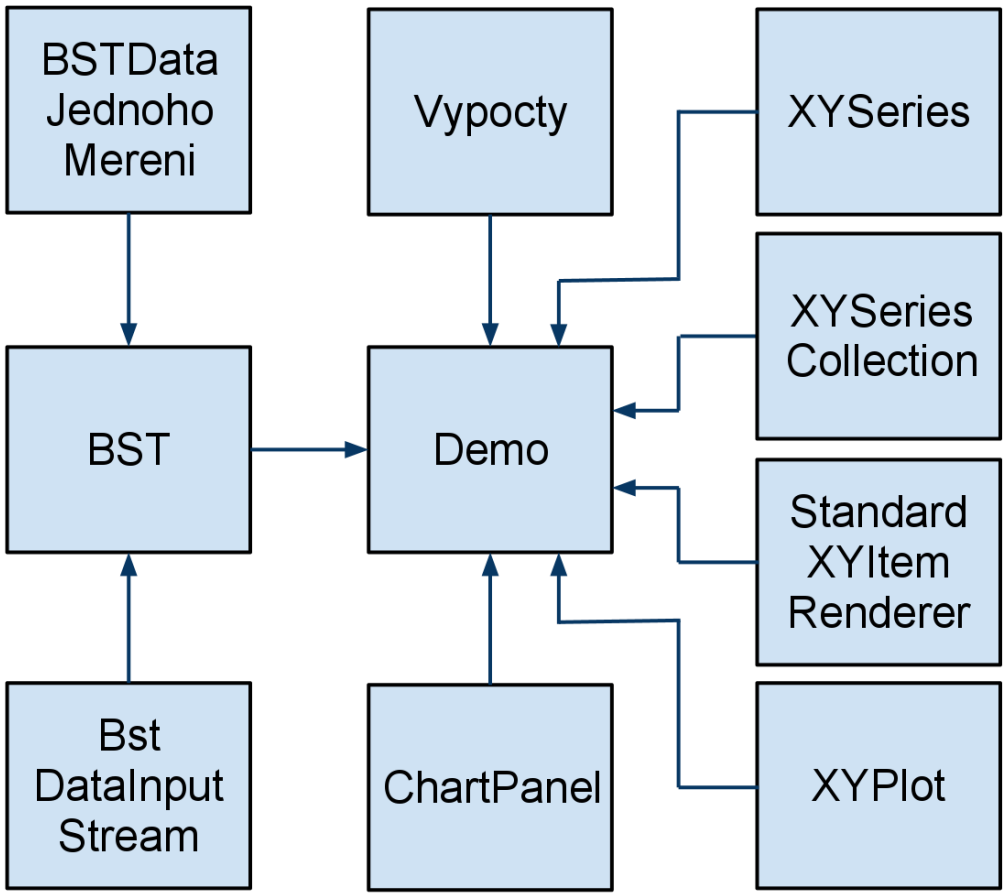

*Obrázek 7 – Integrál křivky*

# 8 Aplikace

Úkolem mojí bakalářské práce bylo vytvořit program, který načte binární soubor vytvořený programem Temporary Display. Ten následně zpracuje, zobrazí křivky v něm uložené a vypočítá jejich plochy. V této popíšu, jak tento můj program pracuje a jaké problémy spojené s tvorbou programu bylo nutné řešit.

## 8.1 Návrh aplikace

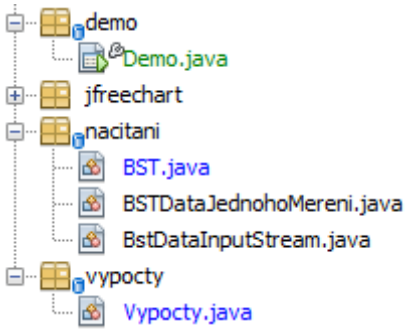
### 8.1.1 UML diagram




*Obrázek 8 – UML diagram*

Na obrázku číslo 8 je zobrazen UML diagram aplikace. Na obrázku je patrné, že většina tříd je asociována se třídou *Demo*. Třída *Demo* je zodpovědná za zobrazování GUI a obslužnost programu. *XYSeries, XYSeriesCollection, StandardXYItemRenderer, XYPlot a CharPanel* jsou třídy zajišťující zobrazení grafu a jsou z balíčku *JFreeChart*.

### 8.1.2 Použité třídy




*Obrázek 9 – Struktura použitých tříd*

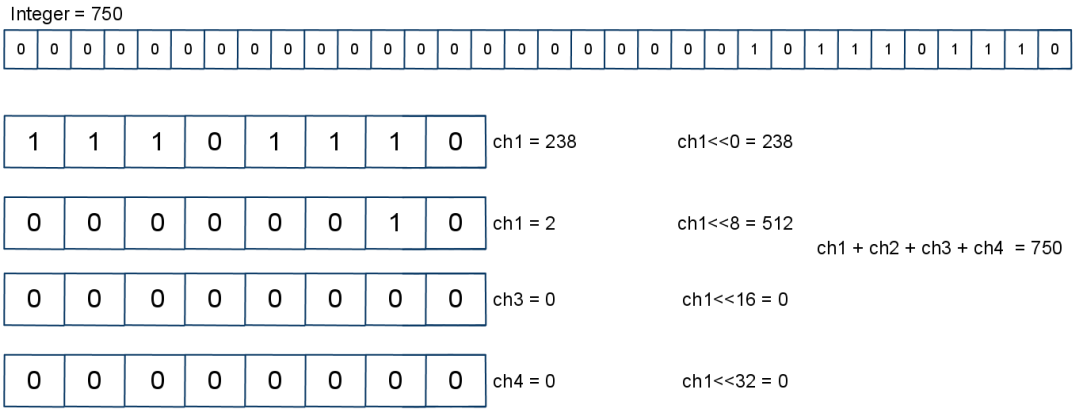
První balíček je pojmenován *demo.* Tento balíček je určen pro spustitelné třídy programu. Třída *Demo* zajišťuje veškeré uživatelské ovládání (načtení nového souboru, prohlížení grafu atd.), je potomkem třídy JFrame.

