ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA EKONOMICKÁ

Diplomová práce

Tvorba SW modulů pro počítačovou podporu KEM/PMO v OOP Java

Creation of Software Modules for KEM/PMO Computer Aided Support in OOP Java

Petr Sloup

Plzeň 2012

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta ekonomická
Akademický rok: **2011/2012**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Petr SLOUP**
Osobní číslo:	**K10N0007P**
Studijní program:	**N6209 Systémové inženýrství a informatika**
Studijní obor:	**Informační management**
Název tématu:	**Tvorba SW modulů pro počítačovou podporu KEM/PMO v OOP Java**
Zadávající katedra:	**Katedra ekonomie a kvantitativních metod**

Zásady pro vypracování:

1. Uveďte téma práce a stanovte cíle práce.
2. Analyzujte třídy modelů z KEM/PMO k algoritmickému zpracování.
3. Naprogramujte vybrané moduly jako Java třídy.
4. Proveďte numerické výpočty modelových příkladů.
5. Proveďte závěrečné hodnocení práce.

Rozsah grafických prací:

Rozsah pracovní zprávy: **60 - 80 stran**

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

Seznam odborné literatury:

- **ANDERSON, D.R., SWEENEY, P.J., WILLIAMS, T.A.** *An Introduction to Management Science: Quantitative Approaches to Decision Making.* **5th Edition, New York: West Publishing Company, 1988. ISBN 0-314-62969-6**
- **BLOCH, J.** *Effective Java.* **2nd Edition, Sant Clara, California: Sun Microsystems, 2008. ISBN 0-321-35668-3**
- **HEROUT, P.** *Java: Grafické uživatelské prostředí a čeština.* **České Budějovice: Kopp, 2007. ISBN 978-80-7232-328-9**
- **LUKÁŠ, L.** *Pravděpodobnostní modely v managementu.* **Praha: Academia, 2009. ISBN 978-80-200-1704-8**
- **SPELL, B.** *Java: Programujeme profesionálně.* **Praha: Computer Press, 2002. ISBN 80-7226-667-5**
- **RENDER, B., STAIR, R.M., HANNA, M.I.** *Quantitative Analysis for Management.* **7th Edition, Prentice Hall, 2000. ISBN 0-13-021538-4**

Vedoucí diplomové práce: **Doc. RNDr. Ing. Ladislav Lukáš, CSc.**
Katedra ekonomie a kvantitativních metod

Datum zadání diplomové práce: **30. listopadu 2011**

Termín odevzdání diplomové práce: **27. dubna 2012**

Doc. Dr. Ing. Miroslav Plevný
děkan

L.S.

RNDr. Mikuláš Gangur, Ph.D.
vedoucí katedry

V Plzni dne 30. listopadu 2011

Čestné prohlášení

Prohlašuji, že jsem diplomovou práci na téma

„Tvorba SW modulů pro počítačovou podporu KEM/PMO v OOP Java“

vypracoval samostatně pod odborným dohledem Doc. RNDr. Ing. Ladislava Lukáše, CSc. za použití pramenů uvedených v přiložené bibliografii.

V Plzni, dne ……

Podpis autora

Poděkování

Tento prostor bych rád využil k poděkování Doc. RNDr. Ing. Ladislavu Lukášovi, CSc. za kvalifikované vedení, podnětné rady a postřehy, kterými přispěl k vypracování této diplomové práce.

Obsah

Úvod

Vzhledem k tomu, že jsem se na vysoké škole učil základům programování v jazyce Java a programování mě zaujalo, a především motivace k tomu, abych se zlepšil v programování nebo zkusil vytvořit něco vlastního, co by ve výsledku mohlo být pro někoho užitečné, jsem se rozhodl pro vytvoření programu jako své diplomové práce. Zvláště když se naskytla možnost vytvořit něco pro programovou podporu předmětu KEM/PMO, jehož problematika mě zajímá, těší mě, že se jí můžu dále zabývat.

V této práci jsem si stanovil následující cíle:

- zvládnutí vývojového prostředí na bázi Eclipse,
- zvládnutí tvorby programového uživatelského rozhraní s využitím potenciálu grafiky java.swing,
- tvorba java tříd, které budou sloužit pro počítačovou podporu KEM/PMO.

V první části práce se budeme věnovat teoretickému popisu vybraných oblastí z předmětu KEM/PMO, zaměříme se především na vzorce, které se používají ve vybraných modelech a které budeme hojně využívat ve výpočetních třídách vytvořeného programu.

V druhé části si představíme program, který je předmětem této práce. Ukážeme si, jak se s ním pracuje, jak jsou řešeny různé oblasti, a budeme si prezentovat samozřejmě i vzhled programu. Také si něco povíme o zvoleném vývojovém prostředí, ve kterém byl program tvořen.

Ve třetí části budeme řešit modelové příklady, na kterých dobře uvidíme, jak jsou počítány pomocí vytvořeného programu.

1 Systémy hromadné obsluhy

Teorie hromadné obsluhy, jinak také „*Teorie front*“ se zabývá zákonitostmi v systémech hromadné obsluhy (SHO), kde existuje vstupní proud požadavků a kde probíhá jejich vyřizování (obsluha). Na tento systém, z hlediska analýzy, lze pohlížet buďto ze strany požadavku (zákazníka), nebo z pohledu obsluhy. Cílem zákazníka je být obsloužen, případně čekat na obsloužení co nejkratší dobu a cílem obsluhy může být obsloužení co nejvíce požadavků, pracovat efektivně bez zbytečných prostojů, nedovolit odchod požadavků do jiného systému z důvodu neakceptovatelně početné fronty aj. Obsluha se tedy snaží s co nejmenšími náklady obsloužit všechny požadavky, které do systému vstupují.

Jako první se teorií obsluhy zabýval dánský matematik A. K. Erlang. V roce 1909 aplikoval teorii pravděpodobnosti na systém provozu telefonní sítě. Dalším, kdo významně rozvinul tuto teorii, byl rus A. N. Kolmogorov, ovšem podoba a popis systémů hromadné obsluhy, které používáme dnes, vznikly v 50. letech pod rukama anglického matematika D. G. Kendalla. [24]

1.1 Popis systémů hromadné obsluhy

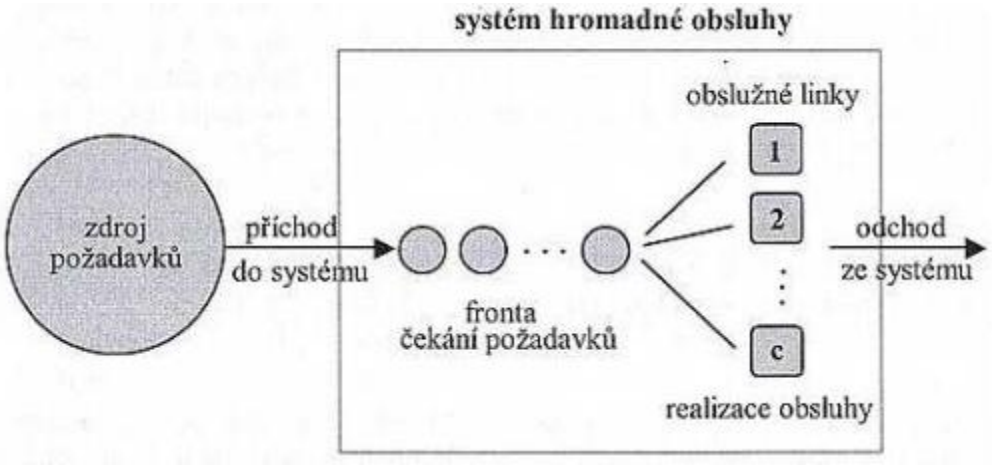
ZÁKLADNÍ POPIS SHO:

- V systému existují zařízení (kanály), která obsluhují vstupní proud požadavků (zákazníků) vstupujících do systému v průběhu času.
- Kapacita obslužných zařízení bývá zpravidla omezená, v důsledku čehož dochází k hromadění se požadavků v systému. Následně požadavky buď setrvávají ve frontě a čekají na uvolnění obslužného zařízení pro realizaci svého požadavku, nebo rezignují na obsluhu a celý systém opouštějí bez realizace svého požadavku.

DŮVODY VZNIKU FRONTY:

- Celková kapacita obslužných zařízení je trvale menší než objem vstupujících požadavků do systému, což vede k růstu fronty nade všechny meze. Takový systém se nazývá nestabilním SHO.
- Nejčastějším důvodem vzniku fronty je to, že požadavky vstupují do systému v nepravidelných časových intervalech (vstupní proud není deterministický, ale je stochastický), což znamená, že když některé požadavky přijdou do systému příliš brzy po sobě, budou na obsluhu čekat. K takovému tvoření fronty může dojít i tehdy, když je kapacita obsluhy větší než nároky vstupujících požadavků v průběhu sledovaného časového intervalu. [11]

Na obrázku 1 můžeme vidět obecné schéma systému hromadné obsluhy.



Obrázek 1 – Schéma systému hromadné obsluhy [17]

1.1.1 Základní vstupní charakteristiky pro SHO

1. Vstupní proud požadavků (intenzita vstupu) – základními údaji, ze kterých vycházíme při výpočtu charakteristik jednotlivých modelů SHO, jsou intenzity vstupu a výstupu. Vstupní proud je charakterizován počtem požadavků přicházejících do SHO za určitý časový interval. Důležitým předpokladem nejjednodušších modelů, ale také v praxi nejčastějších, je ten, že proces příchodu požadavků vyhovuje poissonovu procesu. Vstupní proud je možné charakterizovat také dobou mezi příchody dvou po sobě jdoucích požadavků. V tomto případě odpovídá poissonovu procesu, jenž určuje počet požadavků za časový interval, exponenciální rozdělení dob mezi příchody jednotlivých požadavků. Vstupní proud může být popsán i jinými procesy, u nichž se doba mezi příchody požadavků řídí jiným rozdělením (Erlangovo rozdělení, obecné rozdělení).
    - *Otevřený systém* – zdroj požadavků vstupujících do SHO je nekonečný.
    - *Uzavřený systém* – zdroj požadavků vstupujících do SHO je konečný.
2. Doba obsluhy (intenzita obsluhy) – kolik požadavků obsluha realizuje za daný časový interval. Počty vystupujících požadavků ve zvolených intervalech opět vyhovují poissonově procesu. Stejně tak je to s dobami mezi vystupujícími požadavky, které se řídí exponenciálním rozdělením.
3. Disciplína čekání ve frontě
    - *Trpělivé čekání* - požadavky ve frontě čekají trpělivě, až na ně přijde řada.
    - *Netrpělivé čekání* - po překročení určitého prahu trpělivosti (daným například počtem požadavků ve frontě) požadavek opouští systém, aniž by byl realizován.
    - *Omezený počet požadavků* – systém má omezený počet požadavků, které se v něm mohou nacházet.

Tam, kde požadavek opustí SHO ještě před realizací, vznikají ztráty.

4. Režim fronty - další věcí, kterou je pro výpočty charakteristik nutné znát, je režim vstupu požadavků z fronty do obsluhy. Základní typy režimů jsou:
    - FIFO – *first in, first out,*
    - LIFO – *last in, first out.*

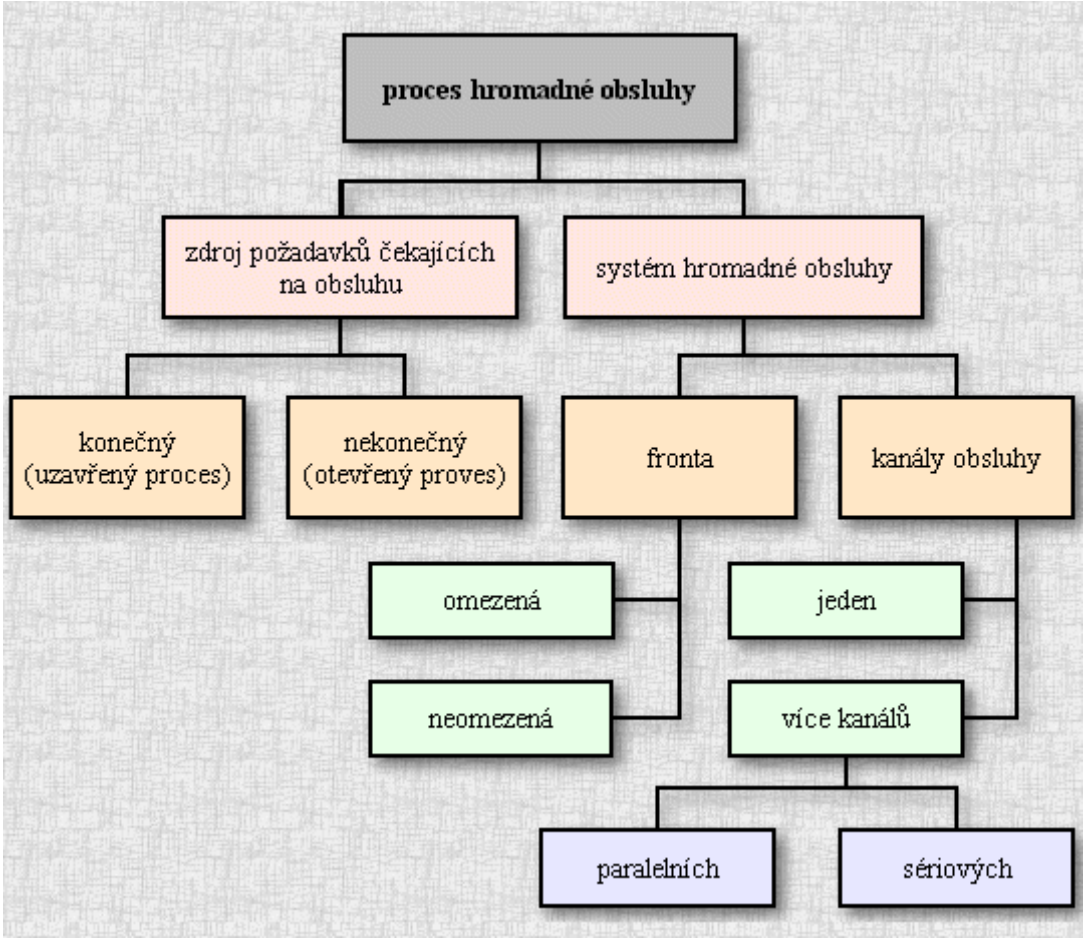
   Příklady dalších jsou SIRO – selection in random order nebo GD – general discipline.

   Také může být režim vyjádřený prioritami požadavků PRI. Existuje priorita **absolutní**, kdy prioritní požadavek je obsloužen okamžitě při vstupu do fronty bez ohledu na obsluhu ostatních neprioritních požadavků a **relativní**, kdy se nejdříve realizuje neprioritní požadavek v obsluze.

   Je nutné znát, zda je fronta jedna, či je front více při paralelním uspořádání obslužných zařízení.

5. Režim obsluhy – zde nás zajímá počet a uspořádání obslužných zařízení (kanálů). V závislosti na počtu obslužných zařízení dělíme SHO na **jednokanálové** a **vícekanálové**. Existují systémy adaptabilní, u kterých je možné měnit počet kanálů podle potřeby. Uspořádání jednotlivých obslužných kanálů může být **sériové** nebo **paralelní.** [10]

Na následujícím obrázku máme veškeré základní prvky, charakterizující SHO, znázorněny graficky v blokovém schématu procesu hromadné obsluhy.



Obrázek 2 – Blokové schéma procesu hromadné obsluhy [20]

1.2 Klasifikace SHO

V roce 1951 navrhl D. G. Kendall klasifikaci modelů SHO podle tří hlavních hledisek ve tvaru A/B/C, kde:

Pozice A – udává pravděpodobnostní rozdělení dob mezi příchody dvou po sobě jdoucích požadavků vstupujících do SHO.

Pozice B – udává pravděpodobnostní rozdělení dob obsluhy požadavků.

Pozice C – počet paralelně uspořádaných kanálů.

V průběhu času byla Kendellova klasifikace rozšířena na tvar **A/B/C/D/E/F**, kde význam A, B, C zůstává stejný a dále:

Pozice D – maximální počet požadavků v systému.

Pozice E – maximální počet požadavků ve vstupním proudu.

Pozice F – režim fronty.

Hodnoty, kterých může nabývat parametr A:

M – časové intervaly mezi příchody dvou po sobě jdoucích požadavků jsou náhodné a řídí se exponenciálním rozdělením, což znamená, že vstupní proud je reprezentován poissonovým procesem,

E_k – vstupní proud se řídí Erlangovo rozdělením,

K_n – vstupní proud se řídí chí-kvadrát rozdělením,

N – normální rozdělení,

U – rovnoměrné rozdělení,

G – doba mezi příchody je dána vlastní distribuční funkcí,

D – konstantní časové intervaly mezi příchody požadavků (deterministicky zadáno).

Parametr B, týkající se výstupního proudu, může nabývat stejných hodnot jako parametr A. [24]

Pozice **C – E** jsou jistá čísla nebo ∞. Není-li uvedeno, automaticky se předpokládá hodnota ∞.

Na pozici **F** můžou být hodnoty: FIFO (FCFS), LIFO (LCFS), PRI, SIRO. Není-li uvedeno, předpokládá se režim FIFO.

1.3 Poissonův proces

V předcházejícím textu jsme zmínili, že většina SHO předpokládá, že vstupní proud požadavků, případně výstupní proud (obsloužených požadavků), vyhovuje poissonově procesu. Nyní si popíšeme, jaké základní podmínky musí takový proud splňovat, aby mohl být za poissonovský proces považován.

1. Homogenita – počty jevů jsou ve stejně dlouhých časových intervalech konstantní.
2. Ordinarita – v dostatečně malém časovém intervalu Δt je pravděpodobnost výskytu více než jednoho jevu zanedbatelně malá. Tedy v intervalu $(t, t + \Delta t)$ se vyskytne, buď jeden jev s pravděpodobností $\lambda \Delta t$, nebo se v tomto intervalu nevyskytne žádný jev a to s pravděpodobností $1 - \lambda \Delta t$.
3. Nezávislost přírůstků – počet jevů, které se vyskytnou v jednom časovém okamžiku, je nezávislý na počtu vyskytnuvších se jevů v jiných intervalech. [24]

1.4 Exponenciální rozdělení

Exponenciální rozdělení se používá pro vyjádření času mezi výskytem dvou náhodných událostí. Je to spojité rozdělení s parametrem λ. Jak jsme si říkali, toto rozdělení pravděpodobnosti úzce souvisí s poissonovým rozdělením. Pokud poissonovo rozdělení vyjadřuje počet výskytů náhodných událostí na určitém časovém intervalu, pak exponenciální rozdělení vyjadřuje dobu do výskytu náhodné události.

$$fx(x) = \lambda e^{-\lambda x} \; pro \; x \geq 0; \; fx(x) = 0 \; pro \; x < 0 \, ,$$

$$E(x) = \frac{1}{\lambda} ,$$

$$D(x) = \frac{1}{\lambda^2} .$$

Pokud máme dána naměřená data a chceme z nich odvozovat vlastnosti modelu, je třeba ověřit, zda se řídí poissonovým rozdělením a zda na ně tedy můžeme použít vzorce pro výpočet daných charakteristik modelu. Pro ověření rozdělení se zde používá χ^2 test dobré shody. [5]

2 M/M/1 – exponenciální model s jednoduchou obsluhou

Model M/M/1 je nejjednodušším modelem. V praxi může jít například o novinový stánek a znamená to tedy, že v tomto systému je pouze jeden kanál obsluhy, zdroj a počet požadavků v systému je neomezený a režim fronty je FIFO. Požadavky ve frontě trpělivě čekají na obsloužení.

Pro všechna n jsou intenzity vstupu i obsluhy konstantní:

$$\lambda_n = \lambda, \qquad n = 0, 1, \dots ,$$

$$\mu_n = \mu, \qquad n = 0, 1, \dots .$$

Podmínka stabilizace SHO: $\rho = \frac{\lambda}{\mu} < 1.$

Kde: ρ … intenzita provozu,

λ … intenzita vstupu,

μ … intenzita obsluhy.

Průměrné intenzity vstupu a obsluhy se u tohoto modelu rovnají průběžným intenzitám:

$$\lambda^* = \lambda, \mu^* = \mu ,$$

pravděpodobnosti, že v systému je n prvků:

$$p_n = p_0 \rho^n, \qquad n = 1, 2, \dots ,$$

$$p_0 = 1 - \rho ,$$

průměrný celkový počet požadavků v systému:

$$E(N) = \frac{\lambda}{\mu - \lambda} ,$$

průměrný počet požadavků ve frontě:

$$\mathrm{E(N_f)} = \frac{\lambda^2}{\mu(\mu - \lambda)} ,$$

průměrný počet volných obslužných míst:

$$E(K) = 1 - \frac{\lambda}{\mu} ,$$

pravděpodobnost, že požadavek bude čekat:

$$P(N > 0) = \rho = \frac{\lambda}{\mu} ,$$

pravděpodobnost vzniku fronty:

$$P(N > 1) = \left(\frac{\lambda}{\mu}\right)^2 .$$

V následující části vyjádříme vzorce pro výpočet časových charakteristik, u kterých je třeba znát průměrnou intenzitu vstupu. V případě tohoto modelu je rovna intenzitě vstupu průběžné, jak jsme si uvedli výše.

Průměrná celková doba strávená požadavkem v SHO:

$$E(T) = \frac{\mathrm{E(N)}}{\lambda^*} = \frac{1}{\mu - \lambda} ,$$

průměrná doba strávená požadavkem ve frontě:

$$\mathrm{E(T_f)} = \frac{\mathrm{E(N_f)}}{\lambda^*} = \frac{\lambda}{\mu(\mu - \lambda)} .$$

[11]

3 M/M/m – vícekanálový exponenciální model

V tomto modelu je již více obslužných míst m oproti předchozímu modelu. Veškeré ostatní předpoklady zůstávají stejné. Obslužná místa jsou řazena paralelně a každé má svojí intenzitu obsluhy μ s tím, že celková intenzita obsluhy je tedy $m\mu$.

Intenzita vstupu je konstantní pro všechna n:

$$\lambda_n = \lambda, \qquad n = 0, 1, \dots ,$$

intenzita obsluhy je následující:

$$\mu_n = n\mu, \qquad n = 1, \dots, m ,$$

$$\mu_n = m\mu, \qquad n = m+1, m+2, \dots .$$

Podmínka stabilizace SHO: $\rho = \frac{\lambda}{m\mu} < 1.$

Průměrná intenzita vstupu: $\lambda^* = \lambda$.