Dalším balíčkem je *jfreechart*, který obsahuje soubory *jar*. V těch jsou uložené potřebné třídy pro vykreslování grafů. Některé třídy, které byli použité v tomto programu zde popíšu. Třída *XYSeries* představuje seznam souřadnic ve formátu (x, y). Ve výchozím nastavení je seznam seřazen podle hodnoty x vzestupně a duplicita proměnné x je povolena. Třída *XYSeriesCollection* představuje kolekci objektů *XYSeries*, která jde následně použít jako datová sada. Tato třída se používá při konstrukci grafu. Třída *StandardXYItemRenderer* zajišťuje jakým způsobem se bude vykreslovat graf (barva atd.). Třída *YXPlot* je obecná třída pro vykreslení bodů podle souřadnice x, y. *XYPlot* potřebuje třídy *XYSeriesCollection* a *StandardXYItemRenderer* aby věděl, co a jak má vykreslit. A třída *ChartPanel* je potomek *JPanel* a pomocí něho se zobrazuje graf. (17)

Dalším balíček je důležitý pro načtení a udržení dat z binárního souboru. Balíček se jmenuje *nacitani* a obsahuje následující třídy. Třída *BST* zajišťuje načtení a uchování dat z binárního souboru. Pro načítání je zde použita třída *BstDataInputStream*, která dědí *FilterInputStream* a po jednotlivých bytech načítá jednotlivé data. Zde uvedu příklad načtení proměnné Integer.

```
public int readInt() throws IOException {
    int ch1 = in.read();
    int ch2 = in.read();
    int ch3 = in.read();
    int ch4 = in.read();
    return ((ch1) + (ch2 << 8) + (ch3 << 16) + (ch4 << 24));
  }
```

Protože je Integer uložen na 4 bytech je potřeba tyto 4 byty načíst, to nám zajišťuje metoda read(), která načte jeden byt a vrátí nám hodnotu načteného čísla. Dále ještě musíme takto načtené hodnoty bitově posunout o náležitý počet bitů a poté sečíst. Na obrázku 10 je znázorněno proč je potřeba provádět bitový posun.




*Obrázek 10 – Bitový posun*

Problém, který jsem musel vyřešit při načítání znaků, byl v tom že java používá pro kódování znaků Unicode kdežto znaky uložené v binárním souboru jsou kódované pomocí ASCII. S anglickými znaky problém nebyl, ten nastal až při načítání českých znaků. Tento problém jsem vyřešil tak to:

```
public char readChar() throws IOException {
    int ch1 = in.read();
    char vystup;
    if (ch1 == 138) {
       vystup = 'Š';
    } else if (ch1 == 198) {
       vystup = 'Ć';
    } else if (ch1 == 157) {
       vystup = 'ť';
    } else if …….
    } else if (ch1 == 154) {
       vystup = 'š';
    } else {
       vystup = (char) ch1;
    }
    return vystup;
  }
```

Tento kód je zkrácený. Jak jde vidět tak podle ordinální hodnoty dosazuji konkrétní české znaky. Další a poslední třídou tohoto balíčku je *BSTDataJednohoMereni*, tato třída uchovává data jednoho měření.

Dalším a posledním balíčkem je balíček jménem *vypocty* a tento balíček obsahuje třídu *Vypocty*. V této třídě jsou metody pro výpočet potřebných hodnot. Například metoda pro výpočet parametrů gama funkce, pro výpočet samotné gama funkce, pro výpočet integrálu, pro výpočet TDCO a podobně. Příklad výpočtu TDCO:

```
public double vypocetTDCO(BSTDataJednohoMereni mereni) {
    float Vi = mereni.getObjemInjektatu();
    float Tk = mereni.getTeplotaKrve();
    float Ti = mereni.getTeplotaInjektatu();
    float vysledek = Vi * 60 * (Tk - Ti);

    vysledek /= this.integral;
    vysledek *= mereni.getTDCO();
    vysledek /= 10;
    vysledek = Math.round(vysledek);
    vysledek /= 100;
    return vysledek;
  }
```

## 8.2 Použití Java Swingu

Swing je knihovna určená pro vytváření uživatelského rozhraní (GUI). Knihovna Swing obsahuje aplikační rozhraní pro tvorbu a obsluhu klasického grafického uživatelského rozhraní. Pomocí Swingu jde vytvářet okna, dialogy, tlačítka a další. Na následujících řádcích popíši některé třídy, které jsem použil při tvorbě programu.

### 8.2.1 Přizpůsobení vzhledu podle použitého operačního systému

O vzhled aplikace se stará UIManager, kterou je součástí balíčku Swing. Abych se použil vzhled podle používaného operačního systému, stačí použít tento kód.

```
try {
    UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
} catch (Exception e) {
}
```

UIManager.setLookAndFeel slouží k nastavení vzhledu aplikace a UIManager.getSystemLookAndFeelClassName vrací vzhled použitého operačního systému. (18)

### 8.2.2 JPanel

JPanel je panel na který můžeme umisťovat další komponenty. Tím můžeme rozdělovat program na více oblastí. Každý panel může mít nastaven svůj Layout.

### 8.2.3 JLabel

JLabel je zobrazovací plocha pro krátký textový řetězec, obrázek čí obojí. Štítek nereaguje na vstupní události. Jde určit, kde se na ploše štítku zobrazuje obsah. Lze zarovnávat jak horizontálně tak vertikálně. Ve výchozím nastavení je zarovnání na střed plochy. Jde také určit pozici textu vzhledem k obrázku. K nastavení textu se používá metoda setText() a k vrácení textu je metoda getText(). (19)

### 8.2.4 JSeparator

JSeparator poskytuje univerzální komponent pro vytváření dělící čáry - nejčastěji používané jako oddělovač mezi položkami menu, které tak lze řadit do logických skupin. Místo používání JSeparator, se může použít JMenu nebo JPopupMenu a metodu addSeparato, který vytvoří a přidá separátor. JSeparator se může také použít kdekoli jinde v GUI. (20)

### 8.2.5 JButton

JButton je klasické tlačítko které po stlačení provede nějakou akci. Dále lze tlačítku nastavit jakýkoliv text nebo ikonu.

### 8.2.6 JFileChooser

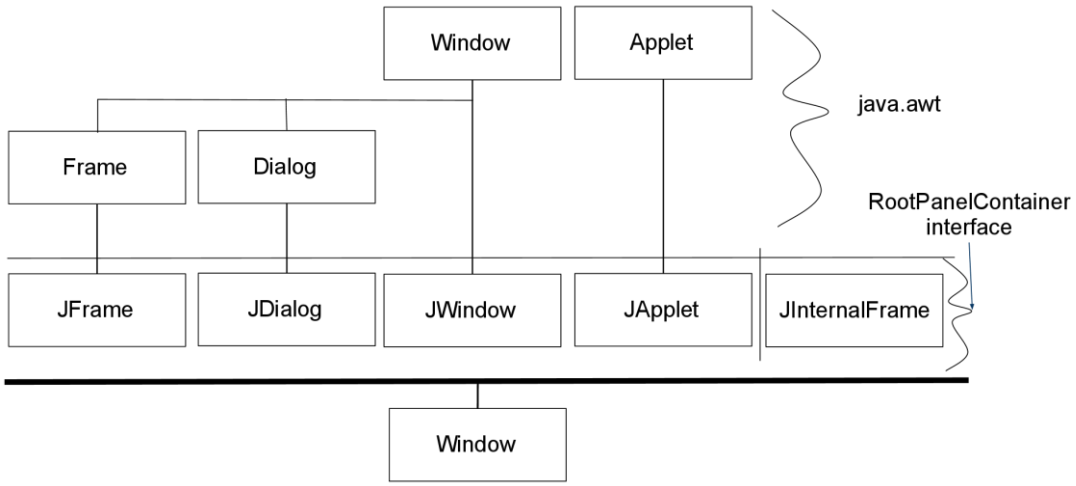
JFileChooser poskytuje jednoduchý mechanismus pro vybrání libovolného souboru. Po zavolání metody showOpenDialog() se nám zobrazí klasický výběrový dialog, který znáte ze svého systému. Další užitečná metoda je setFileSelectionMode(), která nastavuje způsob vybrání souboru. Například může zajistit, aby šel vybrat pouze jeden soubor. Poslední metoda, kterou zde zmíním je setFileFilter() tato metoda nastavuje filtr pro výběr souborů. Můžeme tím nastavit, aby se nám zobrazovali pouze soubory určitého typu. Následující kód zajišťuje otevírání souboru v mém programu.

```
JFileChooser fc = new JFileChooser();
    fc.setFileSelectionMode(JFileChooser.FILES_ONLY);
    fc.setFileFilter(new FileFilter() {
      @Override
      public boolean accept(File f) {
        if (f.isDirectory()) {
          return true;
```

```
        }
        String ext = null;
        if (f == null) {
            return false;
        }
        String s = f.getName();
        int i = s.lastIndexOf('.');

        if (i > 0 && i < s.length() - 1) {
            ext = s.substring(i + 1).toLowerCase();
        }
        if (ext == null) {
            return false;
        }
        if (ext.equals("bst") || ext.equals("BST")) {
            return true;
        }
        return false;
    }

    @Override
    public String getDescription() {
        return "Binární soubory dat (*.bst)";
    }
});

int returnValue = fc.showOpenDialog(this);
if (returnValue == 0) {
    nacteniGrafu(fc.getSelectedFile().toString());
}
```