Pravděpodobnosti, že v systému je n prvků:

$$p_n = p_0 \frac{(m\rho)^n}{n!} , \qquad n = 1, 2, \dots, m ,$$

$$p_n = p_0 \frac{(m^m \rho^n)}{m!} , \qquad n = 1, 2, \dots, m ,$$

$$p_0 = \left(\sum_{n=0}^{m-1} \left(\frac{(m\rho)^n}{n!} + \frac{(m\rho)^m}{m!\,(1-\rho)} \right) \right)^{-1} ,$$

průměrný celkový počet požadavků v SHO:

$$E(N) = \sum_{n=0}^{\infty} (np_n) = E\left(N_f\right) + m\rho ,$$

průměrný počet požadavků ve frontě:

$$E\left(N_f\right) = \frac{p_0 \mu\lambda \left(\frac{\lambda}{\mu}\right)^m}{(m-1)!\,(m\mu - \lambda)^2} ,$$

průměrný počet volných obslužných míst:

$$E(K) = (1 - \rho)m \ ,$$

pravděpodobnost, že požadavek bude čekat:

$$P(N > m - 1) = \frac{p_0 \mu \left(\frac{\lambda}{\mu}\right)^m}{(m-1)!\,(m\mu - \lambda)} \ ,$$

pravděpodobnost, že požadavek nebude čekat:

$$P(N \leq m - 1) = 1 - \frac{p_0 \mu \left(\frac{\lambda}{\mu}\right)^m}{(m-1)!\,(m\mu - \lambda)} \ ,$$

průměrná doba strávená požadavkem v SHO:

$$E(T) = \frac{E(N)}{\lambda^*} = E(T_f) + \frac{1}{\mu} \ ,$$

průměrná doba strávená požadavkem ve frontě:

$$E(T_f) = \frac{p_0 \mu \left(\frac{\lambda}{\mu}\right)^m}{(m-1)!\,(m\mu - \lambda)^2} \ .$$

[11]

4 M/M/1/k – exponenciální model s jednoduchou obsluhou a omezenou kapacitou

Tento model opět uvažuje jednokanálovou obsluhu stejně jako model M/M/1, avšak zde je rozdíl v tom, že je omezena kapacita systému, neboli je dáno, kolik může být v systému maximálně požadavků. A jelikož je zde omezen počet požadavků, znamená to, že ty požadavky, které už jsou přes limit maximálního počtu požadavků v systému, odcházejí z tohoto SHO a dochází tak ke ztrátám.

Intenzita vstupu se rozlišuje následujícím způsobem:

$$\lambda_n = \lambda, \qquad n = 0, 1, \dots, k-1 \ ,$$

$$\lambda_n = 0, \qquad n = k, k+1, \dots \ .$$

Intenzita obsluhy je konstantní:

$$\mu_n = \mu, \qquad n = 0, 1, \dots, k \ .$$

Pravděpodobnosti výskytu n požadavků:

$$p_n = p_0 \rho^n, \qquad n < k \ ,$$

$$p_n = 0, \qquad n \geq k \ ,$$

$$p_0 = \frac{1-\rho}{1-\rho^{k+1}} \ ,$$

průměrný celkový počet požadavků v SHO:

$$E(N) = \rho \frac{1-(k+1)\rho^k + k\rho^{k+1}}{(1-\rho)(1-\rho^{k+1})} \ ,$$

průměrný počet požadavků ve frontě:

$$E(N_f) = E(N) - \rho \frac{1-\rho^k}{1-\rho^{k+1}} \ ,$$

průměrná intenzita vstupu:

$$\lambda^* = \lambda(1-p_k) \ ,$$

p_k - pravděpodobnost, že v systému je právě k požadavků,

průměrná celková doba strávená požadavkem v SHO:

$$E(T) = \frac{E(N)}{\lambda^*},$$

průměrná doba strávená požadavkem ve frontě:

$$E(T_f) = \frac{E(N_f)}{\lambda^*}\ .$$

[11]

5 Modely SHO s netrpělivostí požadavků a s prioritami požadavků

5.1 Netrpělivost požadavků

Netrpělivost požadavků známe dobře z praktického života. Pokud například v obchodním domě přijdeme k pokladně, kde před námi je pět lidí s přeplněnými košíky, tak si budeme rozmýšlet, zda zůstaneme čekat nebo nakoupíme jindy, případně v jiném obchodě. Projevy netrpělivosti závisí na prahu netrpělivosti, jímž je určitý počet požadavků, od kterého se začne netrpělivost projevovat. Od této chvíle klesá intenzita vstupu až na výslednou konečnou intenzitu, která je odpozorována, a poté padá na nulu.

Rozlišujeme dva typy netrpělivostí:

- **Apriorní netrpělivost** – požadavek se vůbec nezařadí do fronty, pokud je v ní už určitý počet požadavků před ním.
- **Aposteriorní netrpělivost** – požadavek se o vystoupení z fronty rozhoduje až v závislosti na délce doby, kterou ve frontě čeká.

Pro uvažování netrpělivosti používáme model M/M/m a předpokládáme režim fronty FIFO. [14]

Vztahy pro apriorní netrpělivost:

$$\lambda_n = \lambda, \qquad n = 0, 1, .., m\,,$$

$$\mu_n = \mu, \qquad n = 0, 1, \dots, m\,,$$

$$\lambda_n = \lambda J(n), \qquad n = m+1, m+2, \dots\,,$$

$$\mu_n = \mu, \qquad n = m+1, m+2, \dots\,.$$

Kde: $J(n)$ … zadaná funkce netrpělivosti.

Mějme práh netrpělivosti n_1, od kterého se netrpělivost začne projevovat. Také mějme dáno n_2, což je počet požadavků ve frontě, při kterém klesne intenzita vstupu na určenou nejnižší úroveň $\lambda(n_2)$.

Zavedeme novou proměnnou q:

$$q = 0, \qquad n < n1 \ ,$$

$$q = n2 - n1, \qquad n \geq n1 \ .$$

Tři nejčastější tvary funkce netrpělivosti v závislosti na q:

$$\boldsymbol{J(q) = \frac{1}{(1+q)}} \ ,$$

$$\boldsymbol{J(q) = e^{-aq}} \ , \qquad a \geq 0 \ ,$$

$$\boldsymbol{J(q) = (1 - aq)}, \qquad 0 \leq aq \leq 1 \ .$$

Následně má průměrná intenzita vstupu tvar:

$$\lambda^* = \lambda \left(\sum_{n=0}^{n1} (p_n) + \sum_{n=n1+1}^{\infty} (J(n) p_n) \right) \ .$$

Vztahy pro aposteriorní netrpělivost:

$$\lambda_n = \lambda, \qquad n = 0, 1, \dots \ ,$$

$$\mu_n = n\mu, \qquad n = 1, 2, \dots, m \ ,$$

$$\mu_n = m\mu + (n - m)\nu, \qquad n = m + 1, m + 2, \dots \ .$$

Kde: ν … parametr, pomocí něhož modelujeme aposteriorní netrpělivost (znázorňuje intenzitu exponenciálního rozdělení). [11]

5.2 Priority požadavků – režim fronty PRI

Režim fronty PRI znamená, že do fronty mimo běžných požadavků vstupují také požadavky, které musí být vyřízeny přednostně. Můžeme si tuto situaci představit například na příjmu v nemocnici. Zde se může objevit jak relativní priorita, tak i absolutní priorita. Tyto dvě priority se liší tím, že u relativní priority je požadavek obsloužen ihned po uvolnění místa v systému, pokud volné místo není dostupné okamžitě. U absolutní priority musí být požadavek obsloužen bezpodmínečně ihned i za cenu přerušení obsluhy jiného požadavku.

Celková intenzita vstupu se rovná součtu intenzit běžných a preferovaných požadavků:

$$\lambda = \lambda_1 + \lambda_2 \ ,$$

$$\mu - konstantní \, ,$$

$$p_0 = 1 - \rho \ ,$$

$$p_n = (1-\rho)\rho^n, \qquad n = 1,2, \dots \ .$$

$$E(T_1) = \frac{\lambda_1}{\mu} \ , \qquad E(T_2) = \frac{\lambda_2}{\mu} \ ,$$

$$E(N_1) = \frac{\left(\frac{\lambda_1}{\mu}\right)\left(1+\rho-\frac{\lambda_1}{\mu}\right)}{1-\frac{\lambda_1}{\mu}} \ , \qquad E(N_2) = \frac{\left(\frac{\lambda_2}{\mu}\right)\left(1-\left(\frac{\lambda_1}{\mu}\right)(1+\rho)\right)}{(1-\rho)\left(1-\frac{\lambda_1}{\mu}\right)} \ .$$

[11]

Další vzorce k obsluze s prioritami najdeme ve zmíněné literatuře str. 76.

6 Teorie zásob

Teorie zásob se zabývá optimalizací zásob tak, aby zadaná kriteriální funkce dosahovala požadovaných hodnot. Teorie zásob se snaží pomocí matematického aparátu modelovat různé systémy, kde jsou zásoby potřebné pro další zpracovávání a výrobu. Jejich nedostatkem nebo naopak jejich přílišným množstvím na skladě můžou být generovány takové náklady, které by, například při lepším naplánování velikostí nebo časů dodávek materiálu na sklad, byly značně nižší. Teorie zásob se tedy na různých modelech zásob snaží optimalizovat potřebné proměnné v zadané kriteriální funkci (nákladová funkce, jiné oceňovací funkce) při zachování nutného množství zásob pro uspokojení výrobních potřeb, které jsou motivovány poptávkou. [10]

ZÁKLADNÍ POJMY TEORIE ZÁSOB:

- Fungování skladu zásob zboží:
    - jednoproduktový, či víceproduktový sklad,
    - stitické, dynamické fungování,
    - zda při dynamickém fungování dochází k čerpání a doplňování skladu periodicky nebo neperiodicky,
    - případně další specifika fungování daného skladu.
- Systém doplňování skladu:
    - okamžitý,
    - postupný,
    - náhodný,
    - se zpožděním.
- Poptávka po zásobách:
    - náhodná,
    - deterministická.
- Ztráty
- Omezení

[10]

DRUHY ZÁSOB:

- běžná zásoba,
- sezónní zásoba,
- pojistná zásoba.

MOTIVACE PRO TVORBU ZÁSOB:

- **Transakční** – zde jsou tvořeny zásoby proto, aby byla uspokojena poptávka. Vychází se z průměrných hodnot poptávky na daném trhu. Pro tento účel se tvoří běžná zásoba.
- **Predikční** – tvorba zásoby pro předpovídané výkyvy poptávky. Může se zde jednat například o sezónní výkyvy poptávky. V tomto smyslu se tvoří sezónní zásoba.
- **Spekulativní** – zde je tvořena zásoba pojistná, která má krýt náhodné výkyvy poptávky, nepředvídatelné nebo předvídatelné jen s určitou pravděpodobností. [10]

7 EOQ – model periodicky doplňovaných zásob s konstantní velikostí dodávky

Model EOQ (Economic Order Quantity) určuje optimální objednací množství materiálu tak, aby náklady na skladování zásob a náklady na objednávky materiálu byly co nejmenší.

Tento model patří mezi nejstarší a nejjednodušší modely řízení zásob podle poptávky, kde poptávka je deterministická, vyjádřená funkcí v čase a má konstantní rychlost, tudíž čerpání zásob ze skladu probíhá rovnoměrně.

Co se týče dodávek zboží, ty můžou chodit periodicky s délkou periody t nebo jsou dodány ve chvíli, kdy hladina zásob klesne na předem stanovenou mez (signální úroveň zásob). Objemy všech dodávek jsou ve stejné výši q.

Na následujícím obrázku (obrázek 3) můžeme vidět, jak se mění stav zásob na skladě v průběhu dvou period u modelu EOQ. V grafu je znázorněna i signální úroveň zásob a zpoždění mezi objednávkou a přijmutím materiálu na sklad.

Značení na obrázku:[1]

Stock Qty *– množství zásob na skladě (svislá osa)* **(z(τ))**,

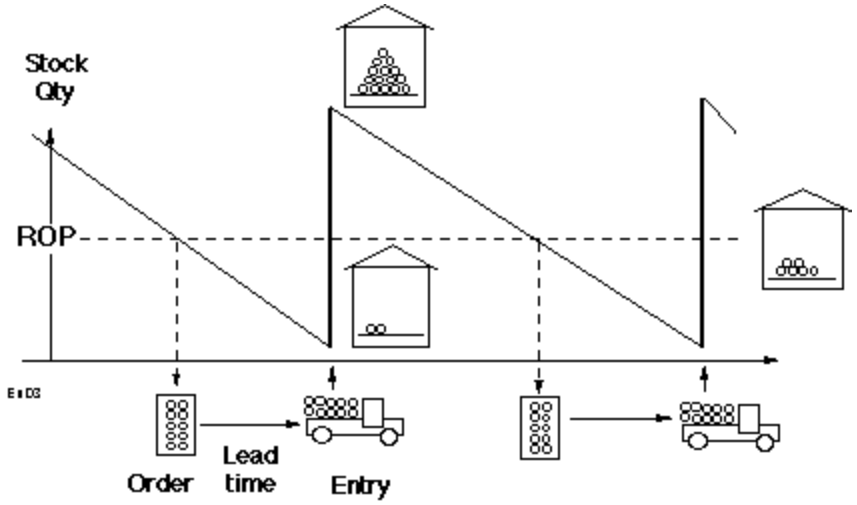
ROP *– signální úroveň zásob* **(s)**,

Order *– čas objednávky* **(t)**,

Lead time *– doba mezi objednávkou a přijetím materiálu* **(t_p)**,

Entry *– přijetí materiálu na sklad,*

Time *– čas (vodorovná osa)* **(τ)**.



Obrázek 3 – Vývoj stavu zásob na skladě (EOQ) [16]

Tvar zobrazované funkce $z(\tau)$ je následující [11]:

$$z(\tau) = q\left(1 - \frac{\tau}{t}\right), \qquad \tau \in \langle 0, t \rangle .$$

[1] Toto značení je použito pouze v příslušném obrázku, naše značení je poznamenáno v závorce

Používané značení pro EOQ model:

T- doba řízení skladu,

Q - celková potřeba zboží za dobu T,

t - délka doplňovacího cyklu (perioda, cyklus),

q – objem dodávky zboží,

c_1 - jednotkové náklady na skladování za jednotku času,

c_3 - fixní náklady na jednu objednávku dodávky.

Pro celkové náklady jednoho cyklu platí:

$$N(q) = c_3 + \frac{c_1 q t}{2} ,$$

celkové náklady za dobu řízení skladu:

$$N_T(q) = nN(q) ,$$

celkové náklady na jednotku času:

$$C(q) = \frac{N(q)}{t} ,$$

optimální velikost dodávky:

$$q_{opt} = \sqrt{\frac{2c_3 Q}{c_1 T}} ,$$

optimální délka dodávkového cyklu:

$$t_{opt} = \frac{q_{opt} T}{Q} = \sqrt{\frac{2c_3 T}{c_1 Q}} ,$$

optimální počet cyklů:

$$n_{opt} = \frac{Q}{q_{opt}} = \sqrt{\frac{c_1 Q T}{2c_3}} ,$$

celkové náklady na jednotku času při q_{opt}:

$$C(q_{opt}) = \sqrt{\frac{2c_1c_3Q}{T}} ,$$

celkové náklady za dobu řízení skladu T:

$$N_T(q_{opt}) = \sqrt{2c_1c_3QT} ,$$

průměrná výše zásob v průběhu jednoho cyklu:

$$q^* = \frac{q}{2} ,$$

$$q^*_{opt} = \sqrt{\frac{c_3Q}{2c_1T}} .$$

[11]

Značení na obrázku:[2]

Costs *– náklady (svislá osa)* **(C_q)**,

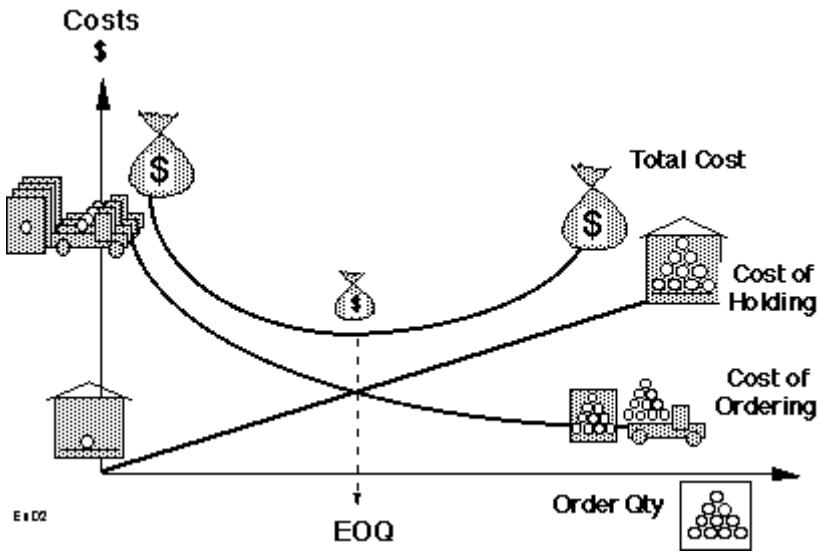
Total costs*– celkové náklady skladování* **(C_q)**,

Cost of Holding *– náklady na skladování* **(c_1)**,

Costs of Ordering *– náklady na objednávku zboží* **(c_3)**,

EOQ *– bod optima =* q_{opt} *, zde jsou nejnižší celkové náklady*,

Order Qty*– objednávané množství (vodorovná osa)* ***(q)***.



Obrázek 4 – Výše nákladů při optimální velikosti dodávky (EOQ) [16]

Na předchozím obrázku (obrázek 4) vidíme, že metoda EOQ hledá nejnižší bod na křivce celkových nákladů, čehož dosáhne položením derivace $\frac{dC(q)}{dq} = 0$. Z této derivace vzejde vzorec pro optimální velikost dodávky, který máme již uveden spolu s ostatními vzorci pro model EOQ.

[2] Toto značení je použito pouze v příslušném obrázku, naše značení uvedeno v závorce

7.1 EOQ s přechodným nedostatkem zásob

Tento model uvažuje případ, kdy je povoleno přechodné neuspokojení poptávky, neboli přechodného nedostatku zásob na skladě. Z toho vyplývá, že funkce $z(\tau)$ může nabývat i záporných hodnot. Pokud je tato funkce záporná, pak při příští dodávce je ihned odčerpána část zboží na uspokojení poptávky ve výši záporného $z(\tau)$.

Pouze ve stručnosti uvedeme důležité vzorce:

$$C(S,q) = \frac{c_1 S^2}{2q} + \frac{c_2 (q-S)^2}{2q} + \frac{c_3 Q}{qT} \ ,$$

kde: S … zásoba zboží na skladě po přijetí dodávky a okamžitém odčerpání neuspokojené poptávky předchozího období $(q - S)$,

c_2 … jednotkové náklady (ztráty) z nedostatku.

$$q_{opt} = \sqrt{\frac{2c_3 Q}{c_1 T}} \sqrt{\frac{c_1 + c_2}{c_2}} \ ,$$

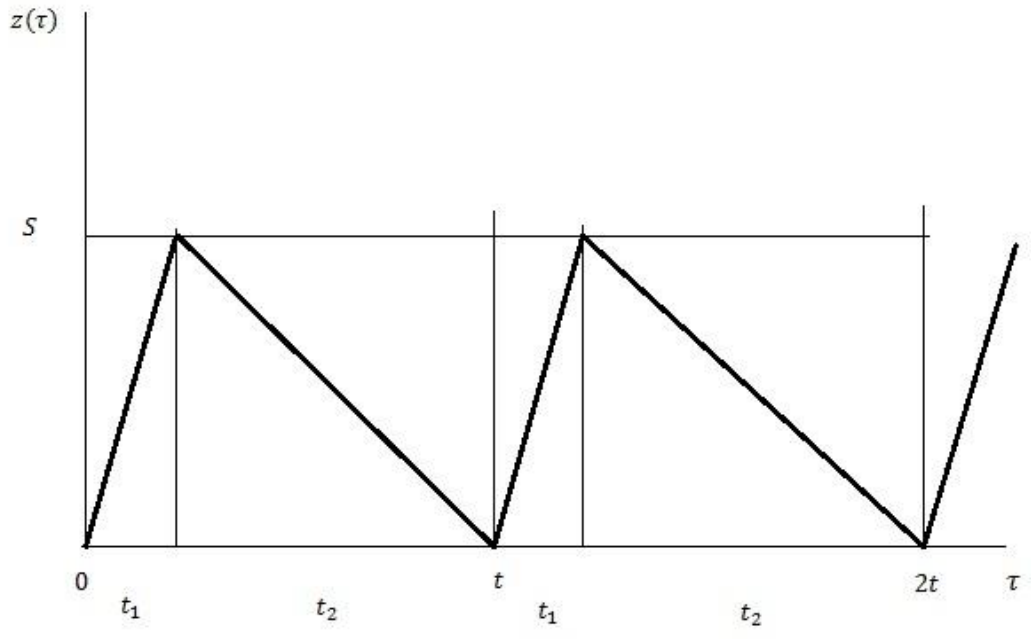
$$S_{opt} = \left(\frac{c_2}{c_1 + c_2}\right) q_{opt} \ ,$$

$$t_{opt} = \sqrt{\frac{2c_3 T}{c_1 Q}} \sqrt{\frac{c_1 + c_2}{c_2}} \ .$$

[10]

8 POQ – model periodicky doplňovaných zásob s konečnou rychlostí doplňování

U POQ (Production Order Quantity) modelu jsou předpoklady stejné jako u EOQ s výjimkou jednoho a to je ten, že dodávky nepřichází okamžitě, ale v určitém okamžiku je nastartována výroba (doplňování skladu), která probíhá současně s čerpáním ze skladu. Z toho jasně vyplývá, že je nutné, aby rychlost doplňování byla vyšší než rychlost čerpání ze skladu. U tohoto modelu tedy ve chvíli potřeby začneme doplňovat zboží, až doplníme sklad na úroveň S (obrázek 5).



Obrázek 5 – Stav na skladě v čase: model POQ. Funkce $z(\tau)$ udává množství zboží na skladu v čase τ [10]

Tvar zobrazované funkce $z(\tau)$ [11]:

$$z(\tau) = \mathrm{z}_1(\tau) = (\mathrm{v} - \mathrm{d})\tau, \qquad \tau \in \langle 0, \mathrm{t}_1 \rangle \ ,$$

$$z(\tau) = \mathrm{z}_2(\tau) = S - d(\tau - t_1), \qquad \tau \in \langle t_1, t \rangle$$

Používané značení pro POQ model:

t - délka doplňovacího cyklu (perioda, cyklus),

t_1 - doba, kdy probíhá současně čerpání i doplňování zboží,

t_2 - doba, kdy probíhá pouze čerpání zboží,

S – nejvyšší hladina zásoby během cyklu,

v - rychlost doplňování skladu,

d - rychlost čerpání ze skladu.

Zbylé značení je stejné jako u modelu EOQ.

Rychlost doplňování skladu je konstantní po dobu t_1a nulová po zbylou dobu t_2. Rychlost čerpání je také konstantní a to po celou dobu t.