### 8.2.7 JFrame

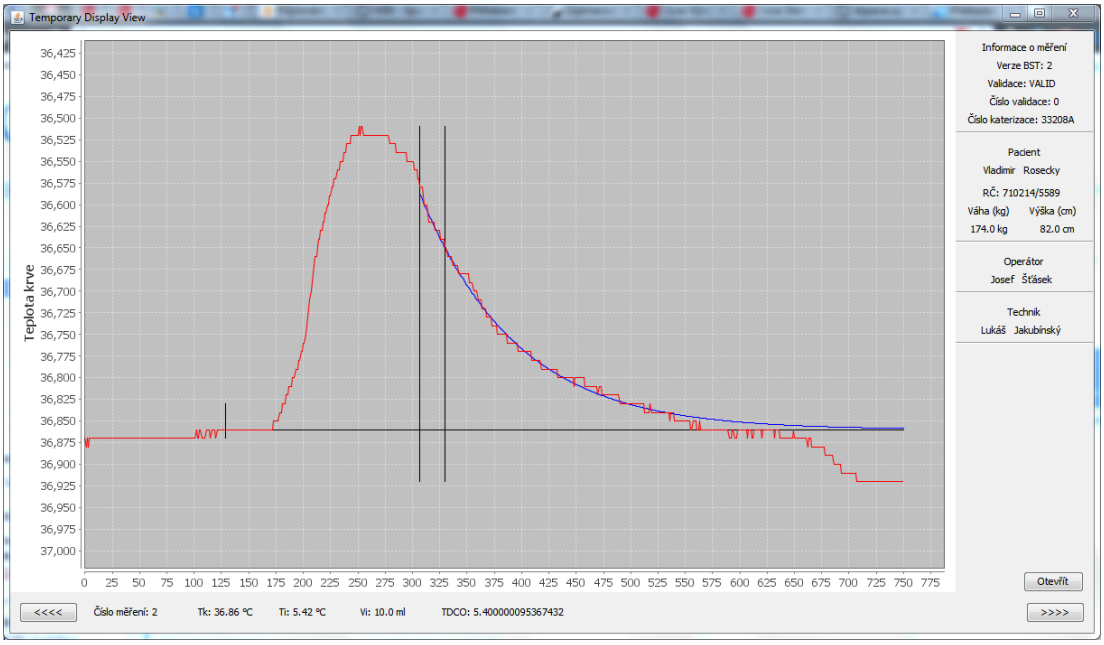
Rozšířená verze java.awt.Frame, která přidává podporu pro JFC/Swing architekturu komponent. Stejně jako všechny ostatní JFC/Swing kontejnery nejvyšší úrovně i JFrame je obsažen v JRootPane jako jeho potomek.




*Obrázek 11 – Vztah mezi třídami, které používají kořenovou tabuli.*

## 8.3 Vzhled a funkce aplikace

Vzhled aplikace je jednoduchý a přehledný. Jsou zobrazeny pouze důležité hodnoty. Okno aplikace se dělí do tří oblastí. Celkové informace o měření, informace o jednom měření a graf měření. V následujících podkapitolách budou jednotlivé oblasti popsané podrobněji.

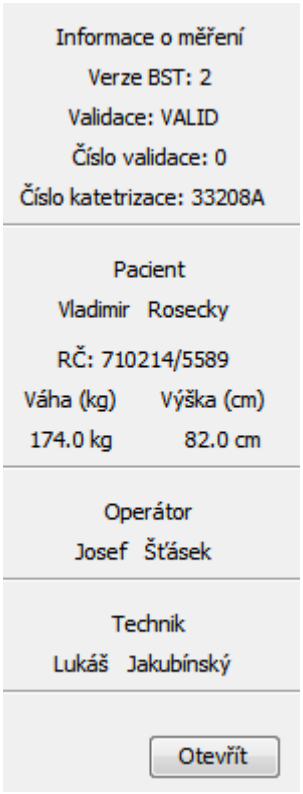



*Obrázek 12 – Celkový vzhled programu*

### 8.3.1 Celkové informace o měření

Oblast celkové informace o měření je dále rozdělena do 4 částí. V první nalezneme verzi BST, validaci, číslo validace a číslo katetrizace. V další části jsou informace o pacientovi, jako jméno, příjmení, rodné číslo, váha, výška. V následujících dvou oblastech je jméno, příjmení operátora a technika, kteří prováděli měření. A v poslední oblasti je tlačítko, které načte nový binární soubor.

Oblast je složená ze swingových komponent. Texty jsou zobrazené pomocí komponenty JLabel, oddělovací čáry jsou realizované pomocí komponenty JSeparator a tlačítko je realizováno komponentou JButton. Všechny tyto komponenty jsou umístěné na JPanel.

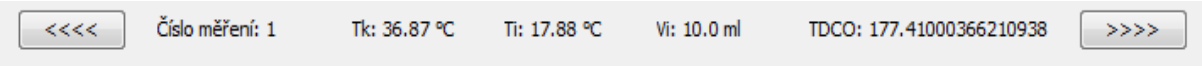



*Obrázek 13 – Celkové informace o měření*

### 8.3.2 Informace o jednom měření

Oblast informace o jednom měření poskytuje informace o právě zobrazeném měření. Je zde informace o pořadovém čísle měření, dále je zde informace o teplotě krve (Tk) a teplotě injektátu (Ti). Také zde zjistíme objem injektátu (Vi), který byl použit při tomto měření. A jako poslední je zde údaj TDCO, který nám ukazuje vypočítanou hodnotu srdečního výdeje. Dále jsou zde dvě tlačítka, která slouží k přepínání mezi měřeními.

Stejně jako předchozí oblast i tato je složená ze swingových komponent. Texty jsou zobrazené pomocí komponent JLabel a tlačítka jsou realizovaná komponentou JButton. Všechny tyto komponenty jsou umístěny na JPanelu.




*Obrázek 14 – Informace o jednom měření*

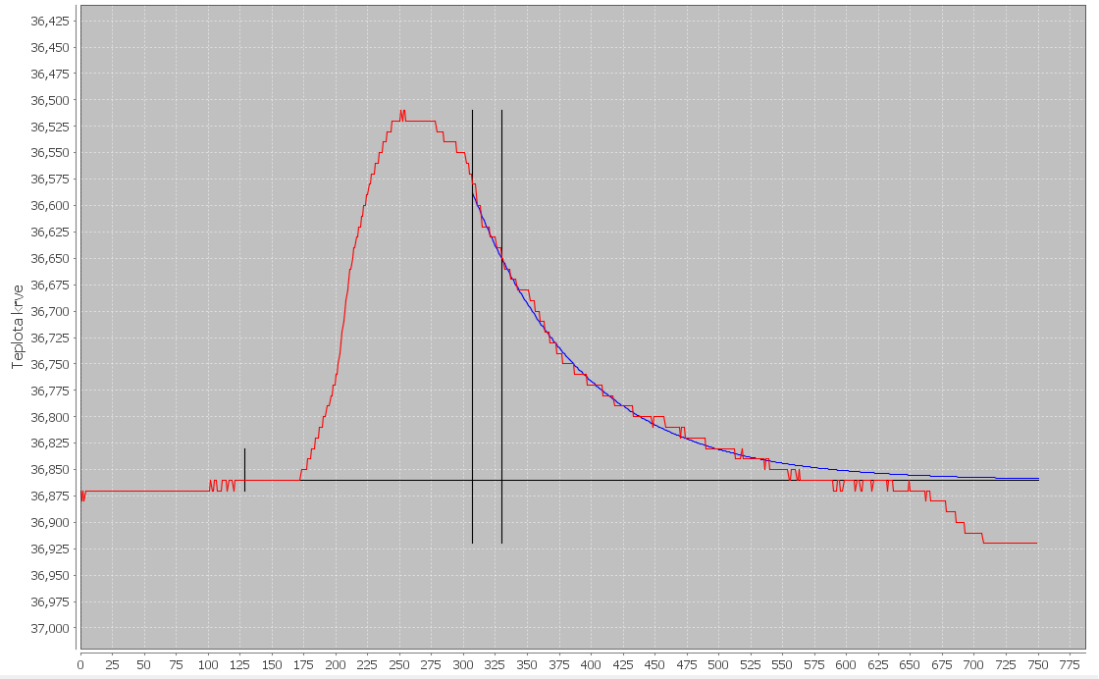
Přepínání mezi jednotlivými grafy je realizováno tímto kódem:

```
if (this.aktualni - 1 >= 0) {
      this.jPanel2.setPreferredSize(this.jPanel2.getSize());
      this.jPanel2.remove(this.seznamGrafu.get(this.aktualni));
      this.aktualni--;
      this.jPanel2.add(this.seznamGrafu.get(this.aktualni));
      this.jPanel2.repaint();
      this.pack();
      BSTDataJednohoMereni mereni;
      mereni = this.bst.getDataJednotlivichMereni().get(this.aktualni);
      jednoMereniTeplotaKrve.setText("Tk: " + mereni.getTeplotaKrve() + " °C");
      jednoMereniTeplotaInjektatu.setText("Ti:"+mereni.getTeplotaInjektatu()+"°C");
      jednoMereniObjemInjektatu.setText("Vi: "+mereni.getObjemInjektatu()+" ml");
      jednoMereniCislo.setText("Číslo měření: " + mereni.getPoradoveCisloMereni());
      jednoMereniTDCO.setText("TDCO: " + mereni.getTDCOVypocteno());
   }
```

### 8.3.3 Graf měření

Oblast graf měření zobrazuje naměření hodnoty v přehledném grafu. Na ose X je vyobrazené pořadové číslo naměřené teploty krve a na ose Y je vyobrazená hodnota teploty krve. Pozor na to že osa Y je převrácená to znamená, že je seřazená sestupně. Červená křivka znázorňuje neměřené hodnoty a modrá křivka znázorňuje dopočítanou gama funkci.

Tato oblast se skládá pouze z jedné swingové komponenty a to je JPanel. Na kterém je graf nakreslen. Přepínání mezi jednotlivými grafy se provádí tak že program má seznam ChartPanelu které se vyměňují na viditelném JPanelu.