Pro velikost dodávky platí:

$$q = vt_1 = dt \ ,$$

celkové náklady na jeden cyklus t:

$$N(q) = c_3 + \frac{c_1 q(t - t_1)}{2} \ ,$$

celkové náklady na jednotku času:

$$C(q) = c_1 q \frac{\left(1 - \frac{T_1}{T}\right)}{2} + \frac{c_3 Q}{qT} \ ,$$

optimální velikost dodávky:

$$q_{opt} = \sqrt{\frac{2c_3 Q}{c_1(T - T_1)}} \ ,$$

optimální délka cyklu, optimální počet cyklů:

$$t_{opt} = \frac{q_{opt}\,T}{Q} \ ,$$

$$n_{opt} = \frac{Q}{q_{opt}} \ ,$$

celkové náklady na jednotku času při q_{opt}:

$$C(q_{opt}) = \sqrt{2\,\frac{c_1 c_3 Q\left(1 - \frac{T_1}{T}\right)}{T}} \ ,$$

celkové náklady za dobu řízení skladu T:

$$N_T(q_{opt}) = \sqrt{2c_1 c_3 Q(T - T_1)} \ .$$

[11]

Porovnáme-li vzorce pro optimální velikost dodávky u modelu EOQ a POQ, na první pohled vidíme, že u POQ modelu vyjde díky nižšímu čitateli ve zlomku optimální dodávka vyšší. To je dáno tím, že když v POQ modelu doplňujeme zboží, tak zároveň čerpáme, a proto je dodávkové množství ve chvíli dovršení doplňování o něco vyšší než byla velikost okamžitého doplnění u EOQ modelu, kde jsme počítali s jednorázovým uskladněním právě přijaté dodávky ve výši q_{opt}.

8.1 POQ s přechodným nedostatkem

t - délka cyklu: $t = t_1 + t_2 + t_3 + t_4$,

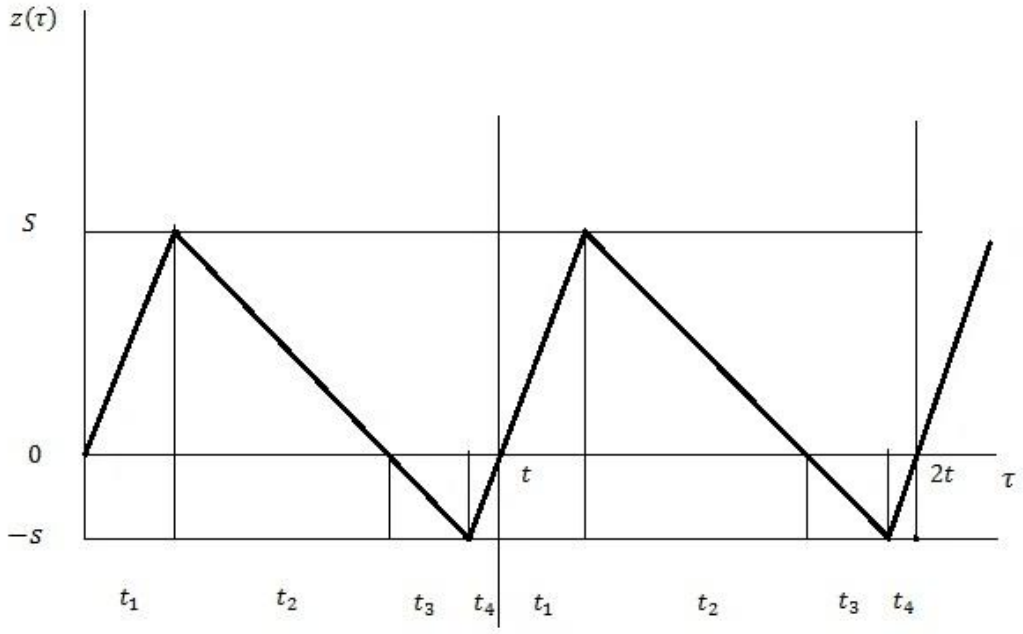
t_1, t_2 – probíhá doplňování i čerpání,

t_3, t_4 – probíhá pouze čerpání,

S - maximální zásoba během cyklu,

$-s$ - minimální zásoba během cyklu,

c_2 - jednotková ztráta z nedostatku zboží.



Obrázek 6 – Stav zásob na skladu: model POQ s přechodným nedostatkem. Funkce $z(\tau)$ znázorňuje množství zboží na skladu v čase τ. [10]

Tvar funkce $z(\tau)$ pro POQ s přechodným nedostatkem [11]:

$$z_1(\tau) = (v - d)\tau, \qquad \tau \in \langle 0, t_1 \rangle \ ,$$

$$z_{2,3}(\tau) = S - d(\tau - \tau_1), \qquad \tau \in (t_1, t_1 + t_2 + t_3 > \ ,$$

$$z_4(\tau) = \ -s + (v - d)\big(\tau - (t_1 + t_2 + t_3)\big), \qquad \tau \in (t_1 + t_2 + t_3, t > .$$

Základní vzorce k modelu POQ s přechodným nedostatkem:

$$\gamma = \frac{v}{d(v-d)} ,$$

$$C(t,S) = \frac{c_3 + 0.5\gamma(c_1+c_2)S^2}{t} + \frac{0.5c_2 t}{\gamma} - c_2 S ,$$

$$S_{opt} = \sqrt{\frac{2c_3 d\left(1-\frac{d}{v}\right)}{c_1\left(1+\frac{c_1}{c_2}\right)}} ,$$

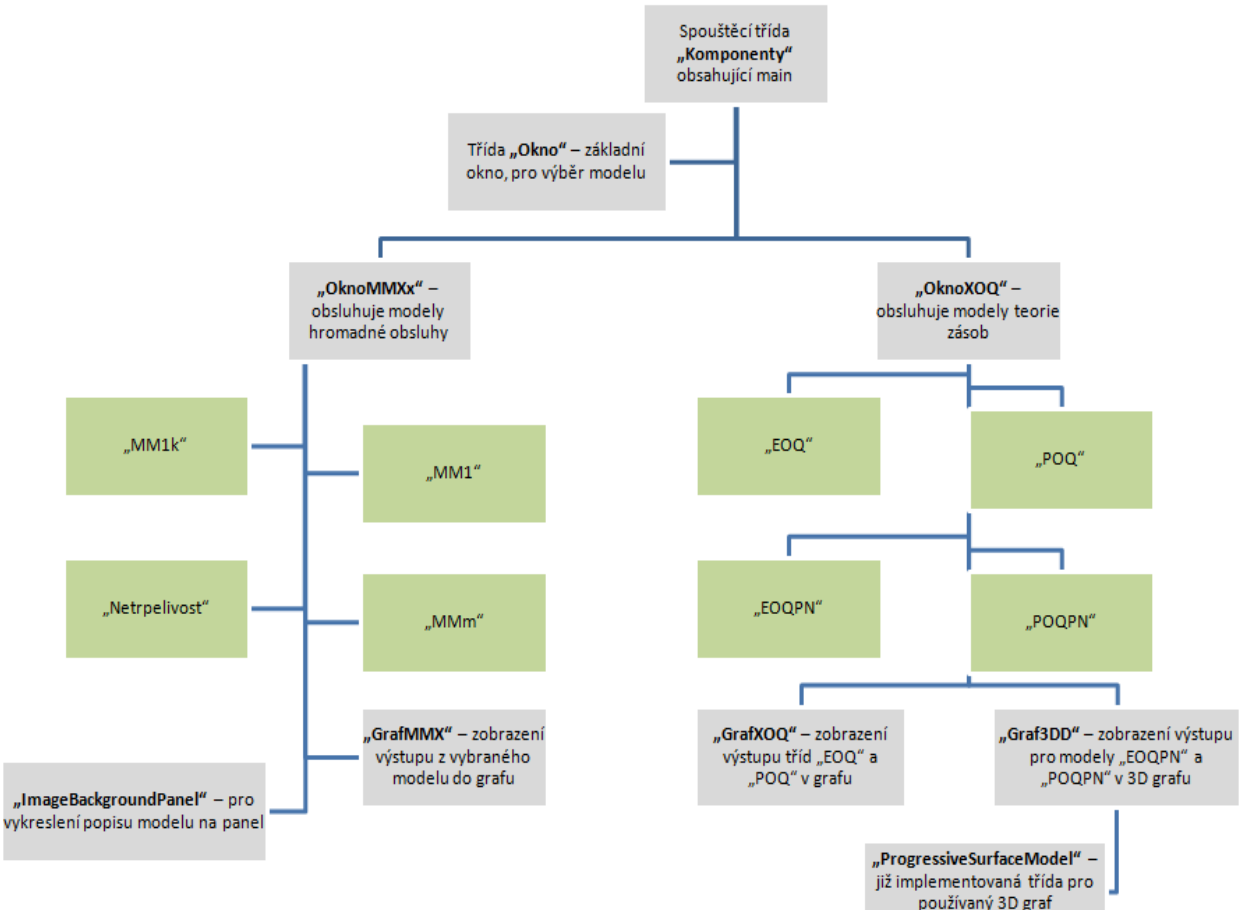
$$t_{opt} = \sqrt{\frac{2c_3\left(1+\frac{c_1}{c_2}\right)}{c_1 d\left(1-\frac{d}{v}\right)}} ,$$

$$q_{opt} = \sqrt{\frac{2c_3 d\left(1+\frac{c_1}{c_2}\right)}{c_1\left(1-\frac{d}{v}\right)}} .$$

[11]

Tímto končíme teoretickou část. Teoretická část by měla čtenáře uvést do problematiky a hlavně by měla být podkladem pro praktickou část, která následuje. V praktické části nejdříve zevrubně popíšeme stavbu vytvořeného výpočetního programu a dále budeme počítat modelové příklady, abychom si demonstrovali práci s programem a jeho výstupy.

9 Struktura vytvořeného programu



Obrázek 7 – Struktura programu vytvořeného pro počítačovou podporu KEM/PMO [vlastní zpracování]

Na obrázku jsou barevně odlišeny třídy výpočetní (zelené) a třídy grafického uživatelského rozhraní (šedé). Navrchu je třída spouštěcí. Program je vypracován v jazyce Java. Programován byl ve vývojovém prostředí Eclipse. Dohromady ho tvoří 17 tříd a externě dodané knihovny, povětšinou knihovny grafických nástrojů. V následující části budeme postupně odhalovat jednotlivé třídy blíže.

10 Vývojové prostředí Eclipse

V této části bych chtěl pouze stručně popsat vývojové prostředí, ve kterém jsem vypracoval program pro podporu KEM/PMO.

Proto, abychom mohli začít programovat v jazyce Java v prostředí Eclipse je třeba založit nový Java Project, v něm si vytvořit balík a pak už vytvářet naše jednotlivé třídy programu. Eclipse svými nástroji přispívá k vyšší rychlosti a efektivnosti programování. Hovořím zejména o automatické doplňování částí kódu nebo automatické nabídce variant, ze kterých můžeme vybírat v případě hledání nějaké vhodné integrované metody nebo deklarované proměnné.

Velice důležitým pomocníkem je při detekování chyb. Ve většině případů rozpozná špatně napsaný úsek a oznámí nám přesnou pozici této chyby v kódu. Pro ostatní problémy je v Eclipsu integrován nástroj Debug, ve kterém pomocí Breakpoints můžeme postupně procházet kód a lokalizovat tak přesně problémové místo. Mimo chyb také hlásí různé jiné skutečnosti, které si vyslouží pozornost – nepoužité proměnné, metody, třídy, upozornění, že program stále běží a my se ho pokoušíme spouštět znovu a spoustu jiných.

V Eclipsu je také mnoho nástrojů pro automatické generování různých standardních částí. Mluvím zejména o automatickém generování různých cyklů (for, switch, while), díky čemuž nemusíme celý cyklus vypisovat, pouze vyplníme proměnné části, jak potřebujeme. Dále jsem hojně využíval automatické generování ošetření výjimek (bloky try – catch) a generování getterů pro vybrané proměnné.

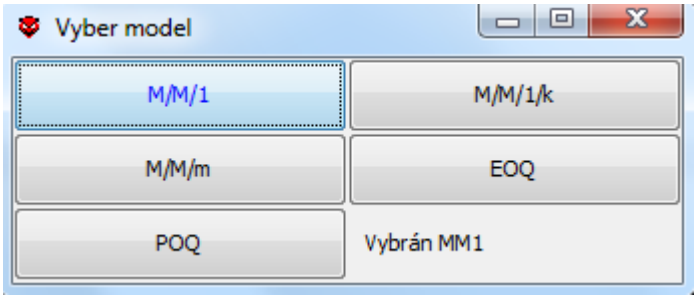
Z Eclipsu je možné exportovat náš projekt ve formě spustitelného JAR souboru a poté tedy náš program spouštět bez nutnosti současného běhu vývojového prostředí.

Dále jsem využil generování dokumentace tzv. Javadoc, která slouží ke stručnému popisu programu, jeho tříd a metod, ve formě textového dokumentu.

Samozřejmostí jsou různá grafická odlišení částí kódu. Eclipse rozlišuje všechna předvolená slovíčka, která jsou nutná k definování tříd, metod, proměnných apod. Graficky rozlišuje charaktery proměnných, které jsou lokální, globální, statické, finální a rozlišuje typ proměnné a její název, který zadáváme podle vlastního uvážení.

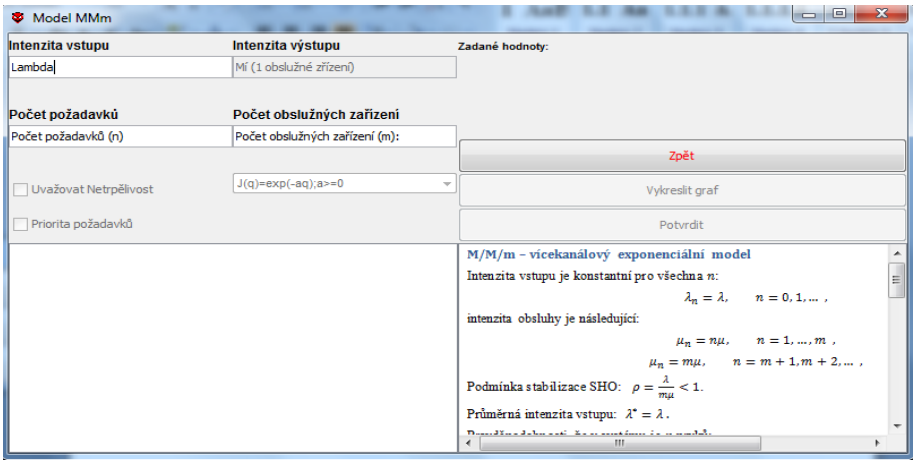
11 Popis programu

Hned po spuštění programu se nám objeví okno „*Vyber model*“ s nabídkou tří modelů hromadné obsluhy a dvou modelů z teorie zásob (obrázek 8). Toto okno ovládá třída „Okno“.



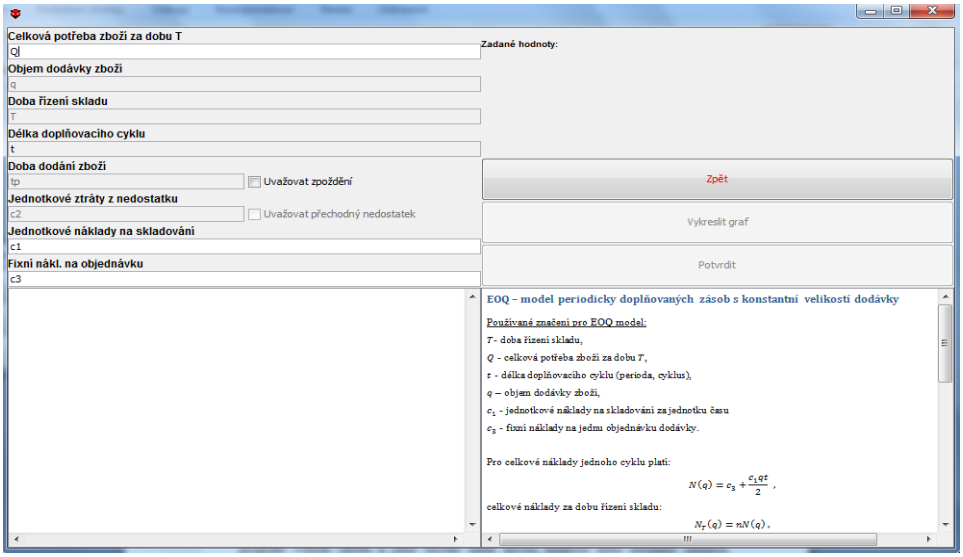
Obrázek 8 – Okno „*Vyber model*“

Z třídy „Okno“ jsou volány, podle toho jaký model vybereme, třídy „OknoMMXx“ nebo „OknoXOQ“. Z názvů můžeme rozpoznat, že první zmíněná třída bude obsluhovat modely hromadné obsluhy a druhá třída bude obsluhovat modely teorie zásob. Ať už se spustí jakákoli třída, zobrazí se před námi podobné okno. Lišit se mezi sebou budou samozřejmě údaji, které jsou potřeba zadat a u jednotlivých modelů také můžou být přidány komponenty navíc. To se, z modelů hromadné obsluhy, týká modelu M/M/m, který v sobě zahrnuje také model s uvažováním netrpělivosti (můžeme z něj spouštět výpočetní třídu „Netrpělivost“), jelikož netrpělivost se modeluje právě na tomto typu M/M/m (okno pro vstupy a výstupy modelu M/M/m je na následujícím obrázku 9).



Obrázek 9 – Okno pro vstupy a výstupy modelu M/M/m

U modelů teorie zásob jsou v obou modelech EOQ i POQ zabudovány také jejich rozšířené verze s přechodnými nedostatky (třídy „EOQPN“ a „POQPN“). Tato rozšíření jsou realizována přes zaškrtávací políčko v hlavním okně, kde jsou zadávány vstupy do modelu. Na obrázku 10 vidíme okno třídy „OknoXOQ“, konkrétně pro model EOQ a EOQ s přechodným nedostatkem.

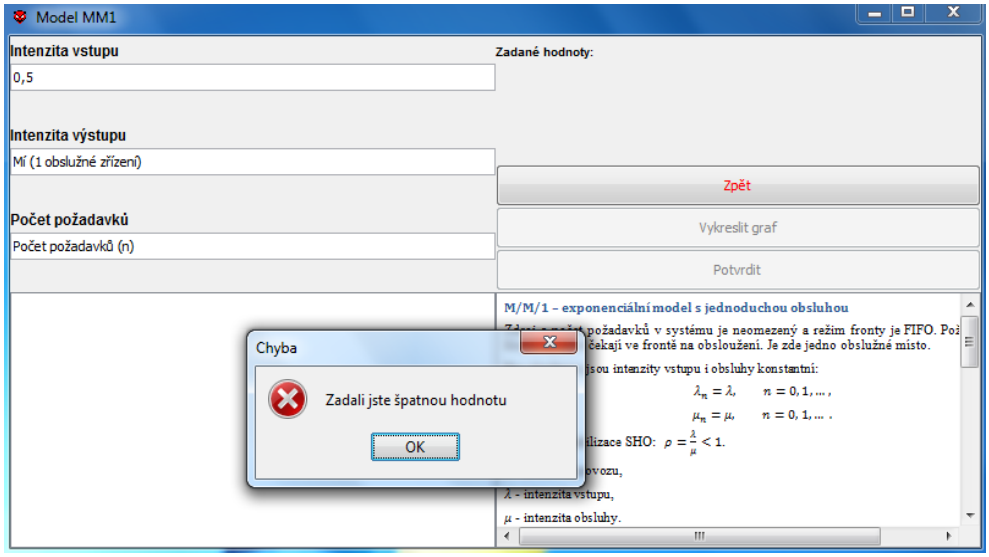


Obrázek 10 – Okno pro vstupy a výstupy modelu EOQ a EOQ s přechodným nedostatkem

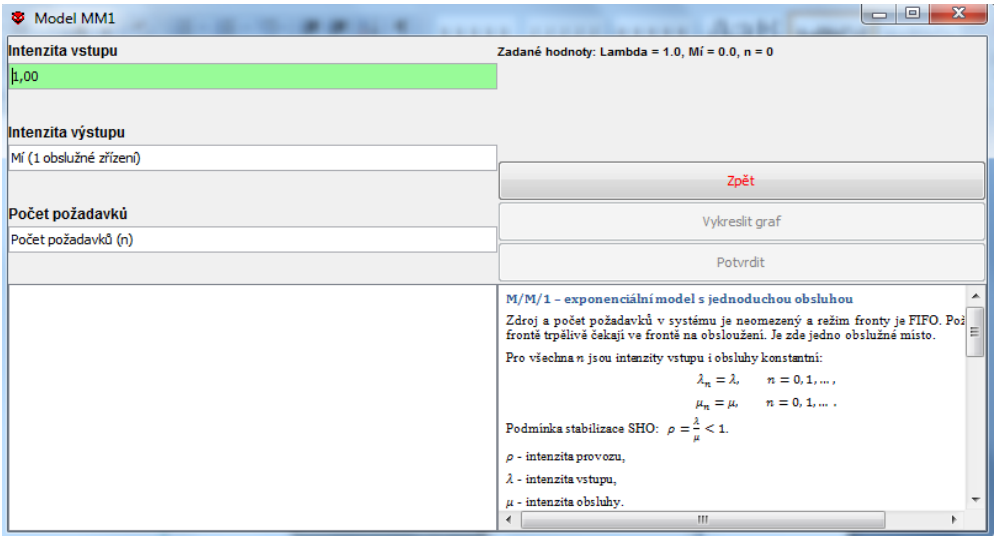
Nyní se budeme zabývat již třídami „OknoMMXx“ a „OknoXOQ“. Následný popis se bude týkat obou těchto tříd. Budeme řešit zadávání hodnot uživatelem a programové ošetření vstupů.

11.1 Zadávání vstupů

Co se týče zadávání vstupů, je třeba zmínit, že při zadávání čísel v desetinném tvaru se musí psát jako desetinný oddělovač tečka. V případě, že napíšeme jiný oddělovač, program vyhodí chybu a jsme nuceni zadat novou hodnotu nebo necháme zadanou hodnotu, kterou v tu chvíli program sám vygeneruje (obrázek 11 a obrázek 12). Načítání vstupů probíhá přes komponenty java.swing - JFormattedTextFields.

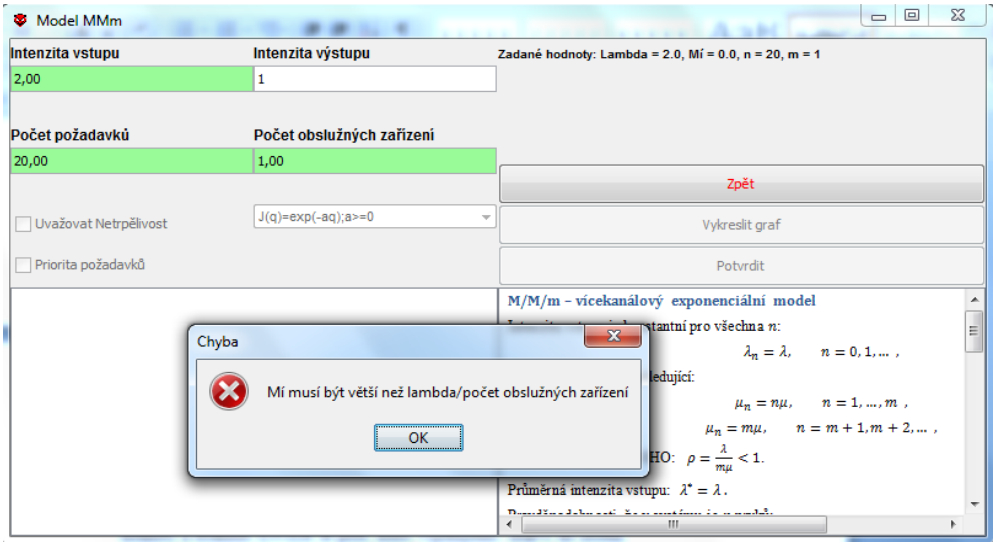


Obrázek 11 – Špatné zadání vstupu



Obrázek 12 – Automaticky dosazená hodnota programem po zadání špatného vstupu

Pokud zadáme jakoukoli jinou neakceptovatelnou hodnotu, program nás na to upozorní (obrázek 13, 14). V případě zadání písmen bude čekat na číselnou hodnotu.