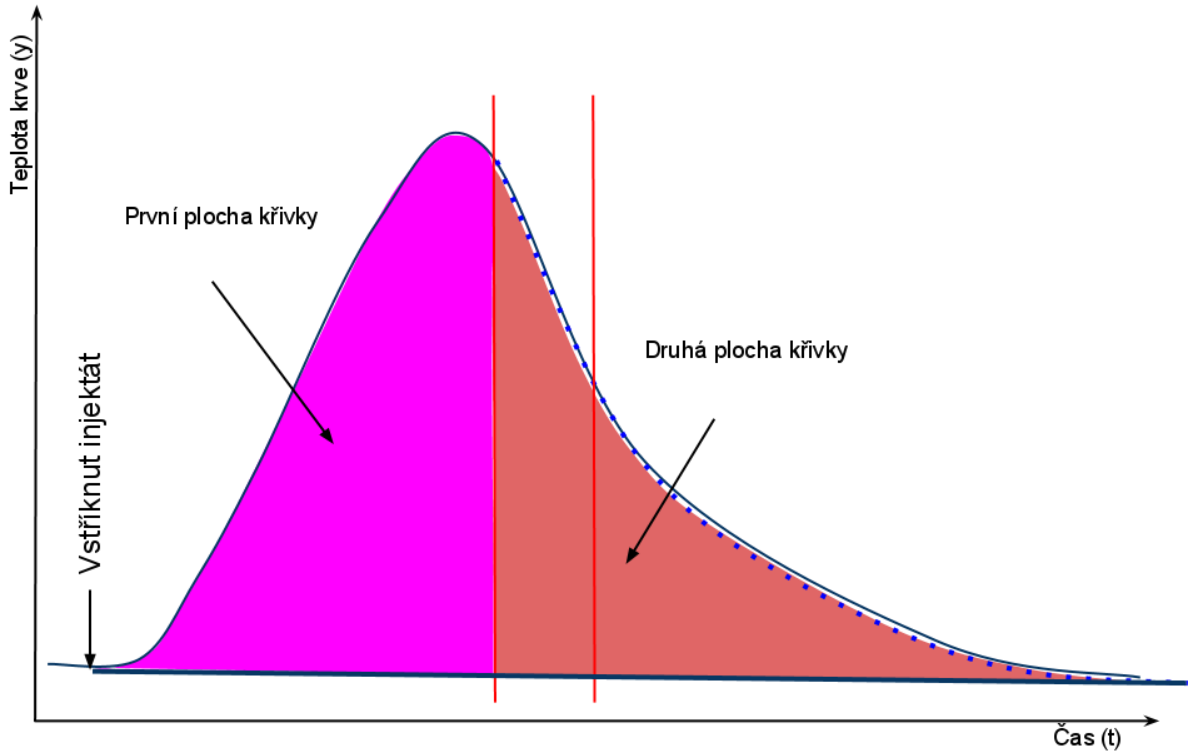

*Obrázek 15 – Graf měření*

### 8.3.4 Postup výpočtu integrálu

Výpočet integrálu v grafu se skládá ze dvou částí. Nejprve se musí vypočítat integrál pod křivkou, který začíná od bodu vstříknutí injektátu a končí začátkem gama funkce. Poté se vypočítá integrál pod dopočítanou gama křivkou. Přehledně to znázorňuje následující obrázek číslo 16. V programu je výpočet integrálu realizován obdélníkovou metodou, která je vysvětlena v teoretické části. K výpočtu integrálu jsou určeny metody ve třídě *vypocty* konkrétně medota

*public void vypocitejIntegral(BSTDataJednohoMereni mereni)* a metoda

*private float vypocetCastiIntegralu(float flValue, float flValueOld, float flXStep).*



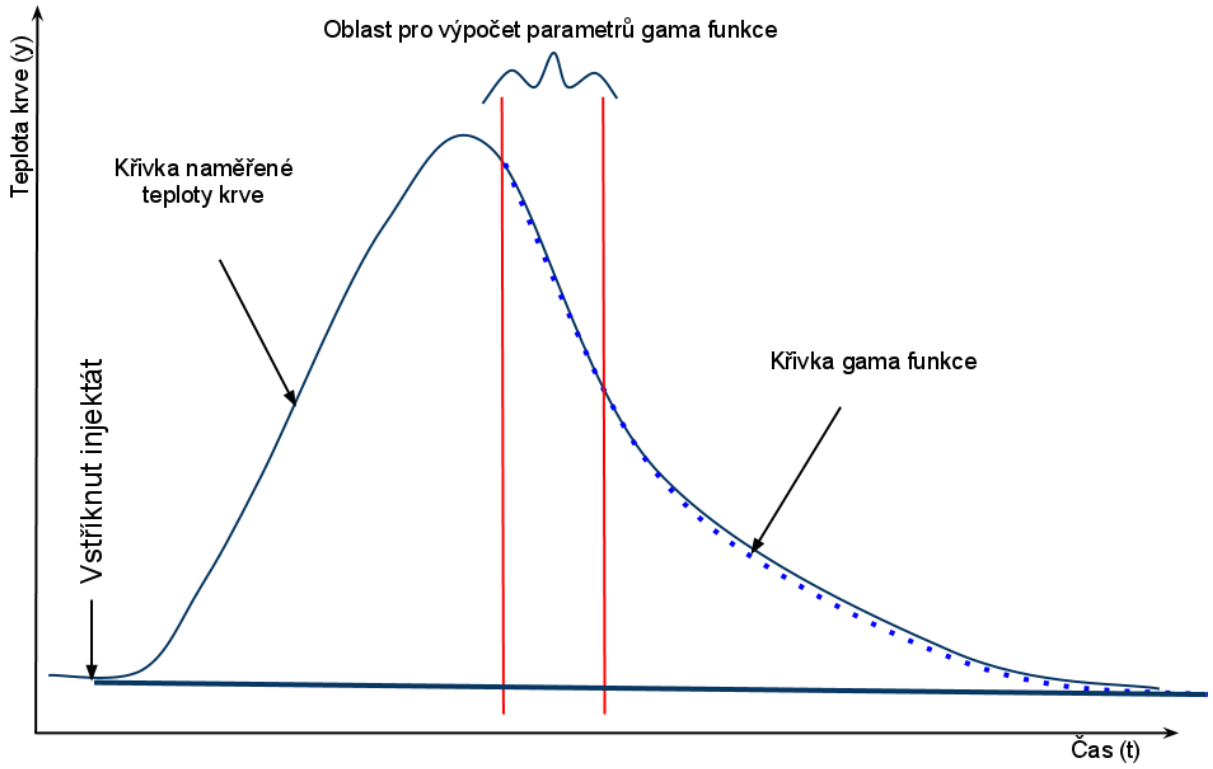

*Obrázek 16 – Postu výpočtu integrálu*

Metody pro výpočet integrálu naleznete v příloze A.

Tento postup výpočtu je nutný protože pokud bude mít pacientovo srdce některou vadu, křivka teploty krve pacienta nebude kopírovat gama křivku. Proto kdybychom počítali integrál pouze pro křivku teploty krve získali bychom chybné výsledky.

### 8.3.5 Postu výpočtu gama funkce

Gama funkce v grafu nám ukazuje průběh grafu u zdravého pacienta. Pokud se gama funkce překrývá s křivkou teploty krve, znamená to, že se u pacienta s největší pravděpodobností nevyskytuje žádná vada. Na obrázku 17 jsou znázorněny křivky zdravého pacienta.




*Obrázek 17 – Rozložení křivek v grafu*

Program gama křivku vypočítává tak že nejprve určí parametry pro výpočet gama funkce. Tyto parametry se určují z hodnot teploty krve, a to konkrétně z hodnot, které jsou mezi dvěma svislými čárami jak je znázorněné na obrázku 17. K výpočtu těchto parametrů v programu slouží metoda

```
public void gamaFunkce(BST vstup, int krivka).
```

po vypočtení parametrů se následně dopočítají hodnoty gama funkce. K tomu slouží metoda

```
public ArrayList<Float> getGama(BSTDataJednohoMereni mereni).
```

Obě tyto metody se nalézají v balíčku vypocty ve třídě vypocty a také v příloze B.

## 8.4 Chyba ve výpočtech

Při psaní tohoto programu jsem narazil na jeden problém a tím problémem byli rozdílné výsledky od původního programu Temporary Display. Původní program byl napsán v programovacím jazyce C++. Nejprve jsem si myslel, že jsem udělal někde chybu ve výpočtu, ale po dlouhém zkoumání jsem přišel na to, že chyba není v zápisu vzorce, ale v tom že Java počítá jinak než C++.

Při vypočtu parametrů gama funkce, který byl popsán výše, jsem v jazyce Java dostal následující výsledky.

| | | |
|---|---|---|
| D | double | 495792.12261891365 |
| DlnA | double | **-0.14166593551635742** |
| Db | double | 117802.58643937111 |
| Dc | double | -12876.995481600985 |
| A | double | **0.999999714263481** |
| B | double | 0.2376047965770506 |
| C | double | 0.0 |

*Obrázek 18 – Výsledky z programovacího jazyka Java*

A při vypočtu stejné křivky v jazyce C++ jsem dostal následující výsledky.

| Name | Value |
|---|---|
| flD | 496942.19 |
| flDLnA | -0.13669872 |
| flDB | 117067.30 |
| flDC | -12881.816 |
| pList->m_compparam.m_gfParam.m_flA | 0.99999970 |
| pList->m_compparam.m_gfParam.m_flB | 0.23557529 |
| pList->m_compparam.m_gfParam.m_flC | -0.025922162 |

*Obrázek 19 – Výsledky z programovacího jazyka C++*

Program Temporary Display počítá obsah pod křivkou s proměnnými typu float. Můj program počítá obsah pod křivkou s proměnnými typu double a v JAVE je double 8 bytový a v C++ je float 4 bytový na používaných 32bitových platformách. Důsledkem různé paměťové náročnosti datového typu float v C++ a double v JAVE jsou právě zanedbatelné rozdíly ve výpočtech v obouch programech.

Protože se hodnoty výsledků lišili pouze v desetinné části, rozhodl jsem se, po dohodě s vedoucím práce, tento problém dále neřešit.

# 9 Závěr

Všechny cíle stanovené v zadání práce se mi podařilo splnit. Tedy vytvořil jsem program, který načítá binární soubor, zobrazuje křivky uloženého měření a vypočítává požadované hodnoty. Program je přitom pro uživatele velmi přehledný a jednoduchý na ovládání.

Program jsem se snažil navrhnout, aby jej bylo možné dále rozšířit. Pro načítání a výpočty jsem vytvořil třídy, které nejsou závislé na grafickém zobrazení. Proto případné rozšiřování, nebo změna grafického rozhraní by neměly trvat příliš dlouho a neměly by být příliš náročné.

Nejsložitější částí programu pro mě nebylo vyřešení žádného algoritmu, ale zjištění proč mé výsledky nesouhlasí s výsledky programu Temporary Display. Strávil jsem několik týdnů hledáním chyby v mém kódu a až po analýze zdrojového kódu Temporary Display jsem zjistil, kde je chyba (viz kapitola 8.4 Chyba ve výpočtech).

Možnost využití tohoto programu v budoucnu vidím jako prohlížeč naměřených dat. Díky tomu, že je implementován v programovacím jazyce Java, je snadno přenositelný mezi platformami, a to mu dává značnou výhodu oproti původnímu programu, který je primárně určen jen pro systém Windows.

Tvorba aplikace mě bavila a nabral jsem i některé zajímavé poznatky ve svém oboru.

# Citovaná literatura


1. **Oracle Corporation and/or its affiliates.** A Brief History of NetBeans. *NetBeans.* [Online] 2011. http://netbeans.org/about/history.html.

2. **Oracle Corporation and/or its affiliates.** Vítejte u NetBeans a na stránkách www.netbeans.org. *NetBeans.* [Online] 2011. http://netbeans.org/index_cs.html.

3. **TIOBE Software BV.** TIOBE Programming Community Index for April 2011. *TIOBE Software.* [Online] 4 2011. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.

4. **Otakar, Hobza.** Kde ke svým názvům přišli? *EMAG technologický magazín.* [Online] 17. 5 2007. http://www.emag.cz/kde-ke-svym-nazvum-prisli/.

5. **DIONÉ.** Programovací jazyk Java. *DIONÉ.* [Online] 2010. http://v1.dione.zcu.cz/java/uvod.html.

6. **Oracle.** Java SE at a Glance. *Oracle.* [Online] 2011. http://www.oracle.com/technetwork/java/javase/overview/index.html.

7. **Oracle.** Java EE at a Glance. *Oracle.* [Online] 2011. http://www.oracle.com/technetwork/java/javaee/overview/index.html.

8. **Oracle.** Java ME at a Glance. *Oracle.* [Online] 2011. http://www.oracle.com/technetwork/java/javame/overview/index.html.

9. **Oracle.** JavaFX Technology At a Glance. *Oracle.* [Online] 2011. http://www.oracle.com/technetwork/java/javadb/overview/index.html.