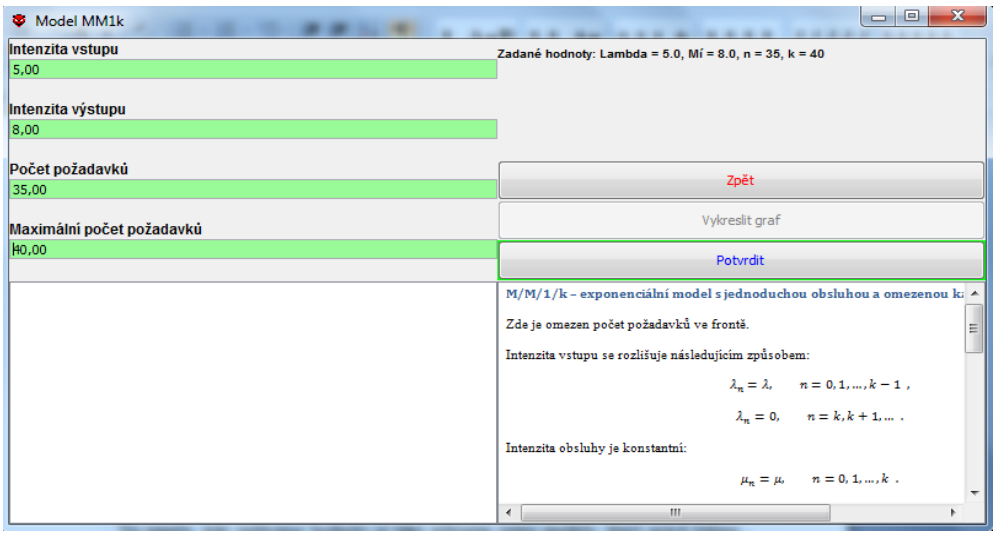


Obrázek 13 – Zadána neakceptovatelná hodnota

```
try {
    N = Integer.parseInt(e.getActionCommand());
} catch (NumberFormatException e1) {
    JOptionPane.showMessageDialog(null, "Zadali jste špatnou hodnotu","Chyba", JOptionPane.ERROR_MESSAGE);
    N=10;
    e1.printStackTrace();
}
```

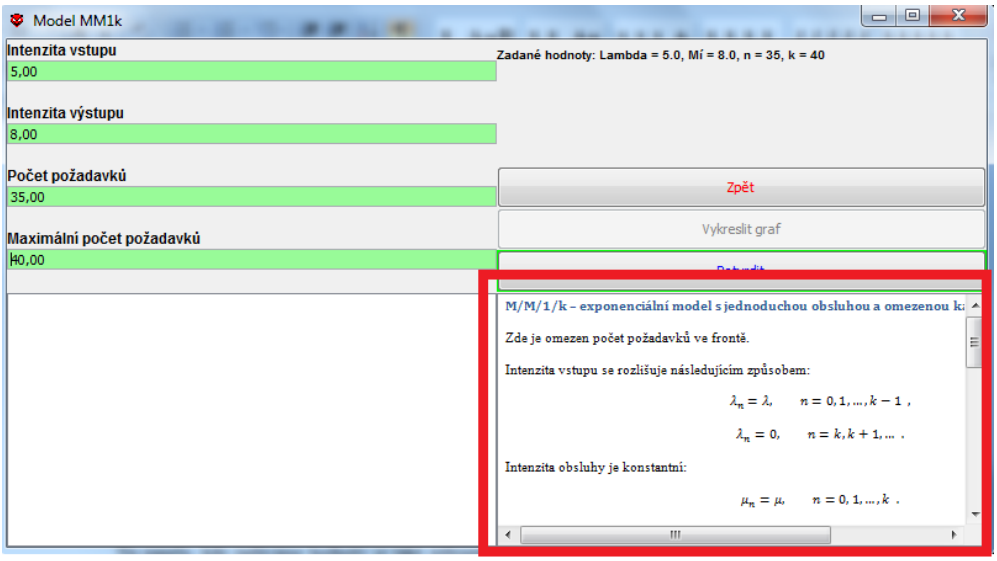
Obrázek 14 – Příklad ošetření vstupu při zadání hodnoty ve špatném formátu

Po zadání akceptovatelné hodnoty a stisknutí ENTER se pole, které vyplňujeme, zbarví do zelena a na návěstí vedle vyplňovaných okének se zobrazí zadané hodnoty (obrázek 15).



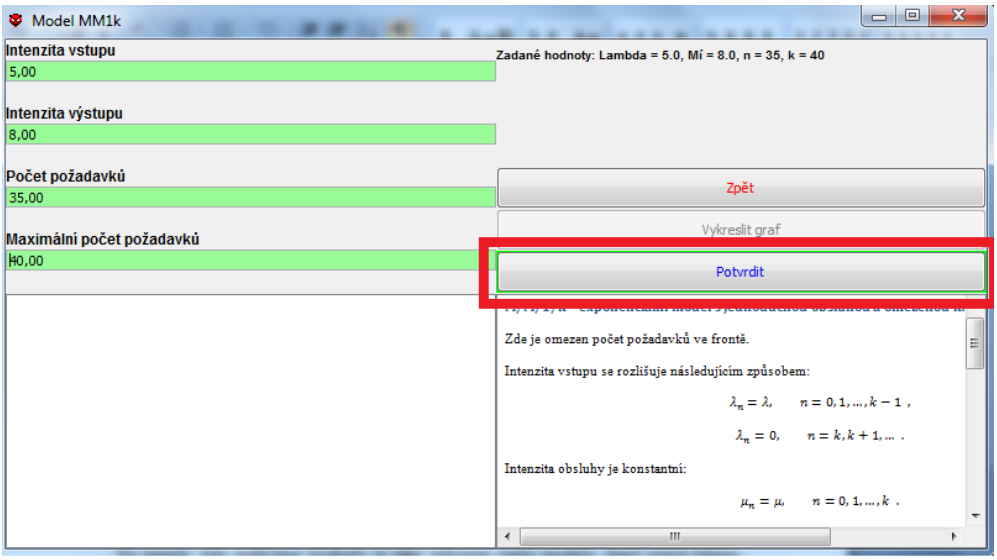
Obrázek 15 – Správné zadání vstupů

Na panelu, kde zadáváme hodnoty je také zobrazen popis modelu, který právě řešíme. V tomto popisu se můžeme zorientovat v problematice daného modelu. Vidíme zde jak jeho stručný popis, tak vzorce, podle kterých se počítají charakteristiky zvoleného modelu (zvýrazněná oblast červeným rámečkem na obrázku 16).



Obrázek 16 – Zobrazení popisu modelu v okně pro vstupy a výstupy

Poté, co jsme zadali všechny potřebné hodnoty a všechna pole, která jsme měli vyplnit, zezelenala, se nám zpřístupní tlačítko „*Potvrdit*“ (zvýrazněná oblast červeným rámečkem na obrázku 17).



Obrázek 17 – Aktivace tlačítka pro provedení výpočtu

Po kliknutí na tlačítko „*Potvrdit*“ se spustí jedna z výpočetních tříd „MM1“, „MM1k“, „MMm“, „Netrpelivost“, „EOQ“, „POQ“, „EOQPN“ nebo „POQPN“, v závislosti na námi vybraném modelu. Tyto výpočetní třídy slouží ke zpracování zadaných údajů a k vypočtení všech charakteristik zvoleného modelu (ukázka výpočtu charakteristik u modelu M/M/1/k je na následujícím obrázku 18).

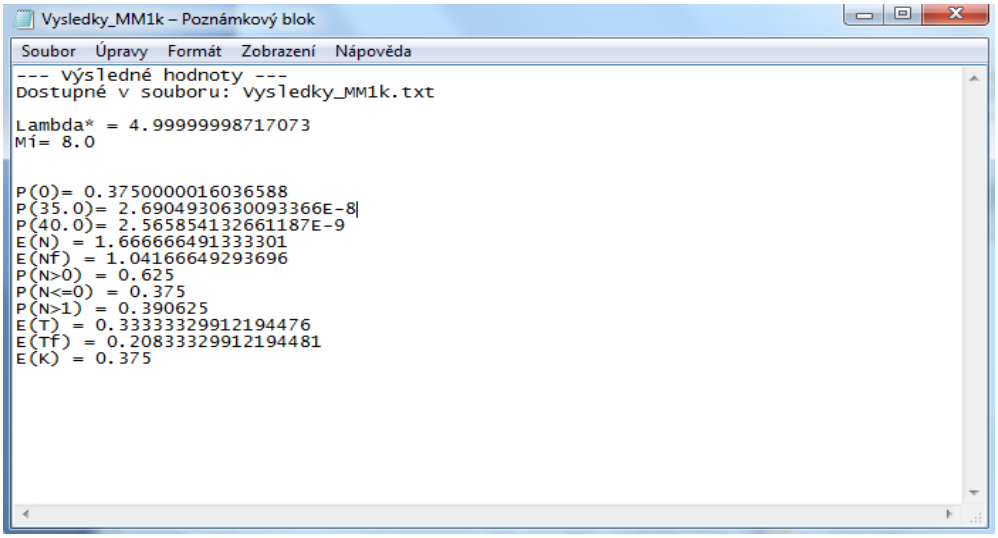
```
double lambdaPrumerna = this.lambda*(1-Pk);
double PrumernaDobaPozadavkuVSHO = PrumernyPocetPozadavkuVSystemu/lambdaPrumerna;
double PrumernaDobaPozadavkuVeFronte = PrumernyPocetPozadavkuFronta/lambdaPrumerna;
double PravdepodobnostCekaniPozadavku = this.ro;
double PozadavekNebudeCekat = 1 - this.ro;
double PrumernyPocetVolnychObsluzMist = 1 - this.ro;
double PravdepodobnostVznikuFronty = Math.pow(this.ro,2);
```

Obrázek 18 – Ukázka výpočtu některých charakteristik

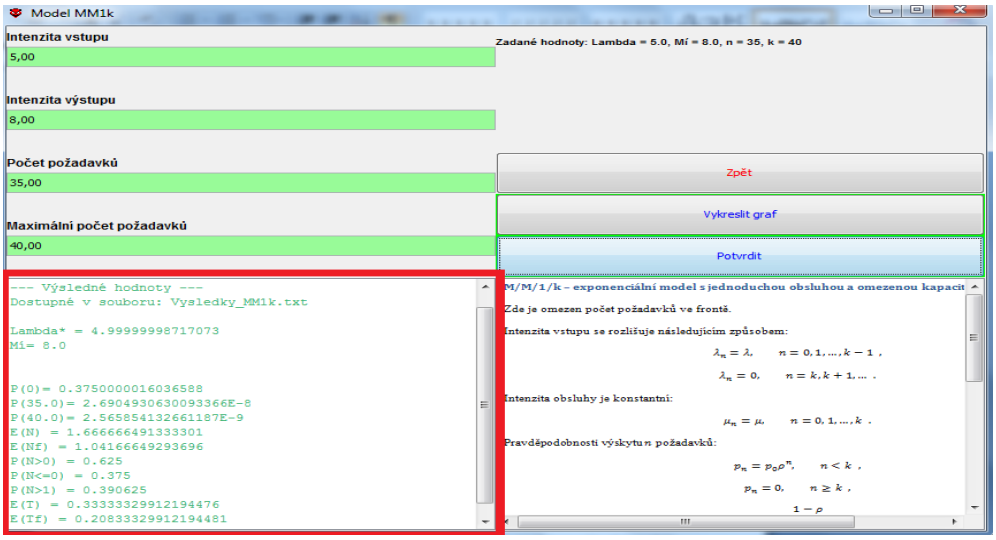
Vypočtené charakteristiky se zapíší do textového souboru (obrázek 19 a 20), ze kterého následně třída, která danou výpočetní třídu spustila, ony charakteristiky načte do panelu zvoleného pro zobrazování výsledků a v tu chvíli je máme před očima (zvýrazněná oblast na obrázku 21).

```
vystup.println("Lambda* = "+lambdaPrumerna);
vystup.println("Mí= "+this.mi);
vystup.println();vystup.println();
vystup.println("P(0)= "+P0);
vystup.println("P("+n+")= "+Pn);
vystup.println("P("+k+")= "+Pk);
vystup.println("E(N) = "+PrumernyPocetPozadavkuVSystemu);
vystup.println("E(Nf) = "+PrumernyPocetPozadavkuFronta);
vystup.println("P(N>0) = "+PravdepodobnostCekaniPozadavku);
vystup.println("P(N<=0) = "+PozadavekNebudeCekat);
vystup.println("P(N>1) = "+PravdepodobnostVznikuFronty);
vystup.println("E(T) = "+PrumernaDobaPozadavkuVSHO);
vystup.println("E(Tf) = "+PrumernaDobaPozadavkuVeFronte);
vystup.println("E(K) = "+PrumernyPocetVolnychObsluzMist);
```

Obrázek 19 – Tisk vypočtených charakteristik do souboru



Obrázek 20 – Vypočtené charakteristiky uložené v textovém souboru



Obrázek 21 – Zobrazení výsledků modelu

Na panelu pro výsledky nám program zobrazí mimo vypočtených charakteristik modelu také název textového souboru, ve kterém je najdeme uloženy (po každém novém proběhnutí výpočtu je soubor přepisován, a proto pokud chceme tyto výsledky zachovat, je třeba soubor někam uložit pod jiným názvem). Výpočetní třída provádí ještě jednu důležitou činnost – plní dvourozměrné pole hodnotami (obrázek 22), které jsou pak využity v grafu (toto platí pro všechny modely, kromě modelů „POQPN“ a „EOQPN“, jejichž grafy jsou z důvodu vykreslení funkce dvou proměnných

třírozměrné. Tyto grafy budeme probírat až v pozdějším textu). Následně pomocí getteru získáme toto pole do naší třídy, která spustila zvolenou výpočetní třídu (spouštějící třída „MMXx“ nebo „XOQ“) a pak je toto pole vloženo jako parametr pro třídu vykreslující graf.

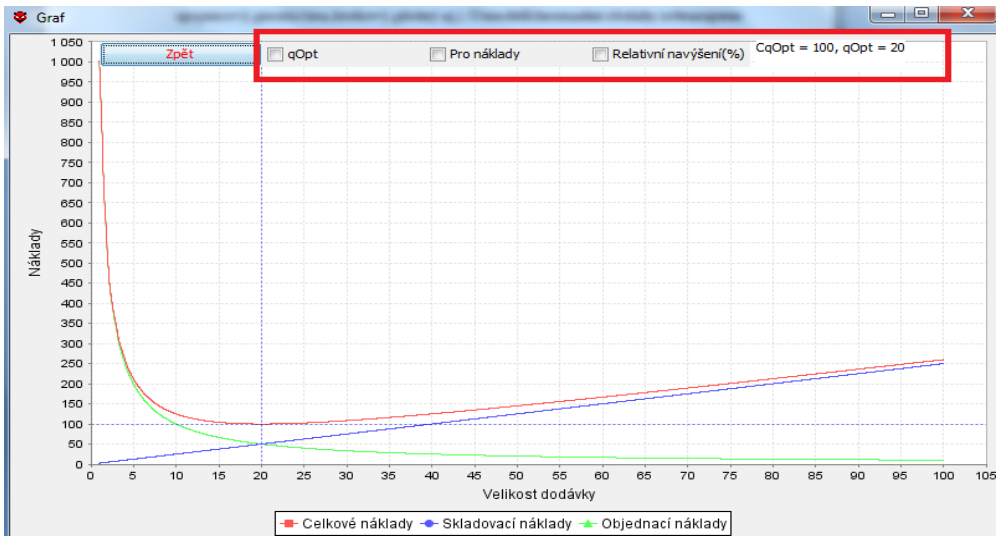
```
//hodnoty pro vytvoreni grafu ukladane do dvourozmerneho pole
double Cqq=0;
double C3q=0;
double C1q=0;
for (int i = 0; i <= 1; i++) {
    for (int j = 0; j < q; j++) {
        if(i==0){
            pole[i][j]= j+1;
            poleC3[i][j]= j+1;
            poleC1[i][j]= j+1;
        }
        else{
            Cqq = (this.c3*this.Q/((j+1)*this.T)) + (this.c1*(j+1))/2;
            pole[i][j]= Cqq;
            C3q = (this.c3*this.Q/((j+1)*this.T));
            poleC3[i][j]= C3q;
            C1q = (this.c1*(j+1))/2;
            poleC1[i][j]= C1q;
        }
    }
}
```

Obrázek 22 – Plnění pole potřebnými hodnotami pro vykreslení nákladů do grafu

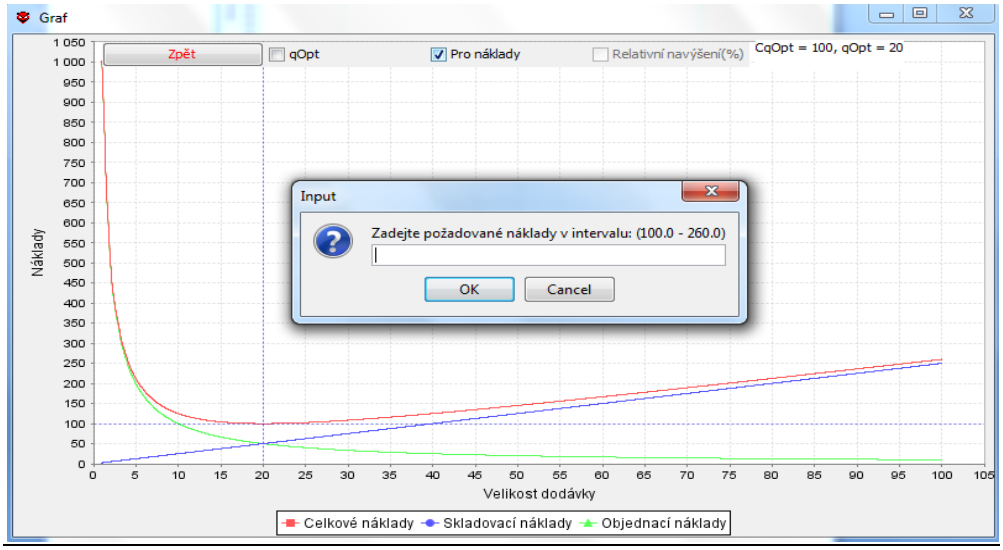
Pro vykreslení grafu je třeba zmáčknout tlačítko „*Vykreslit graf*“, které se zpřístupní vždy po realizaci výpočtu.

11.2 Nástroj pro vykreslení 2D grafů - JFreeChart

Pro vykreslování dvourozměrného grafu je užito nástroje JFreeChart (užívají ho třídy „GrafMMX“ a „GrafXOQ“). Tento nástroj potřebuje jako vstup dvourozměrné pole. Práce s ním je jednoduchá a velice rychlá. Veškeré práce, co se týče rozměrů os a podobně, obstará za nás. Graf lze ukládat ve formě obrázku, lze ho přibližovat a zobrazuje vodící čáry při kliknutí na bod na grafu. Data můžeme zobrazovat v různých typech grafu (bodový, spojnicový, prostá čára, krokový, plošný aj.). U modelů hromadné obsluhy zobrazujeme pravděpodobnosti přítomnosti různého počtu požadavků v systému (příloha A – příloha E). U modelu s netrpělivostí nám graf zobrazí průběh funkce zvolené netrpělivosti (příloha F). Pro modely teorie zásob zobrazujeme průběhy nákladů v závislosti na velikosti dodávky (příloha G – příloha J). U těchto modelů je graf rozšířen o možnost vykreslení velikosti dodávky a nákladů při zvolených nákladech. Náklady můžeme volit buď procentuelně nebo přímo pevnou částkou (Zvýrazněná oblast v obrázku 23). Program nám sám vygeneruje meze, ve kterých je třeba náklady zadat (obrázek 24). Poté, co takto náklady zadáme, se nám zobrazí požadovaný bod a program nám vypíše i přesné hodnoty velikosti dodávky pro zvolené náklady. Celou dobu program zobrazuje nejen graficky, ale i číselně, hodnotu optimální velikosti dodávky a s tím optimální náklady (vlastní rozšíření pro JFreeChart).



Obrázek 23 – Volitelné náklady v grafu



Obrázek 24 – Zobrazení velikosti dodávky podle zvolených nákladů

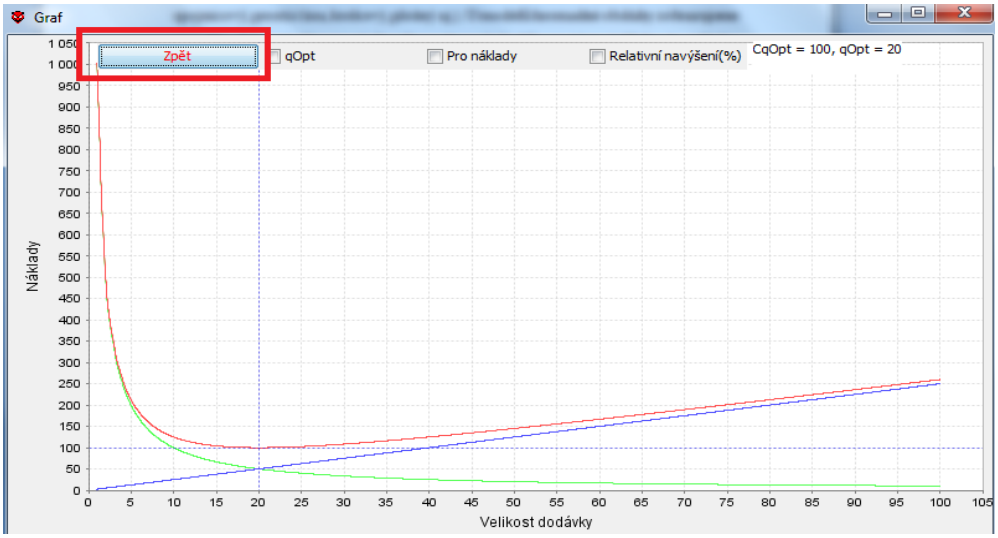
11.3 Nástroj pro vykreslení 3D grafů - SurfacePlotter

Nyní se přesuneme ke třírozměrným grafům zmiňovaných tříd „EOQPN“ a „POQPN“. Jelikož u modelů s přechodným nedostatkem jsou náklady funkcí dvou proměnných, musel být pro vykreslení hodnot použit 3D graf. V nabídce je spousta nástrojů, některé jednoduché, ale pomalé a málo detailní a některé jsou moc náročné. Nakonec jsem vybral nástroj SurfacePlotter (užívá ho třída „Graf3DD“). U něj je slabinou, že k němu neexistuje moc informací, a proto se s ním musí člověk vypořádat sám. Důležitým aspektem bylo, že tento graf má zabudovaný barevný model, který odlišuje různě vysoké hodnoty funkce, čímž zpřehledňuje vyobrazení dat. Vstupní data pro třídu „Graf3DD“ jsou veškeré hodnoty potřebné pro výpočet funkce nákladů.

SurfacePlotter umí vykreslit dvě funkce. První funkcí, kterou vykreslujeme je funkce celkových nákladů a druhou funkcí je konstanta na úrovni optimálních nákladů (plocha na úrovni optimálních nákladů). SurfacePlotter také dokáže zobrazit graf v různých barevných variantách, jde s ním otáčet, posunovat, zvětšovat a můžeme ho i uložit ve formě obrázku. Množství nastavení je k dispozici přímo v zobrazeném okně grafu. Zmiňované dvě funkce vykreslí buď zvlášť, nebo dohromady (příloha O). Třída „Graf3DD“ využívá třídu „ProgressiveSurfaceModel“, která kalkuluje popisky os, krokování výpočtu funkce a podobně.

11.4 Uzavírání aplikace

Ve všech oknech, která se nám zobrazují, se nachází tlačítko „*zpět*“ (zvýrazněná oblast na obrázku 25). Pokud nechceme ukončovat program a chceme se pouze vracet na předchozí nabídky, je třeba využít toto tlačítko. Pokud zmáčkneme křížek, celý program skončí.



Obrázek 25 – Tlačítko „*zpět*“

11.5 Programový kód – názvy proměnných a komponent

Veškeré názvy komponent a proměnných v programu jsem volil v návaznosti na značení, které se používá u modelů hromadné obsluhy, respektive u modelů teorie zásob podle [10] a [11].

Názvy pro vstupní pole, kde uživatel zadává jednotlivé hodnoty, jsou značeny písmeny, odpovídajícími značení v praxi. Následné číslo, které z pole bude získáno, je stejného značení, ovšem psáno velkým písmem (obrázek 26 a obrázek 27). U značení, která jsou již velkým písmem v reálném modelu nebo by se při převodu na velká písmena shodovali s jiným značením, se znaky zdvojují (příklad: T a t, tedy celková doba řízení skladu a délka jednoho cyklu → textové pole pro vstup bude mít název T resp. t) a konkrétní číslo z textového pole získané bude mít značení TT resp. tt)).

```
private JFormattedTextField Q;
private JFormattedTextField q;
private JFormattedTextField T;
private JFormattedTextField t;
private JFormattedTextField t1;
```

Obrázek 26 – Názvy polí pro vstupy

```
private int QQ;
private int qq;
private double TT;
private double tt;
private double T1;
```

Obrázek 27 – Názvy čísel získaných ze vstupních polí

Značení jednotlivých komponent grafického rozhraní je rozlišeno čísly od prvních, co jsou vkládány, po poslední. V případě některých zaklikávacích (JCheckBox) komponent nebo textových návěstí (JLabel) a tlačítek (JButton), u kterých to bylo možné, jsou názvy odvozovány od zobrazovaného textu nebo od jejich funkce (obrázek 28).

```
private JTextArea vysledky;
private JButton provestVypocet;
private JButton graf;
private JLabel zadaneHodnoty;
```

Obrázek 28 – Názvy některých komponent

Podkomponenty se řídí názvem komponenty, na které jsou umístěny a jsou rozlišeny posledním číslem (příklad: panel3 a jeho podkomponenta panel31 atd. – obrázek 29).

```
private JPanel panel3;
private JPanel panel31;
private JPanel panel32;
```

Obrázek 29 – Názvy komponent a jejich podkomponent

Datové typy jsem volil podle charakteru dané proměnné modelu. Například dodávky zboží můžou být pouze celočíselné (jednotku zboží nemůžeme dělit) a naproti tomu čas může být zadán v desetinném tvaru. V tomto smyslu jsem se rozhodoval mezi užitím typů double a integer.

V kódu se vyskytují zakomentované části. Ty jsou tam z důvodu případného dalšího rozšiřování programu a mohou sloužit jako výpomoc. Ostatní zakomentované části se týkají popisu proměnných nebo metod. Vyskytují se zde i některé nevyužité metody. Ty mohou sloužit k jinému způsobu provádění některých činností (např. metoda pro načítání obrázků – můžeme načítat více způsoby. Mohou dobře sloužit při rozšiřování programu).

12 Modelové příklady

12.1 Model M/M/1

Příklad 1a:

Vycházíme z příkladu [11, str. 76 - 78]. Mějme restaurační zařízení. Dostatečné místo pro zákazníky stojící ve frontě a jednočlennou obsluhu. Na základě pozorování byly určeny empirické četnosti v jednominutových intervalech, kterých bylo celkem 25. Dále bylo potřeba otestovat, zda se vstup požadavků řídí poissonovým rozdělením. Poté jsme váženým průměrem získali intenzitu vstupu na hodnotě 3.

Intenzitu výstupu zjistíme z pozorování, které říká, že průměrně každých 15 sekund je obsloužen jeden zákazník. Jelikož pracujeme s minutovými intervaly, pak je intenzita výstupu 4.

$$\lambda = 3, \mu = 4, \rho = 0.75 .$$

Podle vzorců v kapitole 2 vypočteme:

$$p_0 = 0.25, p(5) = 0.59326171, E(N) = 3, E(Nf) = 2.25 ,$$

$$E(K) = 0.25, E(T) = 1, E(Tf) = 0.75, P(N > 0) = 0.75 .$$

Výstup programu nalezneme v příloze A.

Příklad 1b:

Řekněme, že majitel restaurace nechá nainstalovat nové dotykové obrazovky sloužící k zaznamenávání objednávek zákazníků a tím zrychlí obsluhu. Obsluha už neobslouží průměrně zákazníka za 15 sekund, ale zkrátí obslužnou dobu na 10 sekund.

$$\lambda = 3, \mu = 6, \rho = 0.75 .$$

Výsledné hodnoty 2. varianty:

$$p_0 = 0.5, p(5) = 0.015625, E(N) = 1, E(Nf) = 0.5 ,$$

$$E(K) = 0.5, E(T) = 0.3333, E(Tf) = 0.166666, P(N > 0) = 0.5 .$$

Zrychlení obsluhy přineslo o 25 % vyšší pravděpodobnost, že obsluha nebude vyřizovat žádnou objednávku. Průměrná doba čekání zákazníka na objednávku se zkrátila

ze 45 sekund přibližně na 10 sekund (varianta a: $E(Tf) = 0.75$, varianta b: $E(Tf) = 0.166666$). Průměrný počet požadavků čekajících na obsloužení se snížil (varianta a: $E(Nf) = 2.25$, varianta b: $E(Nf) = 0.5$).

Výstup programu nalezneme v příloze B.

12.2 Model M/M/1/k

Příklad 2a:

Systém, kde se může vyskytnout maximálně *k* požadavků. Tomuto systému by mohl odpovídat kadeřnický salón s jedním kadeřníkem. Má malý salónek, kde můžou čekat maximálně tři lidé a střihá pouze on. To znamená, že v systému se mohou objevit maximálně čtyři požadavky (model M/M/1/4). Vypozorovali jsme, že průměrně každou hodinu přijdou 2 zákazníci. Kadeřník je schopen ostřihat v průměru jednoho zákazníka za 25 minut.

Vstupní údaje:

$$\lambda = 2, \mu = 2.4\,.$$

Výsledné hodnoty podle vzorců z kapitoly 4 jsou následující:

$$p_0 = 0.2786497, p(4) = 0.1343797, p(10) = 0, E(N) = 1.6405, E(Nf) = 0.91916\,,$$

$$E(K) = 0.1666, E(T) = 0.94759, E(Tf) = 0.530923, P(N > 0) = 0.8333\,.$$

Výstup programu nalezneme v příloze C.

Příklad 2b:

Kadeřník investoval a rozšířil čekárnu o dvě sedadla pro čekání. Změnila se nám tedy kapacita kadeřnictví (M/M/1/6), intenzity vstupu a obsluhy se nemění.

Výsledné charakteristiky jsou následující:

$$p_0 = 0.231186, p(6) = 0.07742, p(10) = 0, E(N) = 2.29016, E(Nf) = 1.52135\,,$$

$$E(K) = 0.1666, E(T) = 1.24117, E(Tf) = 0.82451, P(N > 0) = 0.8333\,.$$

Výstup programu nalezneme v příloze D.

12.3 Model M/M/m

Příklad 3a:

Mějme obchodní dům, kde je k dispozici 6 pokladen. Vedení firmy provozující obchodní dům chce vědět, jaké jsou charakteristiky stavu systému v závislosti na počtu pokladen v provozu, aby se na jejich základě mohla rozhodnout pro počty spuštěných pokladen v jednotlivých časech provozu. Z pozorování byly zjištěny dvě různé intenzity vstupů – v hodinách 8:00 – 13:30 je intenzita vstupu odpozorována na hodnotě 1 zákazník za 40 sekund. V hodinách 13:30 – 19:00 byla intenzita odpozorována na 1 zákazníka za 15 sekund. Jedna pokladna je schopna odbavit jednoho zákazníka průměrně za 1 minutu (výstupy programu nalezneme v příloze E).

a) Vstupní údaje (čas 8:00 - 13:30):

Intenzity jsou uvedeny pro dvouminutové intervaly.

$$\lambda = 3, \mu = 2\,.$$

Již ze zjištěných intenzit vidíme, že je třeba, aby v čase 8:00 až 13:30 byly v provozu alespoň dvě pokladny pro dodržení podmínky stability systému.

Výsledné charakteristiky podle vzorců z kapitoly 3:

$$p_0 = 0.16666, p(10) = 0.018771, p(20) = 0.001057, E(N) = 3.75,$$

$$E(Nf) = 2.25\,, E(K) = 0.5, E(T) = 1.25, E(Tf) = 0.75,$$

$$P(N > 0) = 0.75\,.$$

b) Vstupní údaje (čas 13:30 – 19:00):

Intenzity opět uvedeny pro dvouminutové intervaly.

$$\lambda = 8, \mu = 2\,.$$

Zde vidíme, že v odpoledních hodinách je třeba, aby bylo v provozu alespoň pět pokladen. Podle vzorců z kapitoly 3 vypočteme následující charakteristiky:

$$p_0 = 0.01874, p(10) = 0.05242, p(20) = 0.00562, E(N) = 7.1999,$$

$$E(Nf) = 3.1999\,, E(K) = 0.9999, E(T) = 0.8999, E(Tf) = 0.3999,$$

$$P(N > 0) = 0.7999\,.$$

Příklad 3b:

Pokračujme s předchozí situací, kdy máme obchodní dům. Nyní ovšem budeme řešit problém v případě existence apriorní netrpělivosti zákazníků v obchodě. Model budeme počítat pro dopolední čas v obchodním domě (8:00 - 13:30). Pro zadání netrpělivosti jsou důležité další tři parametry. Počátek projevování se netrpělivosti n_1, konečná hodnota intenzity vstupu $\lambda(n_2)$ a n_2, kdy intenzita vstupu spadne na konečnou hodnotu. Řekněme, že netrpělivost se začne projevovat od počtu 15 zákazníků u pokladen a padne na konečnou úroveň při 22 zákaznících u pokladen. Po dosažení n_2 je intenzita vstupu rovna nule.

Vstupní údaje:

$$\lambda = 3, \mu = 2, n_1 = 15, n_2 = 22, \lambda(n_2) = 0.5\,.$$

Příklad vyřešíme stejným způsobem jako předchozí, v programu si však zatrhneme políčko „*Uvažovat netrpělivost"* a program nám v tu chvíli vypočte funkci netrpělivosti a vykreslí ji do grafu, který se nám zobrazí po kliknutí na tlačítko „*Vykreslit graf*".

Programový výstup nalezneme v příloze F.

12.4 Model EOQ

Příklad 4a:

Máme za úkol řídit sklad s co nejnižšími možnými celkovými náklady na jeden cyklus. Časovým horizontem je jeden rok, tedy 12 měsíců. Za tuto dobu je celková potřeba zboží 1200 ks. Dodávky přichází měsíčně o velikosti 100 ks. Jeden cyklus tedy trvá jeden měsíc. Jednotkové skladovací náklady jsou 5 peněžních jednotek a náklady na jednu objednávku jsou 10 peněžních jednotek. Vstupní hodnoty:

$$Q = 1200, q = 100, T = 12, t = 1, c_1 = 5, c_3 = 10, n = 12\,.$$

Podle vzorců z kapitoly 7 vypočteme následující hodnoty:

$$N(q) = 260, N_T(q) = 3120, C(q) = 260, q_{opt} = 20, t_{opt} = 0.2, n_{opt} = 60,$$

$$C(q_{opt}) = 100, N_T(q_{opt}) = 1200, q^* = 10\,.$$

Z výsledků vyplývá, že pokud budeme objednávat zboží ve výši 20 ks ($q_{opt} = 20$) jednou za 6 dní ($t_{opt} = 0.2$) při daných nákladech c_1, c_3, budou náklady na jeden cyklus minimální ($C(q_{opt}) = 100$).

Výstup programu se nalézá v příloze G.

Příklad 4b:

Pojďme se podívat, jak se změní situace v případě, že se nám z důvodu zdražení pronájmu zvýší jednotkové skladovací náklady o tři peněžní jednotky. Také dodavatel zvýšil cenu objednávek o jednu peněžní jednotku, a tak je výchozí situace následující:

$$Q = 1200, q = 100, T = 12, t = 1, c_1 = 8, c_3 = 11, n = 12\,.$$

Změny můžeme sledovat na výsledných hodnotách:

$$N(q) = 411, N_T(q) = 4932, C(q) = 411, q_{opt} = 17, t_{opt} = 0.1658, n_{opt} = 72,$$

$$C(q_{opt}) = 132.665, N_T(q_{opt}) = 1591.98, q^* = 8\,.$$

Po zdražení je pro hospodaření s minimálními náklady na jeden cyklus nutné objednávat ve velikosti 17 ks ($q_{opt} = 17$) jednou za 5 dní ($t_{opt} = 0.166$).

V případě, že předpokládáme zpoždění dodávek 1 den ($t_p = 0.0333$), vypočteme signální úroveň hladiny zboží na skladě, při které máme objednávat. Při tomto zpoždění nám vyjde $s = 14$, což znamená, že po odčerpání 14 ks zboží ze skladu, je třeba zadat novou objednávku.

V příkladě 4a by při stejném zpoždění bylo $s = 17$.

Pokud se podíváme do grafu v přiloženém programu pro vypočtený model, máme možnost zde volit svoje náklady (procentní či absolutní změnou) a program nám zobrazí graficky i číselně hodnoty objednávaného množství pro námi zvolené náklady.

Výstup programu nalezneme v příloze H.

12.5 Model POQ

Příklad 5a:

Budeme vycházet ze stejného zadání jako u modelu EOQ v příkladech 4a a 4b. Můžeme sledovat případné odlišnosti výsledků, které vygeneruje model POQ v porovnání s výsledky modelu EOQ. Výstup programu je v příloze I.

Vstupní hodnoty:

$$Q = 1200, q = 100, T = 12, t = 1, t_1 = 0.0666\ (2\ dny), c_1 = 5, c_3 = 10, n = 12\ .$$

Podle vzorců z kapitoly 8 vypočteme následující hodnoty:

$$N(q) = 243.33, N_T(q) = 2920, C(q) = 243.33, q_{opt} = 21, t_{opt} = 0.193,$$

$$n_{opt} = 58, C(q_{opt}) = 96.61, N_T(q_{opt}) = 1159.3, q^* = 10\ .$$

Z výsledků vyplývá, že pokud budeme objednávat zboží ve výši 21 ks ($q_{opt} = 21$) jednou za 6 dní ($t_{opt} = 0.19$) při daných nákladech, budou náklady na jeden cyklus minimální ($C(q_{opt}) = 96.61$).

Porovnáme-li výsledek modelu POQ s výsledky modelu EOQ, optimální velikost dodávky je u POQ vyšší a celkové náklady na jeden cyklus, který je teoreticky o něco málo kratší než u EOQ, jsou nižší.

Příklad 5b:

Otestujeme, jak si model POQ stojí po navýšení nákladů. Předpokládáme stejné změny jako v příkladě 4b u modelu EOQ. Výstup programu je v příloze J.

$$Q = 1200, q = 100, T = 12, t = 1, t_1 = 0.0666\ (2\ dny), c_1 = 8, c_3 = 11, n = 12\ .$$

Změny můžeme sledovat na výsledných hodnotách:

$$N(q) = 384, N_T(q) = 4612, C(q) = 384, q_{opt} = 17, t_{opt} = 0.1602, n_{opt} = 70,$$

$$C(q_{opt}) = 128.2, N_T(q_{opt}) = 1537.9, q^* = 9\ .$$

Po zdražení je pro hospodaření s minimálními náklady na jeden cyklus nutné objednávat ve velikosti 17 ks ($q_{opt} = 17$) jednou za necelých 5 dní ($t_{opt} = 0.1602$).

12.6 Model EOQ s možností přechodného nedostatku

Příklad 6a:

Dále budeme modifikovat předchozí příklad a to na model EOQ s možností přechodného nedostatku. V praxi přechodný nedostatek znamená, že je z předchozího cyklu převedena část poptávky do cyklu dalšího, jelikož už v předchozím cyklu tato poptávka nemohla být uspokojena. Se zavedením tohoto předpokladu nám do modelu vstupují dva nové parametry. Jednak jsou to ztráty z nedostatku, které vyjadřujeme v jednotkovém tvaru (c_2) a pak je to disponibilní množství zboží na skladě po odčerpání neuspokojené poptávky předchozího období (S). Jednotkovou ztrátu zvolíme 20 peněžních jednotek a disponibilní množství máme určeno jako 90% z velikosti dodávky.

Vstupní hodnoty:

$$Q = 1200, q = 100, T = 12, t = 1, c_1 = 5, c_3 = 10, c_2 = 20, n = 12, S = 90\,.$$

Podle vzorců z kapitoly 7.1 vypočítáme:

$$N(S,q) = 222.5, N_T(S,q) = 2670, C(S,q) = 222.5, q_{opt} = 22, t_{opt} = 0.2236,$$

$$n_{opt} = 54, C(S_{opt}, q_{opt}) = 89.44, N_T(S_{opt}, q_{opt}) = 1073.3, S_{opt} = 18\,.$$

Pokud srovnáme výsledky s jednoduchým modelem EOQ, vidíme, že zde dosahujeme nižších hodnot celkových nákladů. V následujícím příkladě zvedneme ztráty z nedostatku na 30 peněžních jednotek. Výstupy programu jsou v příloze K a příloze L.

Příklad 6b:

Vstupní hodnoty:

$$Q = 1200, q = 100, T = 12, t = 1, c_1 = 5, c_3 = 10, c_2 = 30, n = 12, S = 90\,.$$

Výsledné hodnoty modelu:

$$N(S,q) = 227.5, N_T(S,q) = 2730, C(S,q) = 227.5, q_{opt} = 22, t_{opt} = 0.216,$$

$$n_{opt} = 56, C(S_{opt}, q_{opt}) = 92.6, N_T(S_{opt}, q_{opt}) = 1111, S_{opt} = 19\,.$$

Po zvýšení jednotkových nákladů z nedostatku se optimální hodnota S_{opt} podle předpokladů zvedla, jelikož se nám přechodný nedostatek prodražil.

12.7 Model POQ s možností přechodného nedostatku

Příklad 7a:

V příkladu pro model POQ s přechodným nedostatkem použijeme již známé vstupní hodnoty z předchozích, abychom mohli dobře srovnávat výsledky jednotlivých variant. U tohoto modelu provádíme minimalizaci funkce nákladů přes proměnné S a t z důvodu ilustrace minimalizace funkce přes jinou proměnnou než q. Velikost S počítáme přes zadávanou hodnotu $|-s|$, která znamená maximální možný nedostatek zboží během cyklu. Výstupy programu nalezneme v příloze M a příloze N.

Vstupní hodnoty:

$$Q = 1200, q = 100, T = 12, t = 1, c_1 = 5, c_3 = 10, c_2 = 20, n = 12, |-s| = 10.$$

Podle vzorců z kapitoly 8.1 vypočítáme:

$$N(t,S) = 228.5, N_T(t,S) = 2742.4, C(t,S) = 228.5, q_{opt} = 23, t_{opt} = 0.2317,$$

$$n_{opt} = 52, C(t_{opt}, S_{opt}) = 86.3, N_T(t_{opt}, S_{opt}) = 1035.64, S_{opt} = 17\,.$$

Přes optimální délku cyklu jsme dokázali výsledné minimální náklady ještě snížit oproti modelu bez nedostatku. Tím, že zkrátíme délku cyklu z jednoho měsíce na 7 dní ($t_{opt} = 0.2317$), počet cyklů navýšíme na 52 a velikost dodávky bude tedy 23 ks a množství $S_{opt} = 17$, budeme minimalizovat celkové náklady.

Příklad 7b:

Prověříme, co se stane, zvýšíme-li náklady z nedostatku na 30 peněžních jednotek.

Vstupní hodnoty:

$$Q = 1200, q = 100, T = 12, t = 1, c_1 = 5, c_3 = 10, c_2 = 30, n = 12, |-s| = 10.$$

Výsledné hodnoty:

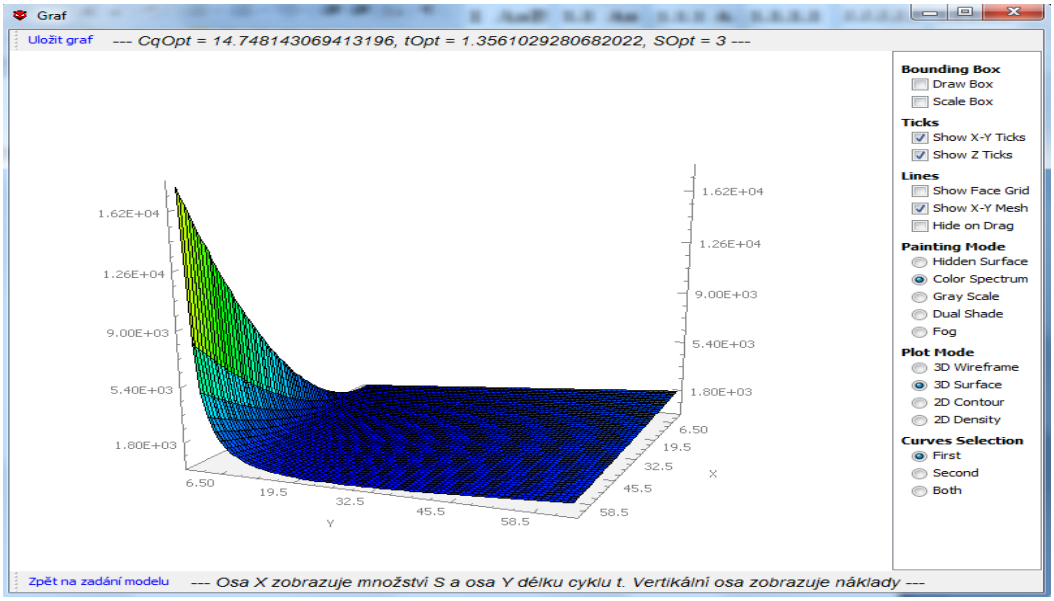
$$N(t,S) = 229.1, N_T(t,S) = 2748.6, C(t,S) = 229.1, q_{opt} = 22, t_{opt} = 0.224,$$

$$n_{opt} = 54, C(t_{opt}, S_{opt}) = 89.33, N_T(t_{opt}, S_{opt}) = 1072, S_{opt} = 18\,.$$

Povolený přechodný nedostatek se snížil o jeden kus a celkové náklady stouply o 36 peněžních jednotek oproti předchozí variantě.