10. **Oracle.** Java DB. *Oracle.* [Online] 2011. http://www.oracle.com/technetwork/java/javadb/overview/index.html.

11. **Oracle.** Java Card Technology. *Oracle.* [Online] 2011. http://www.oracle.com/technetwork/java/javacard/overview/index.html.

12. **Lindholm, Tim a Yellin, Frank.** The JavaTM Virtual Machine Specification. *Java Sun.* [Online] 1999. http://java.sun.com/docs/books/jvms/second_edition/html/VMSpecTOC.doc.html.

13. **Object Refinery Limited.** Welcome To JFreeChart! *JFreeChart.* [Online] 2009. http://www.jfree.org/jfreechart/.

14. **Costenoble, Stefan Waner & Steven R.** Numerical Integration. *Finite Mathematics & Applied Calculus Resource Page.* [Online] 1999. http://people.hofstra.edu/stefan_waner/realworld/integral/numint.html.

15. **Kreml, Pavel, a další.** *Matematika II.* Ostrava : VŠB TU-Ostrava, 2007. ISBN 978-80-248-1316-5 .

16. **Endrys, Jiří.** *Invazivní hemodynamické metody.* Hradec Králové : Nucleus HK, 2005. ISBN 80-86225-66-6.

17. **Object Refinery Limited.** JFreeChart 1.0.13 API Documentation. *JFreeChart.* [Online] Object Refinery Limited, 2009. http://www.jfree.org/jfreechart/api/javadoc/index.html.

18. **Oracle and/or its affiliates.** Java 2 Platform. *Class UIManager.* [Online] Oracle, 2010. http://download.oracle.com/javase/1.4.2/docs/api/javax/swing/UIManager.html.

19. **Oracle and/or its affiliates.** JavaTM Platform. *Class JLabel.* [Online] Oracle, 2010. http://download.oracle.com/javase/1.4.2/docs/api/javax/swing/JLabel.html.

20. **Oracle and/or its affiliates.** Java 2 Platform. *Class JSeparator.* [Online] Oracle, 2010. http://download.oracle.com/javase/1.4.2/docs/api/javax/swing/JSeparator.html.

# Příloha A – Metody pro výpočet integrálu

Tyto metody zajišťují výpočet integrálu křivky naměřených teplot krve. Jde o výpočet integrálu pomocí obdélníkové metody.

```
private float vypocetCastiIntegralu(float flValue, float flValueOld, float flXStep) {
      float flY, flYDelda, flSurface;
      if (flValue > flValueOld) {
         flY = flValueOld;
         flYDelda = flValue - flValueOld;
      } else {
         flY = flValue;
         flYDelda = flValueOld - flValue;
      }
      flSurface = flXStep * flY;
      flSurface += flXStep * flYDelda * 0.5f;
      return flSurface;
   }

   public void vypocitejIntegral(BSTDataJednohoMereni mereni) {
      // Metoda urci integral krivky (plochu)
      this.integral = 0;
      float flXStep = 1 / (float) mereni.getPocetNamerenechHodnotZaSekunku();
      float flValue = 0.0f, flValueOld = 0.0f;
      for (int nsize = mereni.getIndexTeplotyKdyBylVstriknutInjekrar(); nsize <=
mereni.getIndexGamafunkceDo(); nsize++) {
         float pocatecniHodnota =
mereni.getNtKratTeplotaKrve().get(mereni.getIndexTeplotyKdyBylVstriknutInjekrar());
         flValue = pocatecniHodnota - mereni.getNtKratTeplotaKrve().get(nsize);
         if (nsize > 0 && flValue >= 0.0f && flValueOld >= 0.0f) {
            this.integral += this.vypocetCastiintegralu(flValue, flValueOld, flXStep);
         }
         flValueOld = flValue;
      }
      int sizeCnt = this.gama.size() - 1;
      for (int nsize = mereni.getIndexGamafunkceDo(); nsize < sizeCnt; nsize++) {
         float pocatecniHodnota =
mereni.getNtKratTeplotaKrve().get(mereni.getIndexTeplotyKdyBylVstriknutInjekrar());
         float pom = this.gama.get(nsize);
         flValue = pocatecniHodnota - pom;
         if (flValue >= 0.0f && flValueOld >= 0.0f) {
            this.integral += this.vypocetCastiintegralu(flValue, flValueOld, flXStep);
         }
         flValueOld = flValue;
      }
   }
```

# Příloha B - Metody pro výpočet gama funkce

```
public ArrayList<Float> getGama(BSTDataJednohoMereni mereni) {
      ArrayList<Float> teploraKrve = mereni.getNtKratTeplotaKrve();
      int indexMinimum = mereni.getIndexTeplotyKdyBylVstriknutInjekrar();

      ArrayList<Float> vystup = new ArrayList<Float>();
      int t = 0;
      float pomocnaStara = 100;
      while (true) {
         float pomocna = (float) (this.A * Math.pow(t, this.B) * Math.pow(Math.E,
this.C * t));
         pomocna = teploraKrve.get(indexMinimum) - pomocna;
         vystup.add((float) pomocna);

         if (t >= mereni.getIndexGamafunkceOd()) {
            if (pomocna == pomocnaStara) {
               break;
            }
         }
         if (!(t <= 1000 + mereni.getIndexGamafunkceOd())) {
            break;
         }
         pomocnaStara = pomocna;
         pomocna = 0;
         t++;
      }
      this.gama = vystup;
      return vystup;
   }
```

# Příloha C – Přiložené CD

Přiložené CD obsahuje zdrojové kódy aplikace, s kompilovanou aplikaci ve formátu .jar a knihovnu jFreeChart.