Příklad 8:

V předchozích příkladech pro model POQ s přechodným nedostatkem jsme uvažovali takové hodnoty, aby bylo možné porovnávat výsledky s ostatními modely. Nyní si zadáme takové hodnoty, abychom na grafu lépe viděli minimální náklady. Na následujícím obrázku (Obrázek 30) je graf, u kterého jsme volili takové vstupy, aby měla funkce viditelné sedlo, ze kterého lépe uvidíme minimum funkce. Vstupy jsme pozměnili tak, že za T jsme dosadili 365 a t je v tu chvíli tedy $\frac{365}{12}$. Dobu současného čerpání a výroby jsme zvolili na hodnotě 7. Ostatní hodnoty jsme zachovali z předchozího příkladu 7a. V příloze O jsou zobrazeny obě funkce, které graf umí zobrazit. První funkce je funkcí nákladů, druhá funkce je pouze konstanta na úrovni optimálních nákladů a nakonec je máme zobrazeny obě v jednom obrázku.



Obrázek 30 – Funkce nákladů pro model POQ s přechodným nedostatkem

13 Závěr

Všechny stanovené cíle byly splněny. Bylo vytvořeno programové prostředí pro podporu KEM/PMO v předem vybraných stěžejních oblastech tohoto předmětu a byly vyřešeny modelové příklady. Prostředí Eclipse spolu s jazykem Java byly zvoleny také proto, že tvoří základní platformu vývoje aplikací pro nový trend počítání na chytrých telefonech (mobile computing), především pro systém Android.

Dále bych zmínil některé z možností rozšíření programu, která by se dala realizovat. V první řadě se nabízí možnost zahrnout další modely, ať už z teorie obsluhy nebo teorie zásob, jejichž problematikou se zabývá předmět KEM/PMO. Také by mohla být vytvořena databáze příkladů, která by byla napojena na program, a v programu by se přímo zobrazovalo zadání, případně by zde mohlo být i správné řešení. Je zde velký prostor pro vykreslování různých funkcí, a tak se nabízí zobrazovat i další funkce, například výše nákladů v závislosti na jiných veličinách než na velikosti dodávky nebo délce cyklu, a to například na výši jednotkových nákladů na skladování nebo nákladů na objednávku.

Jsem rád, že jsem si vybral vytvoření programu pro podporu KEM/PMO jako svou diplomovou práci. Osvěžil jsem si programování, ač nevím, zda se moje kariéra bude ubírat právě směrem počítačového programování. Zorientoval jsem se v tvorbě grafického uživatelského rozhraní pomocí java.swing a osvěžil jsem si látku, jíž se zabývá předmět KEM/PMO. Především mě potěšilo seznámení se s grafickými nástroji v Javě, které jsou velkými pomocníky při vykreslování různých funkcí do grafu.

Byl bych rád, kdyby program jakkoli sloužil pro počítačovou podporu zmíněného předmětu a stal se tak užitečnou pomůckou, která vzešla z této diplomové práce.

15 Seznam obrázků

16 Seznam použitých zkratek

SHO – systémy hromadné obsluhy

FIFO – first in, first out

LIFO – last in, first out

PRI – priority

GD – general discipline

SIRO – selection in random order

EOQ – economic order quantity

POQ – production order quantity

17 Použitá literatura a internetové zdroje

Knižní publikace

[1] ANDERSON, D. R. – SWEENEY, P. J. – WILLIAMS, T. A. *An Introduction to Management Science: Quantitative Approaches to Decision Making.* 5th Edition, New York: West Publishing Company, 1988. ISBN 0-314-62969-6

[2] BLOCH, J. *Effective Java.* 2nd Edition, Sant Clara, California: Sun Microsystems, 2008. ISBN 0-321-35668-3

[3] HEROUT, P. *Java: Grafické uživatelské prostředí a čeština.* České Budějovice: Kopp, 2007. ISBN: 978-80-7232-328-9

[4] HEROUT, P. Skripta Předmětu KIV/PPA1. Plzeň: ZČU, fakulta aplikovaných věd, 2007.

[5] HINDLS, R. – HRONOVÁ, S. – SEGER, J. – FISCHER, J. *Statistika pro ekonomy.* 8. vydání, Praha: Proffesional Publishing, 2007. ISBN 978-80-86946-43-6

[6] HUŠEK, R. - LAUBER, J. *Simulační modely*. 1. vydání, Praha : SNTL, 1987.

[7] JABLONSKÝ, J. *Operační výzkum: kvantitativní modely pro ekonomické rozhodování*. 2. vydání Praha: Professional Publishing, 2002. ISBN 80-86419-42-8.

[8] KOŘENÁŘ, V. *Stochastické procesy,* 1. Vydání, Praha: VŠE, 2002. ISBN: 978-80-24516-46-2

[9] LUKÁŠ, L. *Pravděpodobnostní modely v managementu – teorie zásob a statistický popis poptávky*. Praha: ACADEMIA, 2011. ISBN 978-80-200-2005-5.

[10] LUKÁŠ, L. *Pravděpodobnostní modely v managementu.* Praha: Academia, 2009. ISBN 978-80-200-1704-8

[11] LUKÁŠ, L. *Pravděpodobnostní modely.* 1. Vydání, Plzeň: Západočeská univerzita, 2005. ISBN 80-7043-388-4

[12] MACEK, J. - MAINZOVÁ, E. *Základní metody operační analýzy*. Plzeň : ZČU, 1995. ISBN 80-7082-200-7.

[13] PLEVNÝ, M. - ŽIŽKA, M. *Modelování a optimalizace v manažerském rozhodování*. 2. Vydání, Plzeň: ZČU, 2010. 298 s. ISBN 978-80-7043-933-3.

[14] RENDER, B. – STAIR, R. M. – HANNA, M. I. *Quantitative Analysis for Management.* 7th Edition, Upper Saddle River, New Jersey: Prentice Hall, 2000. ISBN 0-13-021538-4

[15] SPELL, B. *Java: Programujeme profesionálně.* Praha: Computer Press, 2002. ISBN 80-7226-667-5

Internetové zdroje

[16] *Administrative Information Servis* [online]. [Cit. 2012-2-20]. Dostupné na WWW: < http://ais.web.cern.ch/ais/apps/lims/EXCEPT~1convert.html>

[17] CIBULKOVÁ, I. *Systémy hromadné obsluhy* [online]. Dostupné na WWW: < http://www.fd.cvut.cz/department/k611/pedagog/K611THO_soubory/studenti_THO/Cibulkova.pdf>

[18] *Drawing of Images in Java* [online]. [cit. 2012-2-12]. Dostupné na WWW: <http://docs.oracle.com/javase/tutorial/2d/images/drawimage.html>

[19] *Graphical Tool: SurfacePlotter* [online]. [cit. 2012-2-15]. Dostupné na WWW: <http://code.google.com/p/surfaceplotter/>

[20] CHMELAŘ, J. *Diplomová práce – teorie hromadné obsluhy* [online]. Dostupné na WWW:< http://www.mti.tul.cz/files/oa/obsluha/index.html>

[21] *Java Tutorial* [online]. [cit. 2012-3-3]. Dostupné na WWW: <http://www.java2s.com/Tutorial/Java/CatalogJava.htm>

[22] *JFreeChart* [online]. [cit. 2012-2-10]. Dostupné na WWW: <http://www.jfree.org/jfreechart/>

[23] *SurfacePlotter* [online]. [cit. 2012-2-15]. Dostupné na WWW: <http://www.butlercc.edu/mathematics/Utilities/surface_plotter_applet/index.html>

[24] ŠEDA, M. *Modely hromadné obsluhy* [online]. Dosupné na WWW: <**h**ttp://web2.vslg.cz/fotogalerie/acta_logistica/2011/2_cislo/3_seda.pdf>

[25] *Youtube: Java Programming Tutorials* [online]. [cit. 2012-4-5]. Dostupné na WWW: <http://www.youtube.com/watch?v=L06uGnF4IpY&feature=related>

Ostatní zdroje

[26] Konzultace k diplomové práci s Doc. RNDr. Ing. Ladislavem Lukášem, CSc.

18 Seznam příloh

Příloha A – Vstupy a výstupy programu pro modelový příklad 1a

Příloha B – Vstupy a výstupy programu pro modelový příklad 1b

Příloha C - Vstupy a výstupy programu pro modelový příklad 2a

Příloha D - Vstupy a výstupy programu pro modelový příklad 2b

Příloha E - Vstupy a výstupy programu pro modelový příklad 3a

Příloha F - Vstupy a výstupy programu pro modelový příklad 3b: zadání a vykreslení netrpělivosti

Příloha G - Vstupy a výstupy programu pro modelový příklad 4a

Příloha H - Vstupy a výstupy programu pro modelový příklad 4b

Příloha I - Vstupy a výstupy programu pro modelový příklad 5a

Příloha J - Vstupy a výstupy programu pro modelový příklad 5b

Příloha K - Vstupy a výstupy programu pro modelový příklad 6a

Příloha L - Vstupy a výstupy programu pro modelový příklad 6b

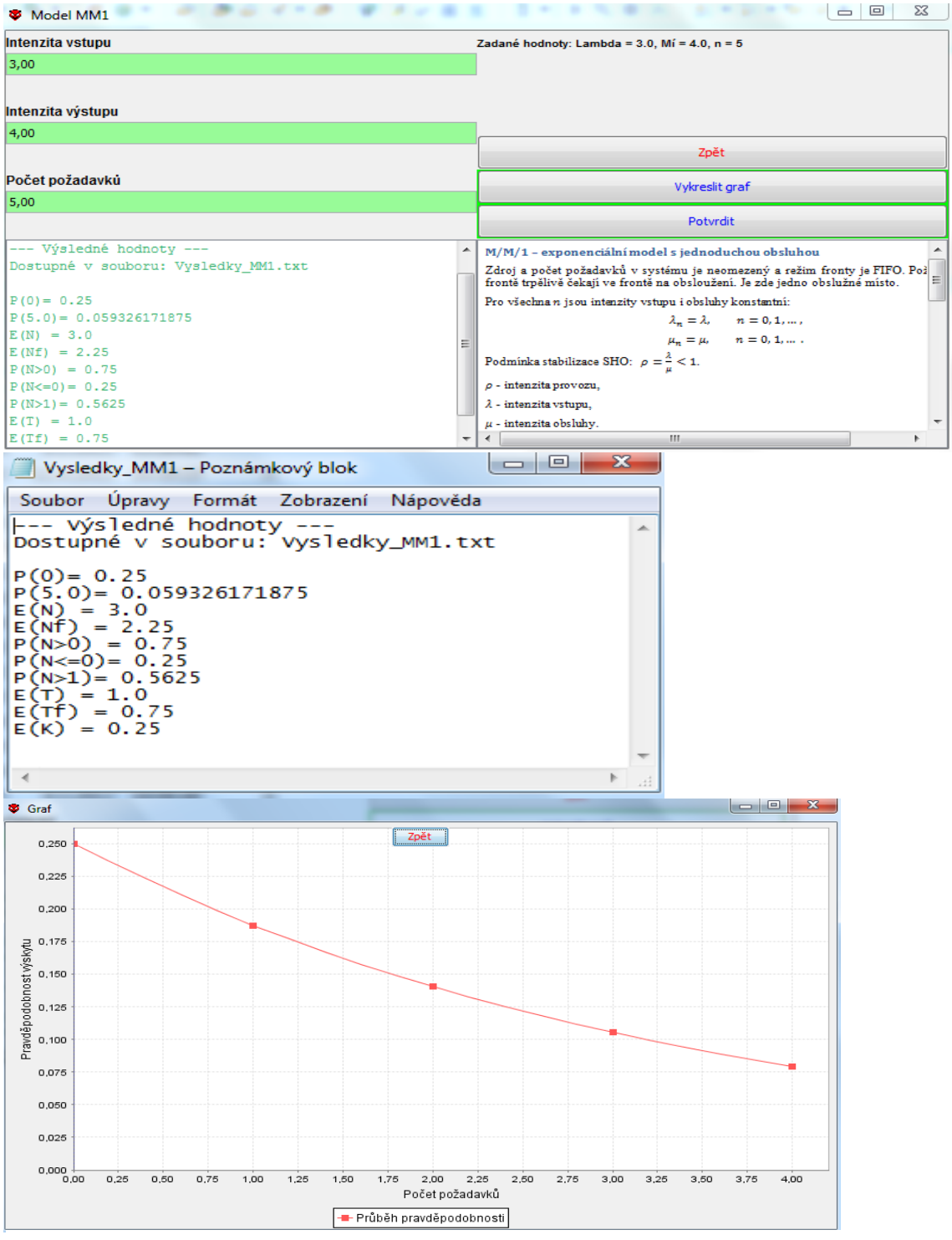
Příloha M - Vstupy a výstupy programu pro modelový příklad 7a

Příloha N - Vstupy a výstupy programu pro modelový příklad 7b

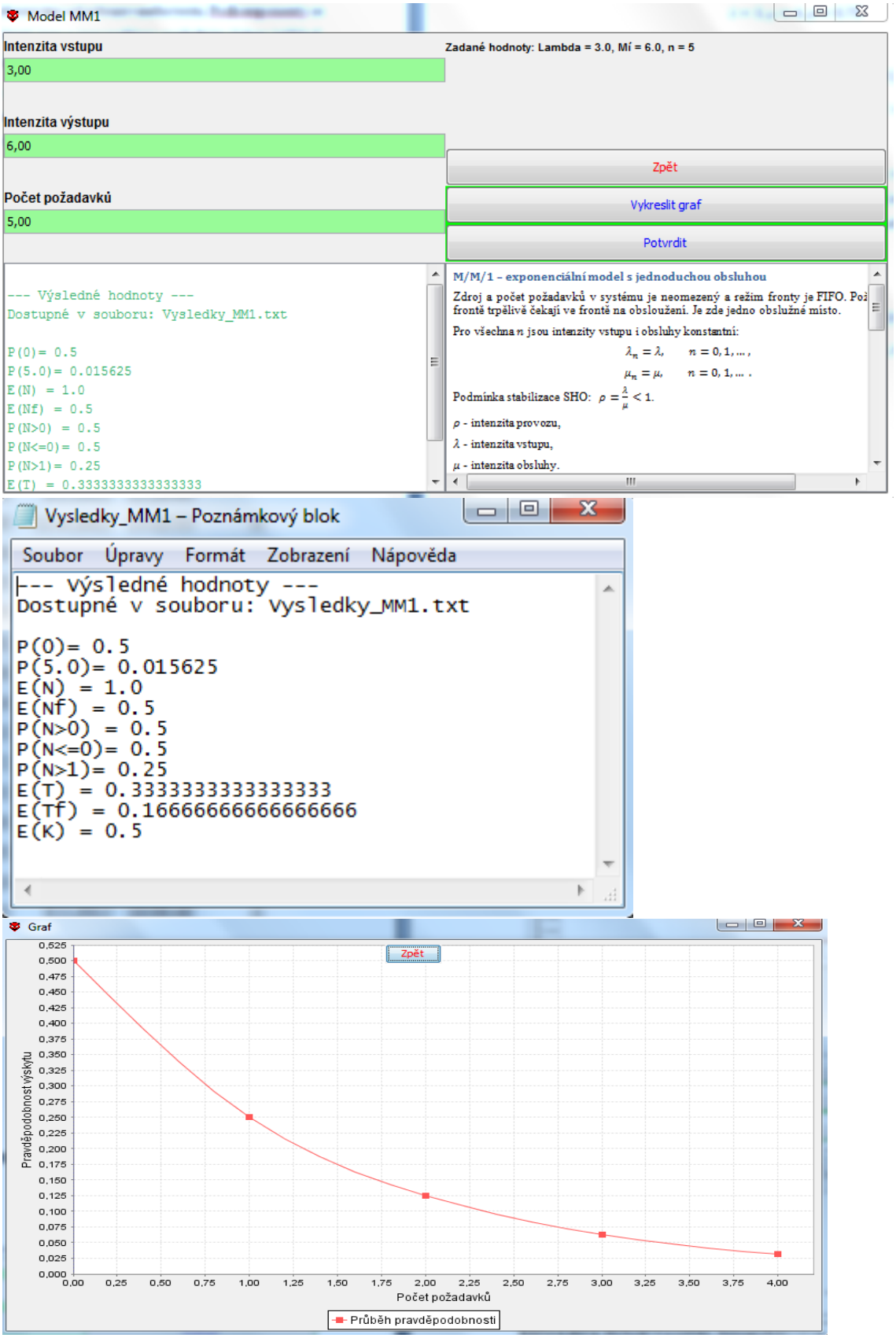
Příloha O – Výstupy programu pro modelový příklad 8

Příloha P - CD-ROM s vytvořeným programem a písemnou prací ve formátu PDF

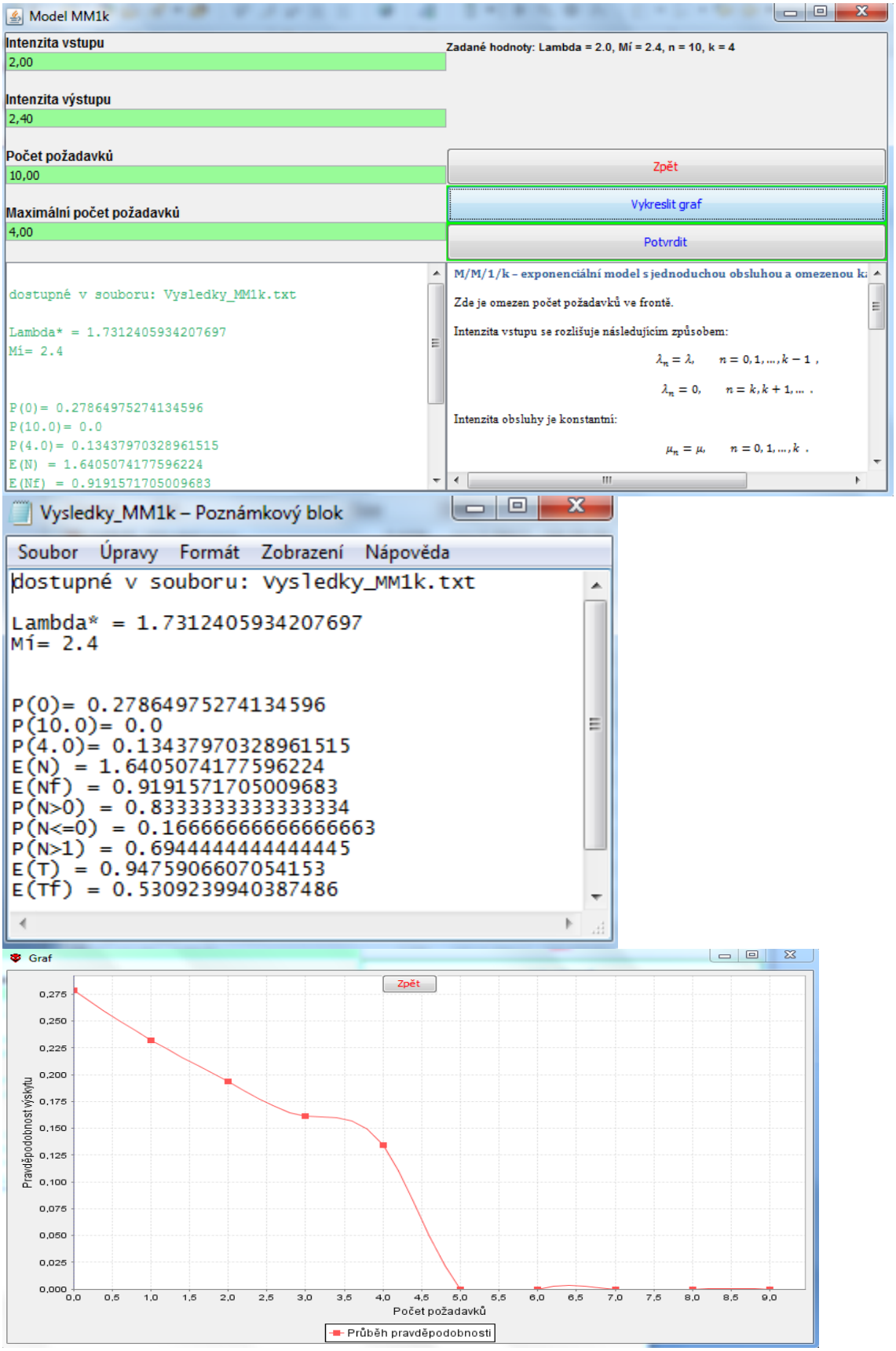
Přílohy



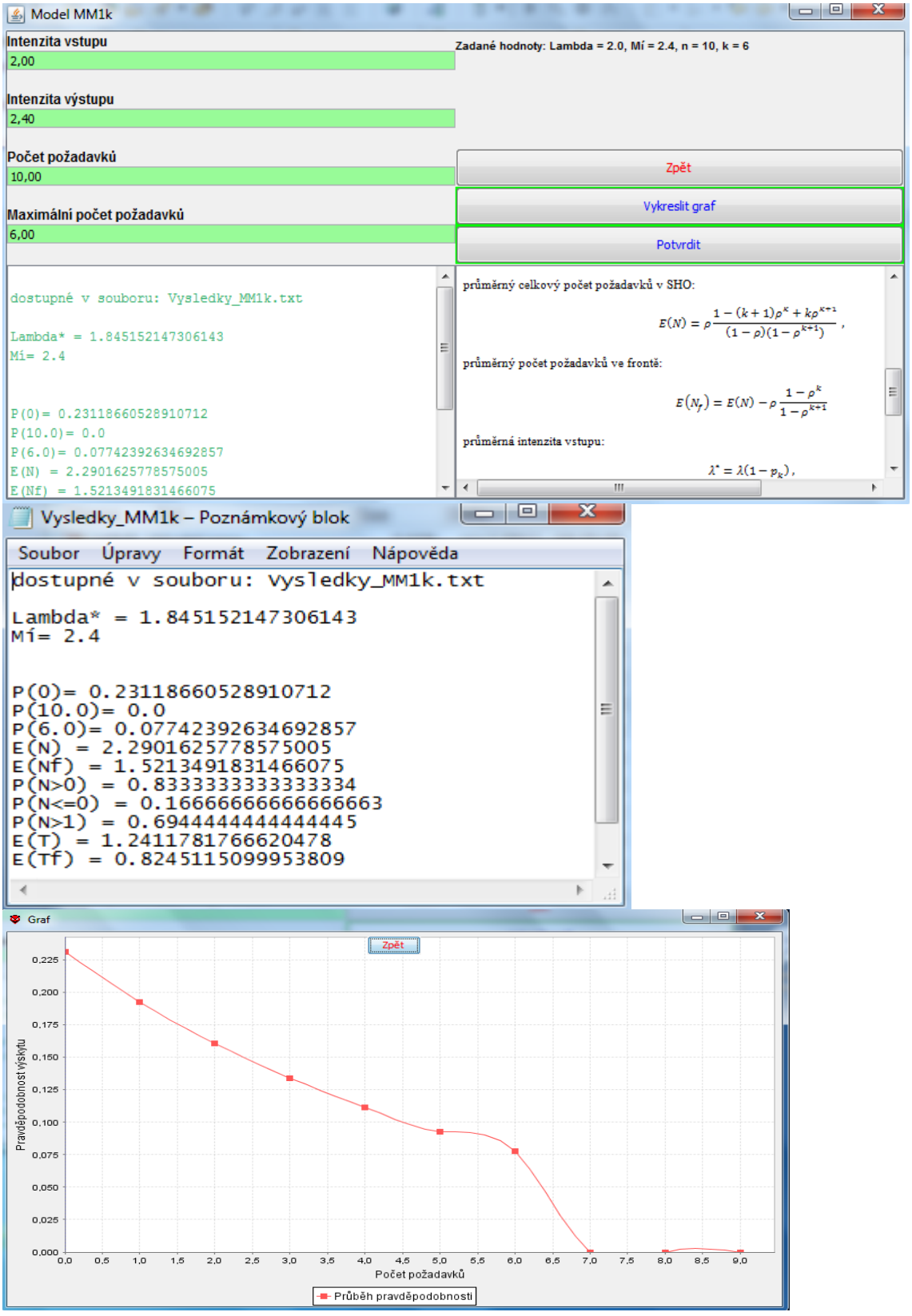
Příloha A – Vstupy a výstupy programu pro modelový příklad 1a



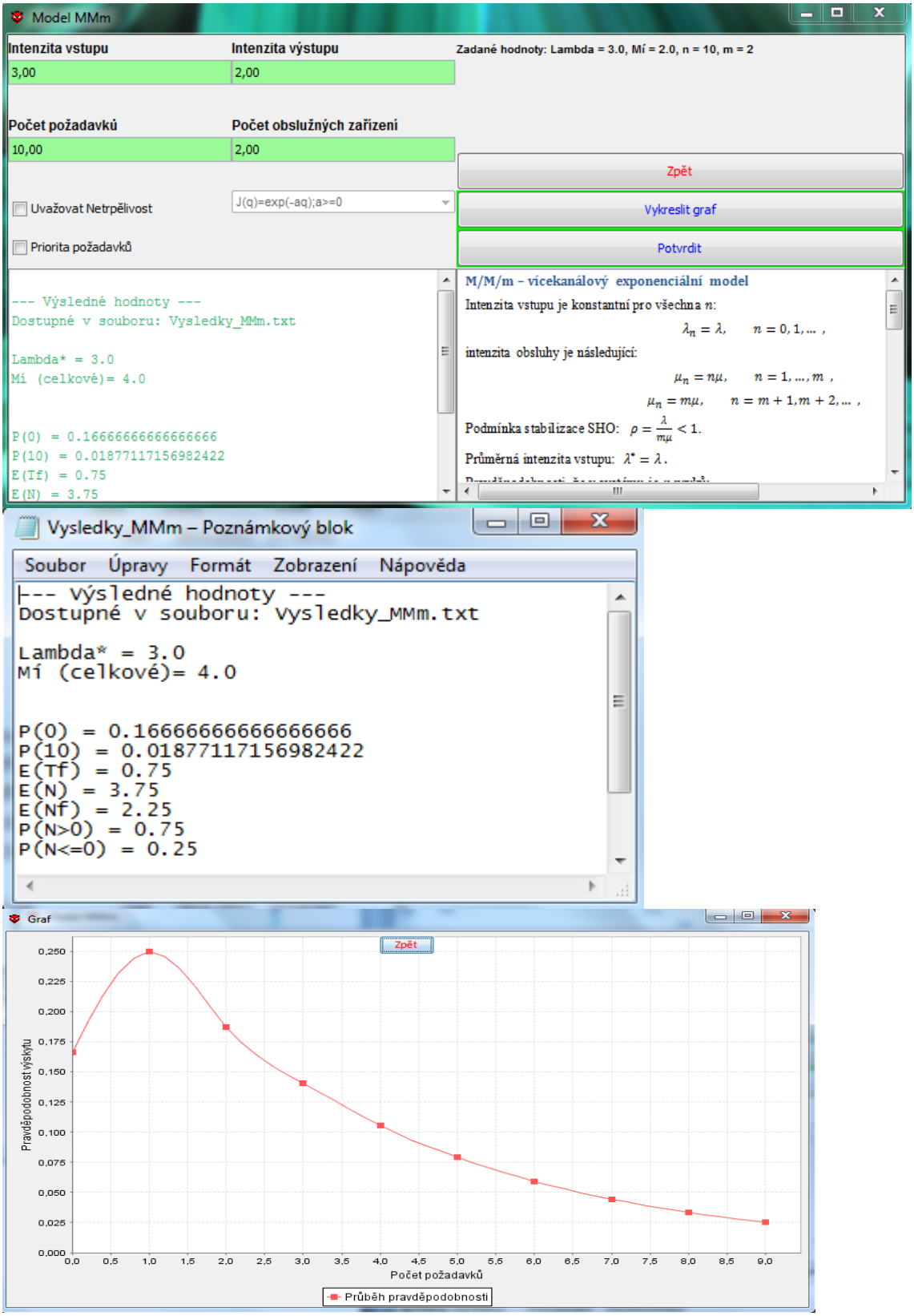
Příloha B – Vstupy a výstupy programu pro modelový příklad 1b



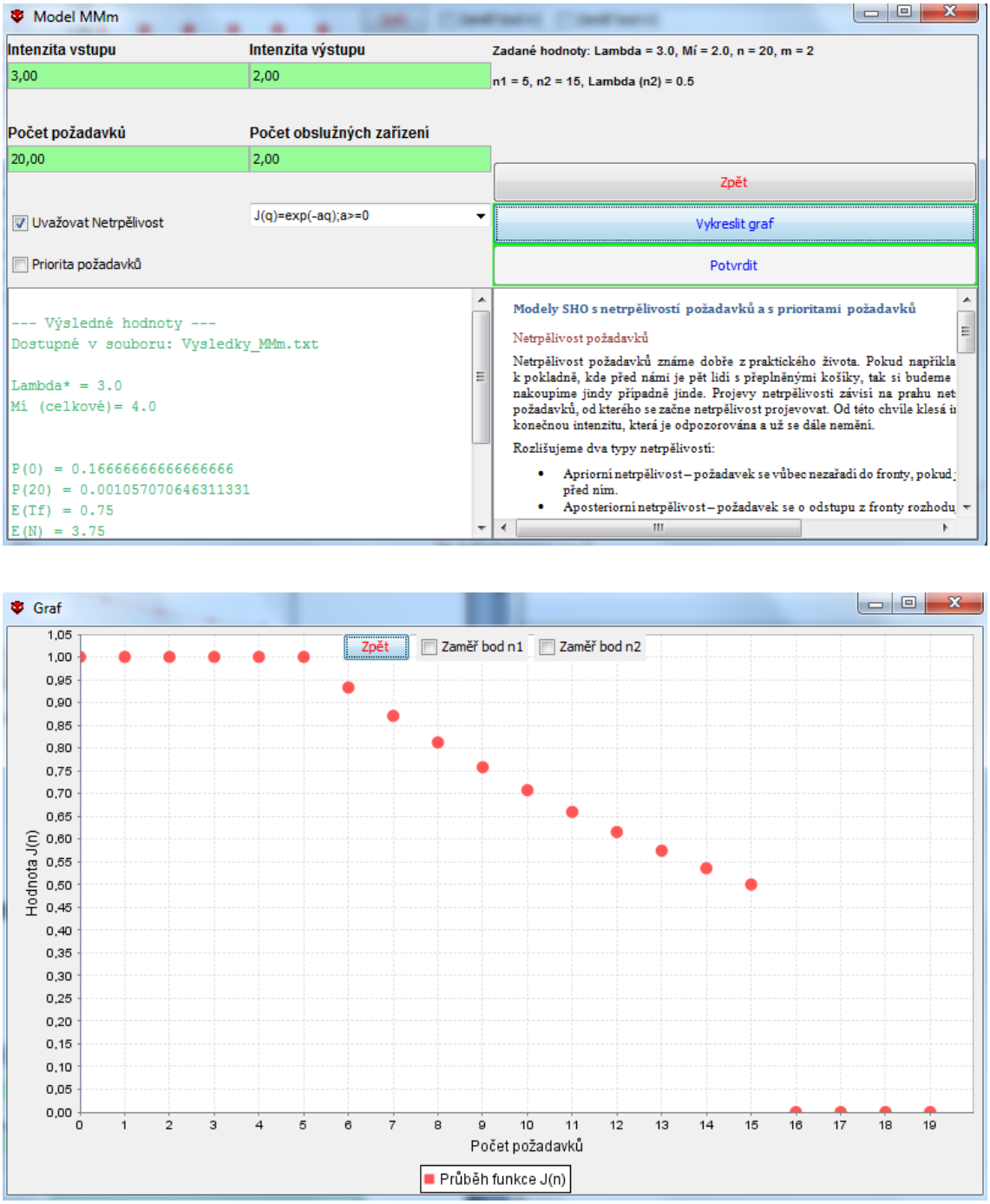
Příloha C - Vstupy a výstupy programu pro modelový příklad 2a



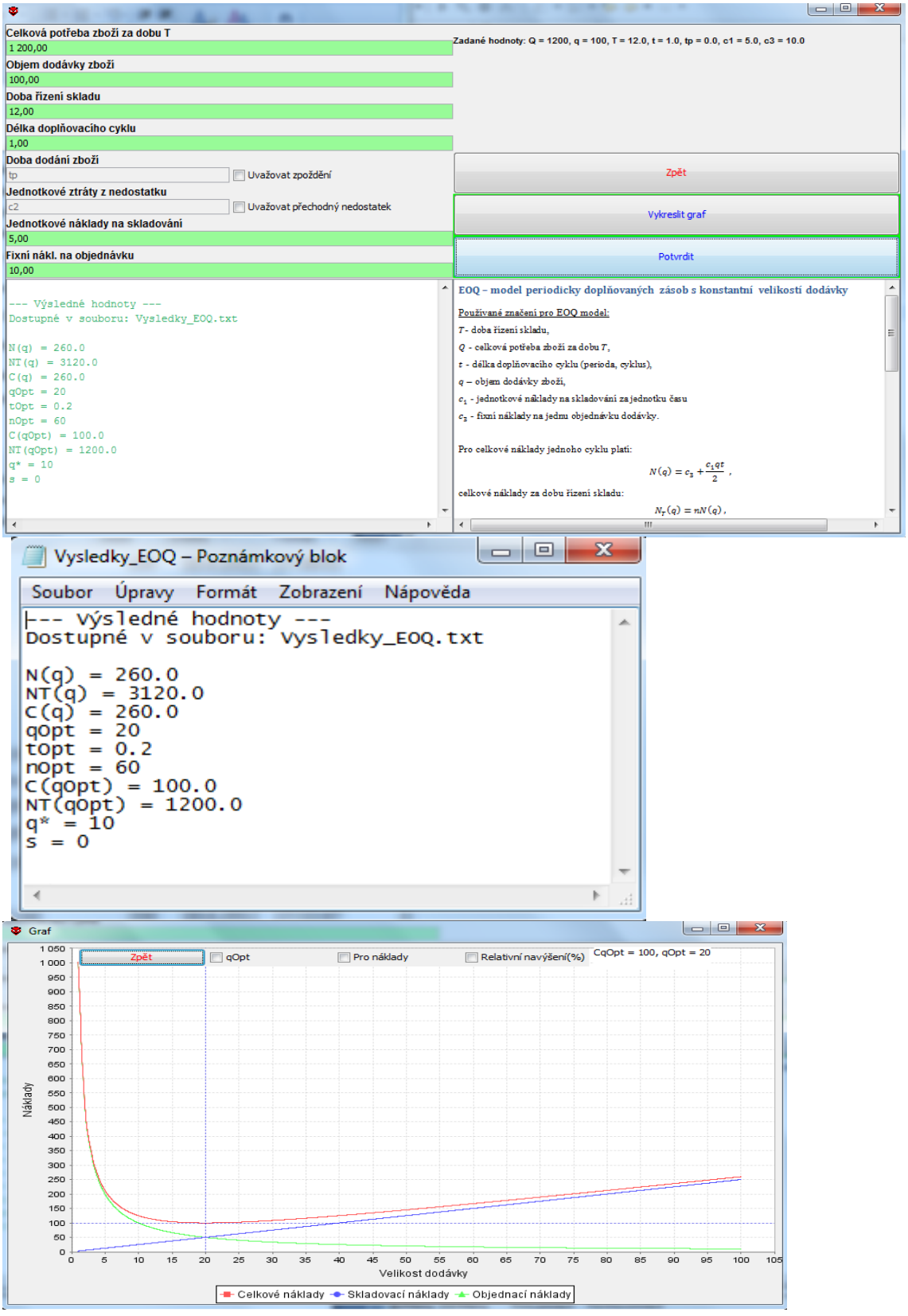
Příloha D - Vstupy a výstupy programu pro modelový příklad 2b



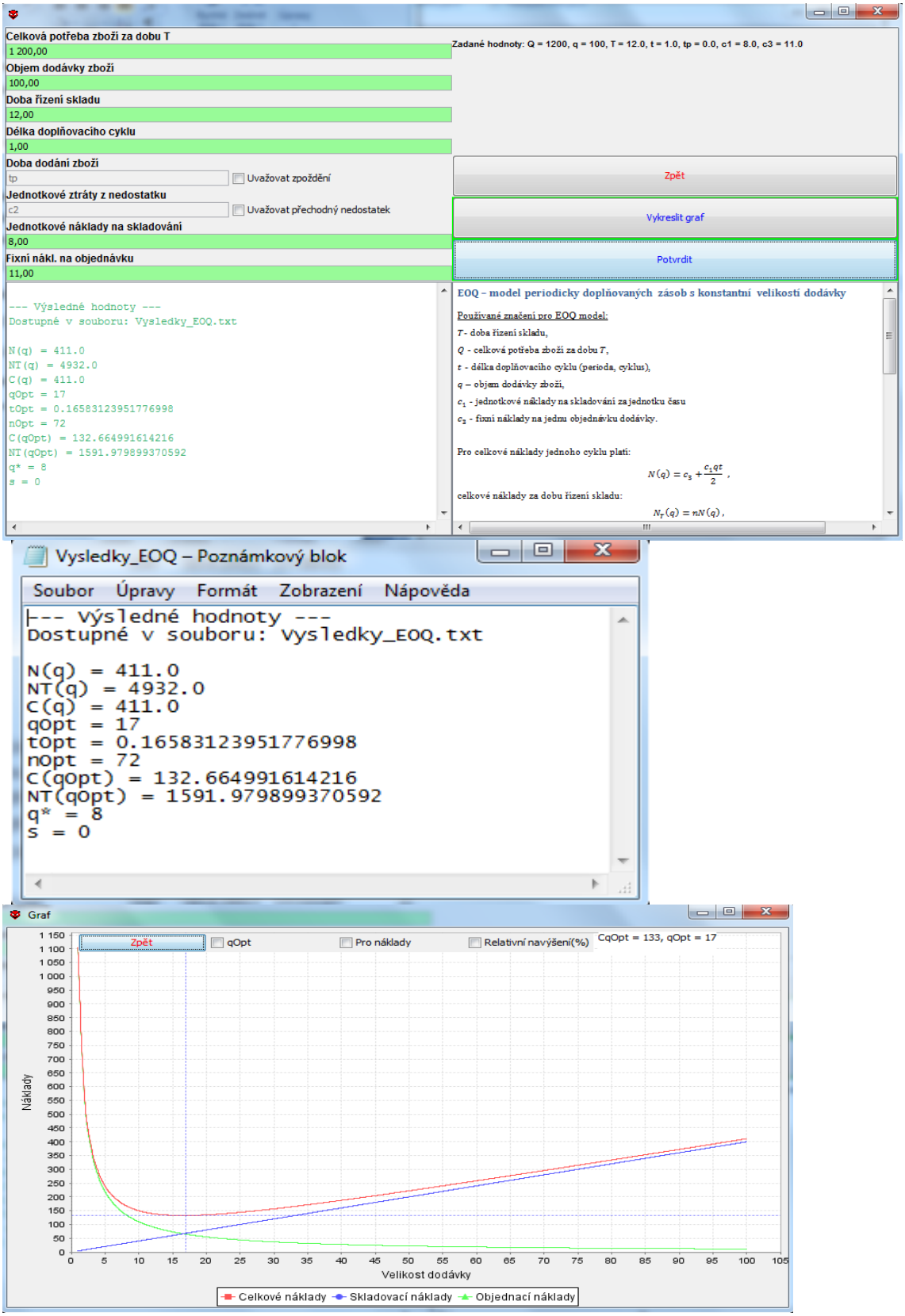
Příloha E - Vstupy a výstupy programu pro modelový příklad 3a



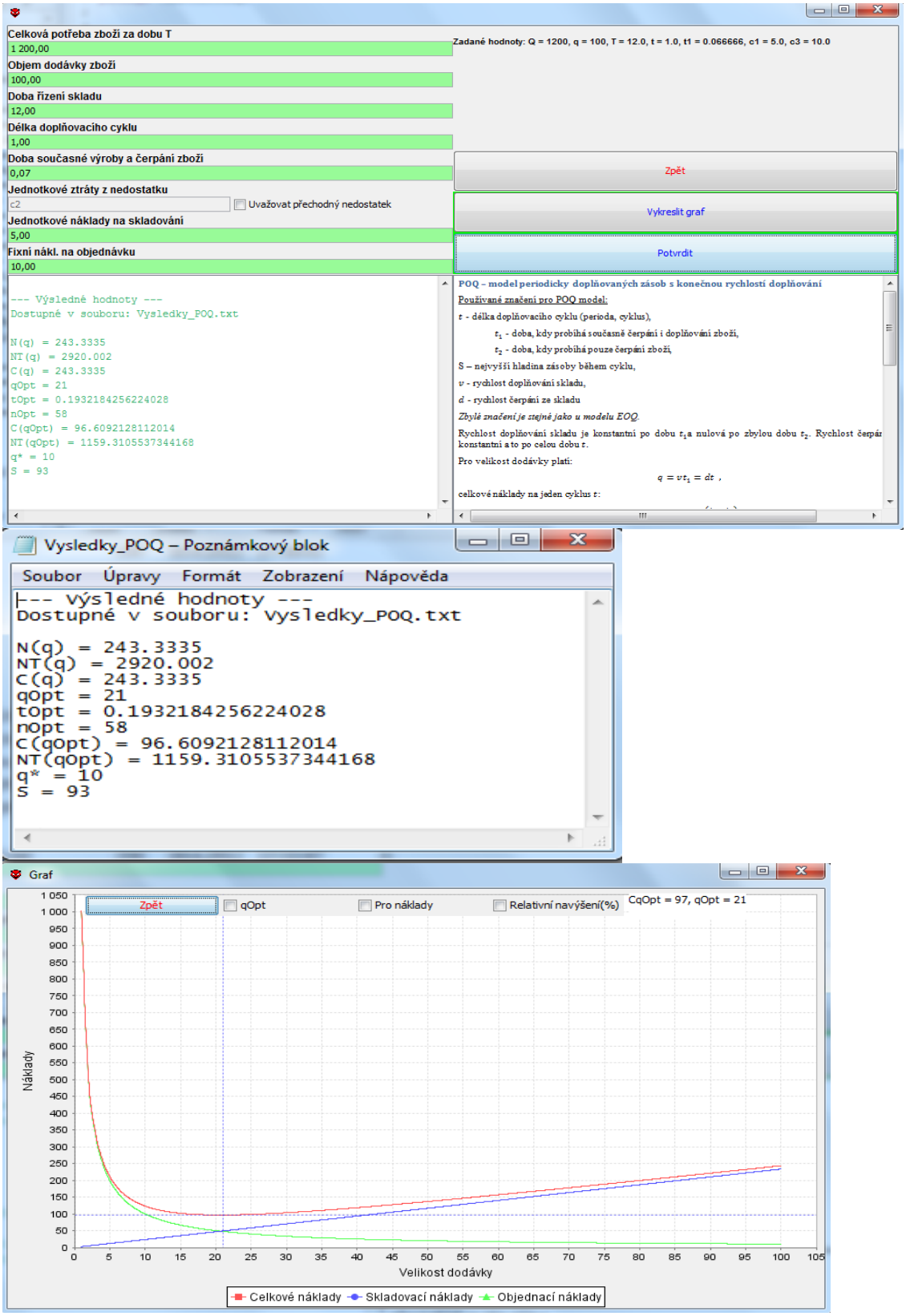
Příloha F - Vstupy a výstupy programu pro modelový příklad 3b: zadání a vykreslení netrpělivosti



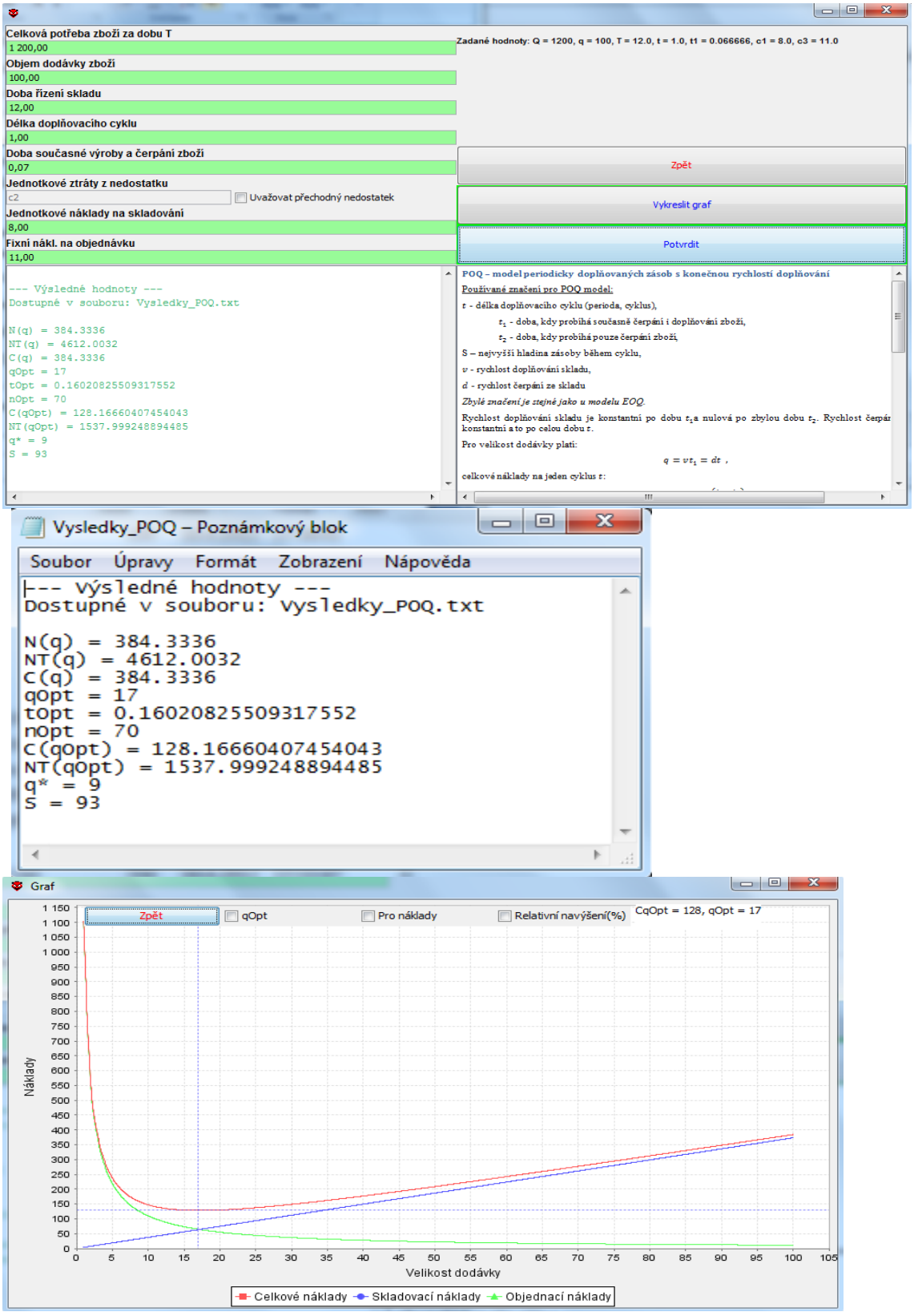
Příloha G - Vstupy a výstupy programu pro modelový příklad 4a



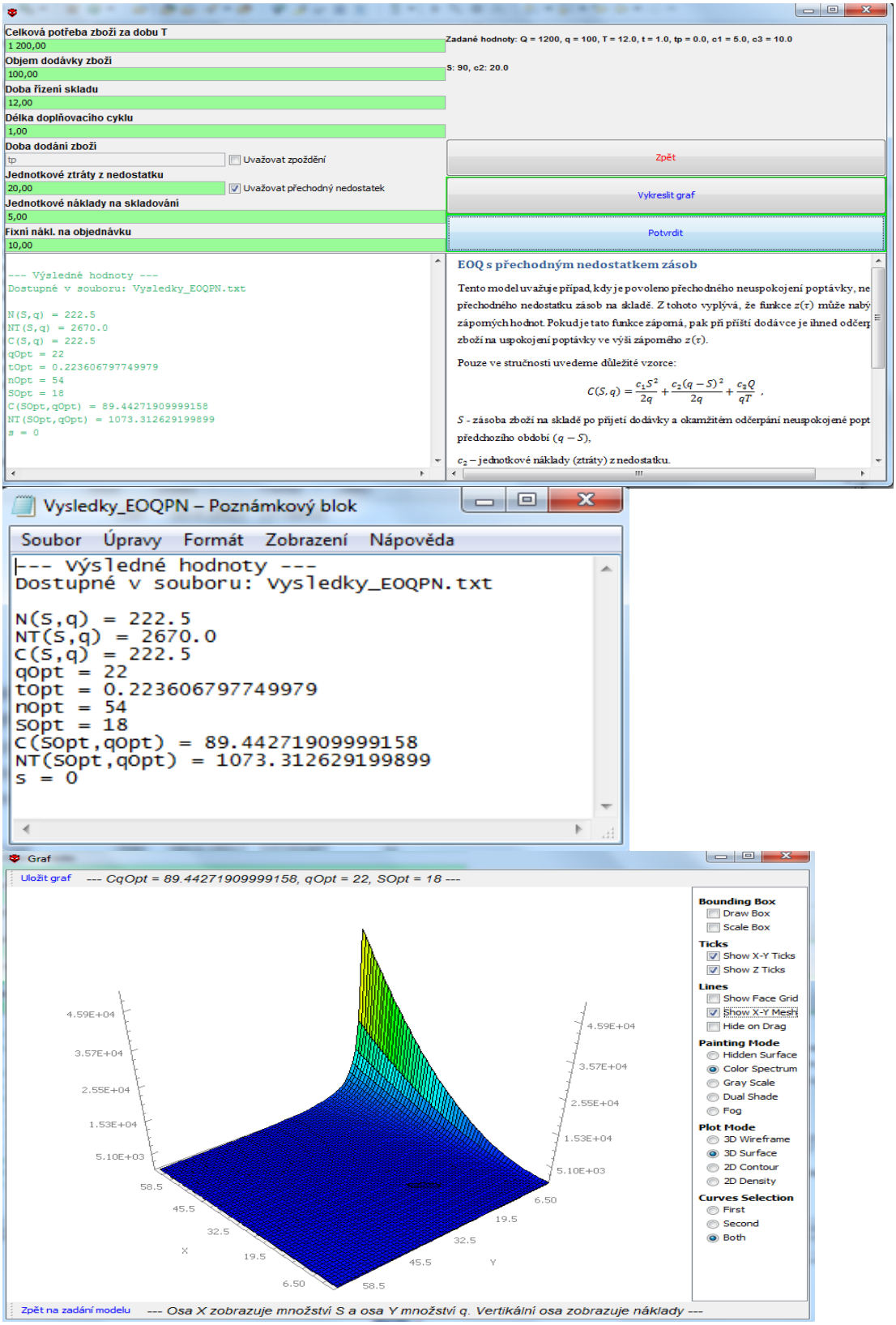
Příloha H - Vstupy a výstupy programu pro modelový příklad 4b



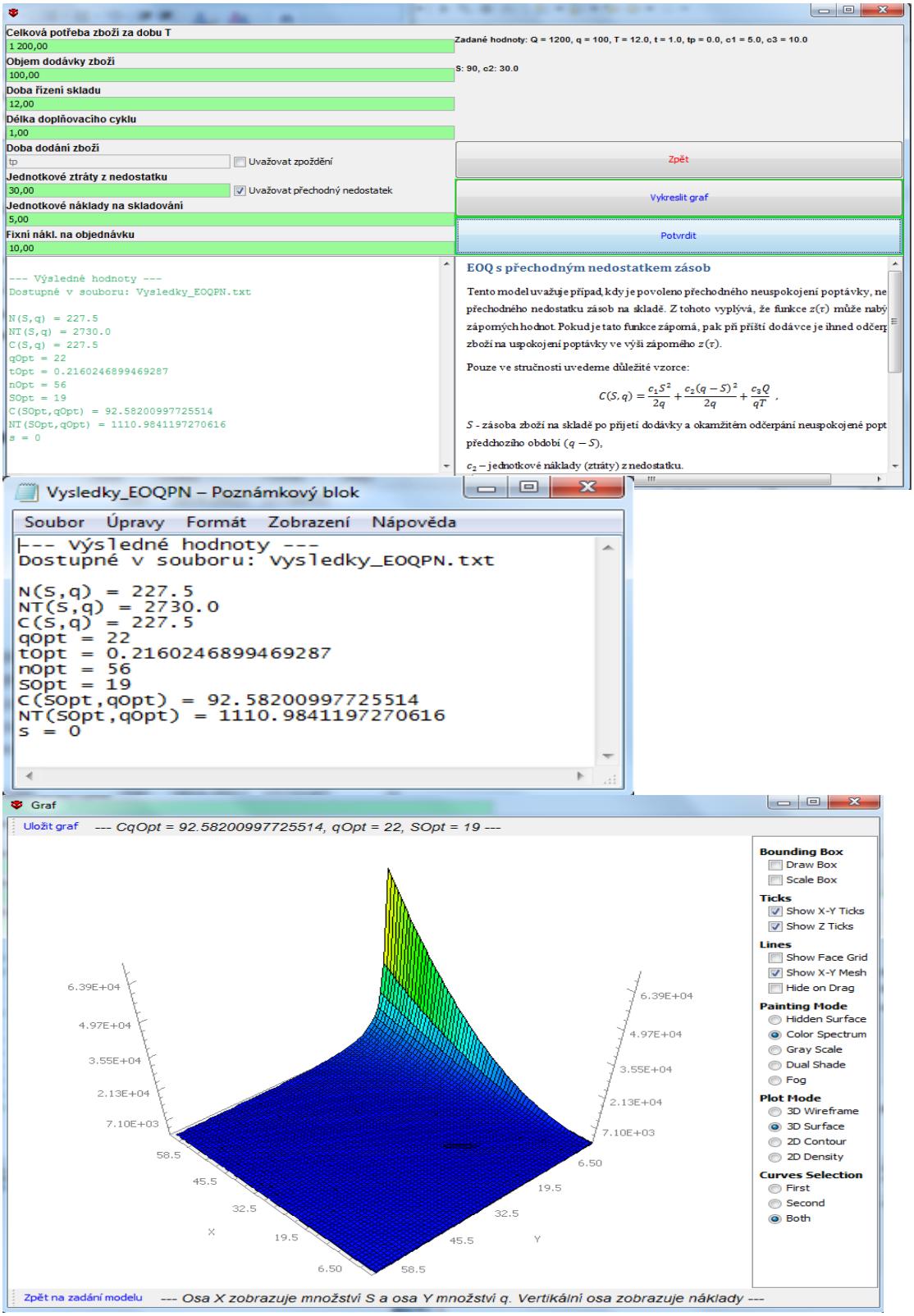
Příloha I - Vstupy a výstupy programu pro modelový příklad 5a



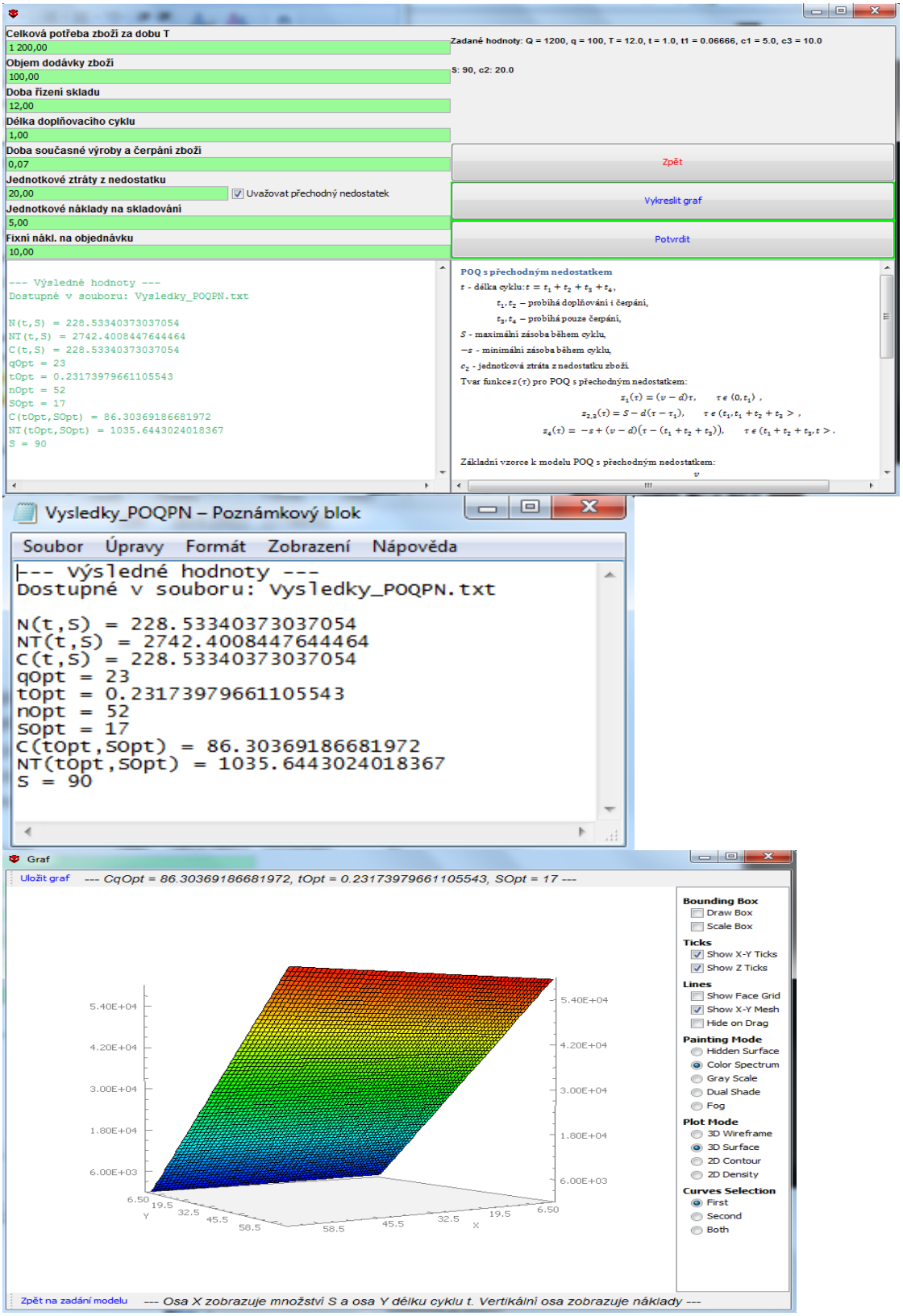
Příloha J - Vstupy a výstupy programu pro modelový příklad 5b



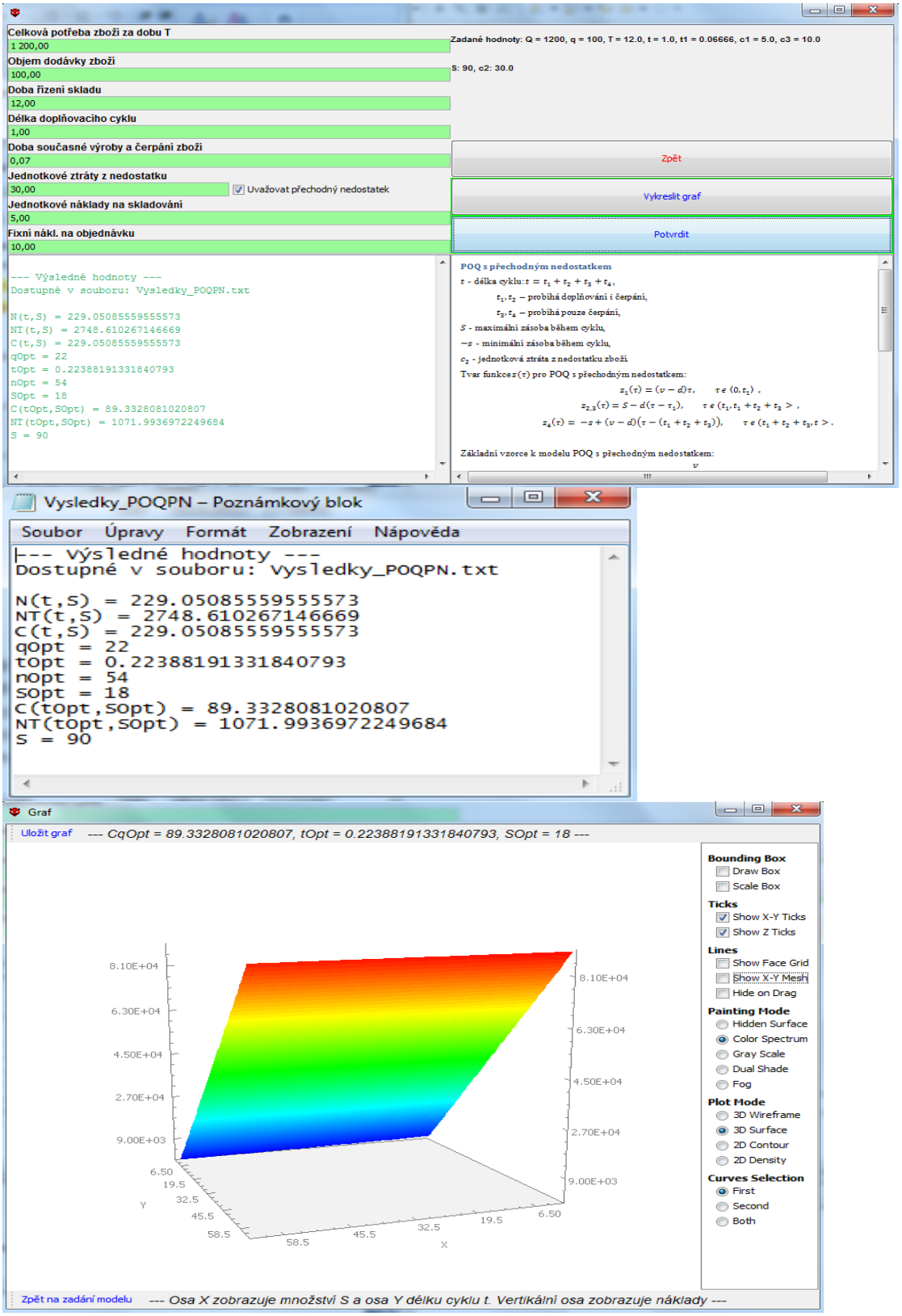
Příloha K - Vstupy a výstupy programu pro modelový příklad 6a



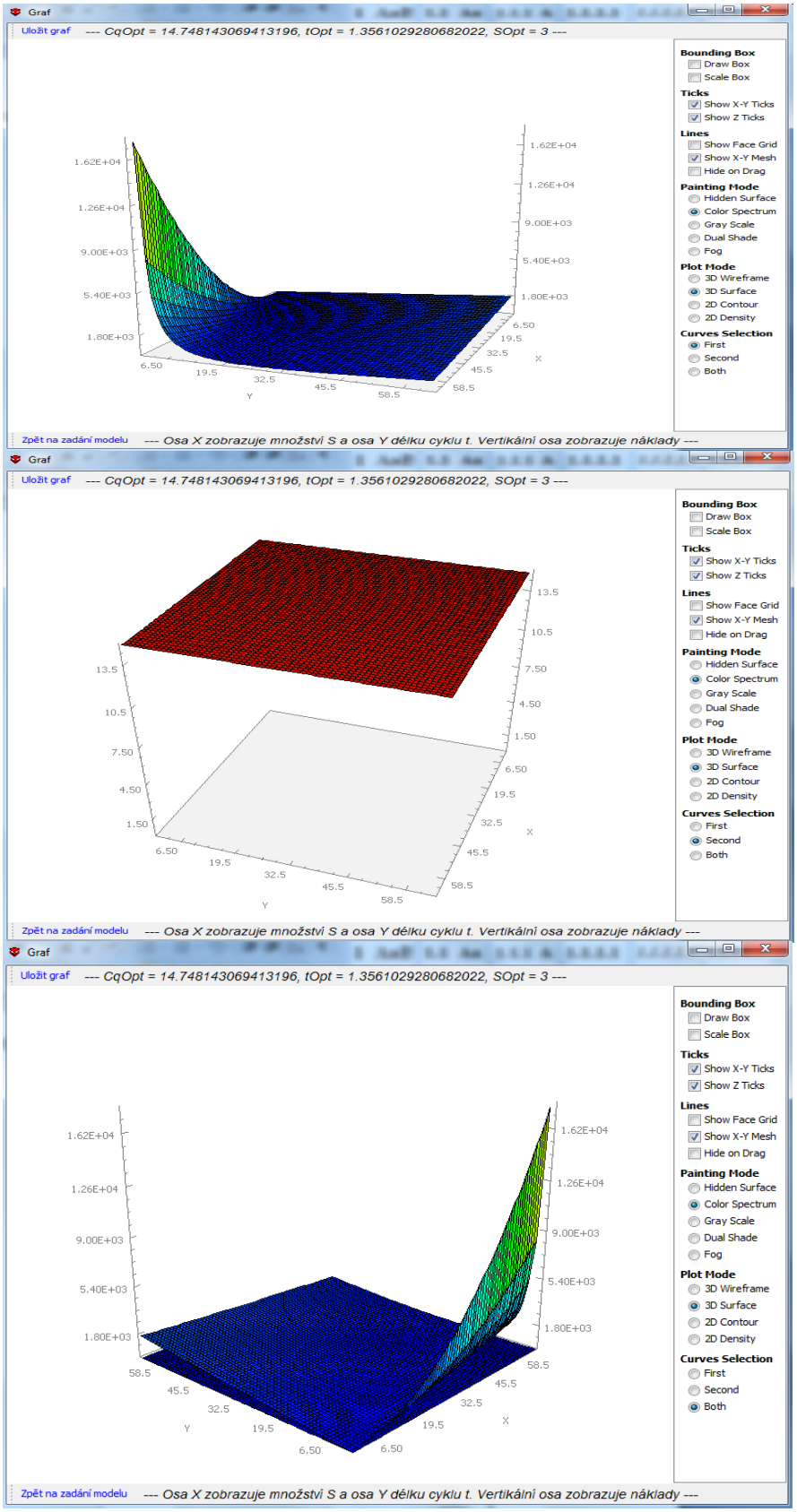
Příloha L - Vstupy a výstupy programu pro modelový příklad 6b



Příloha M - Vstupy a výstupy programu pro modelový příklad 7a



Příloha N - Vstupy a výstupy programu pro modelový příklad 7b



Příloha O – Výstupy programu pro modelový příklad 8

Abstrakt

SLOUP, P. *Tvorba SW modulů pro počítačovou podporu KEM/PMO v OOP Java*. Diplomová práce. Plzeň: Fakulta ekonomická ZČU v Plzni, 71 s., 2012

Klíčová slova: systémy hromadné obsluhy, teorie zásob, Java, grafické uživatelské rozhraní

Předmětem předložené práce je počítačový program vytvořený za účelem podpory KEM/PMO. Program řeší stěžejní oblasti zmíněného předmětu, jako jsou systémy hromadné obsluhy a modely teorie zásob. Je programován v jazyce Java a vytvářen byl ve vývojovém prostředí Eclipse. V této práci je uveden pouze nutný teoretický základ pro výše uvedenou problematiku, především se zde seznamujeme s programem a učíme se s ním zacházet. Na závěr jsou počítány modelové příklady, na nichž je demonstrováno vytvořené programové řešení.

Abstract

SLOUP, P. *Creation of Software Modules for KEM/PMO Computer Aided Support in OOP Java.* Diploma Thesis. Pilsen: Faculty of Economics, University of West Bohemia in Pilsen, 71 pages, 2012

Key words: queuing systems, inventory theory, Java, graphical user interface

The subjekt of this Thesis is a computer application designed for KEM/PMO computer aided support. The application solves key areas of KEM/PMO, such as queuing systems and inventory theory. It is programed in Java and was created in a development environment Eclipse. There is only necessery theoretical background given in this Thesis for the above mentioned issues, it is particularly focused on using the application. In the final part model examples are solved, in which the created software solution is demonstrated.