UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Programový generátor trendů

Václav Bárta

Bakalářská práce
2011

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

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Václav BÁRTA**
Osobní číslo:	**I07575**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Programový generátor trendů**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cíl: Realizace softwarového generátoru časových trendů, jako náhrada fyzických senzorů.
V teoretické části bakalářské práce bude proveden rozbor vlastností formátu XML z hlediska přenosu informací mezi aplikacemi a se zvláštním zaměřením na dynamický výstup opakujících zpráv. Dále bude zpracován přehled integrace technologie XML do Java.
V implementační části bude realizován program s těmito vlastnostmi:
1. Generátor časových trendů podle scénářů zadané lomenou čarou.
2. Pro každý trend bude zobrazován výstup v podobě časového průběhu.
3. Výstup bude ve formátu XML. Cílem bude též návrh schématu XML.
4. Pro každý trend bude možné zapnout rušení.
5. Trendem může například být cena akcií, jejich denní prodej nebo nákup. Nebo z fyzikální oblasti například teplota, tlak, vlhkost, rosný bod.
Další požadavky budou upřesněny vedoucím bakalářské práce.
Program realizujte v programovacím jazyku JAVA, v prostředí NetBeans a za využití grafické knihovny JFreeChart.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

[1] Pecinovský,Rudolf, Návrhové vzory, Computer Press, 2007 vydání první, ISBN 978-80-251-1582-4

[2] Page-Jones,Meilir , Základy objektově orientovaného návrhu v UML, Grada 2001, ISBN 80-247-0210-X

[3] Patton, Ron, Testování softwaru, Computer Press, 2002 vydání první, ISBN 80-7226-636-5

[4] Gunderloy, Mike, Z kodéra vývojářem, Computer Press, 2007 vydání první, ISBN 978-80-251-1517-6

[5] Herout, Pavel, Java a XML, Kopp, 2007, ISBN 978-80-7232-307-4

Online zdroje

*** Java Dokumentace - http://download.oracle.com/javase/6/docs/**

Vedoucí bakalářské práce: **Ing. Karel Šimerda**
Katedra softwarových technologií

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

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

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2011

Prohlášení autora

Prohlašuji, že jsem tuto práci 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 30. 4. 2011 Václav Bárta

Poděkování

Na tomto místě bych rád v první řadě poděkoval vedoucímu své bakalářské práce panu Ing. Karlu Šimerdovi za pomoc a cenné rady, které mi poskytl nejenom v souvislosti s touto prací. Dále slečně Veronice Dufkové za korektury a nejen za ně. Následně pak i celé rodině a přátelům za jejich podporu a trpělivost.

Anotace

Bakalářská práce pojednává o implementaci aplikace generující trendy s možností rušení a zobrazení těchto trendů. Součástí práce je seznámení s problematikou trendů, jejich generování a rušení, popis formátu XML a s ním souvisejícími technologiemi, dále popis metod zpracování XML dokumentů v jazyce Java a popis vývoje aplikace. Aplikace byla vytvořena s využitím těchto technologií: XML, XSD, XSLT, JAXB, Java, JFreeChart, Swing.

Klíčová slova

generátor, trend, rušení, xml, xsd, java aplikace, jaxb, jfreechart.

Title

Software trend generator.

Annotation

This bachelor work deals with implementation of application which generates trends. Application enables jamming and displaying of these trends. Bachelor work includes introduction to trends, their generating and jamming, description of the XML format and related technologies. Bachelor work also deals with description of methods of processing XML documents in programing language Java, and description of evolving of the application. This application was created by using these technologies: XML, XSD, XSLT, JAXB, Java, JFreeChart, Swing.

Keywords

generator, trend, jam, xml, xsd, java application, jaxb, jfreechart.

Obsah

Seznam zkratek

XML	Extensible Markup Language
DTD	Document Type Definition
XSD	XML Schema Definition
RNG	RelaxNG
XPath	XML Path Language
XSLT	Extensible Stylesheet Language Transformations
UTF-8	UCS Transformation Format
DOM	Document Object Model
SAX	Simple API for XML
StAX	Streaming API for XML
JAXP	Java Architecture for XML Procesing
JAXB	Java Architecture for XML Binding
XJC	XML to Java Compiler
JDOM	není oficiálně akronym[1] (neoficiálně Java Document Object Model)
JWSDP	Java Web Services Developer Pack
SWING	API for providing a graphical user interface (GUI) for Java programs

[1] Is JDOM an acronym?
Nope. Just like JDBC is not officially an acronym, neither is JDOM. This ensures we comply with Sun's trademark policies as explained at http://www.sun.com/policies/trademarks.

Seznam obrázků

Seznam tabulek

1 Úvod

Ve své bakalářské práci se budu zabývat vytvořením programového generátoru trendů – náhrady fyzických senzorů. Cílem práce je realizace softwarového generátoru časových trendů podle scénářů zadaných lomenou čarou, pro trendy, jednotlivé trendy budou mít zobrazitelný výstup, pro trendy bude možno zapnout rušení. Vstupní a výstupní trendy budou využívat formát XML, cílem práce je též návrh schémat pro tyto dokumenty.

Práce obsahuje část teoretickou a praktickou.

Teoretická část se zaměřuje na vysvětlení pojmu trend a proces jeho generování, součástí jsou popisy rušení a konkrétních rozdělení pravděpodobností, z nichž jsou generovány. Dále seznamuje se značkovacím jazykem XML a jeho strukturou, porovnává ho s binárními formáty, zabývá se schémovými jazyky (DTD, XSD, RNG) používanými pro omezení XML dokumentů dle požadavků a s tím související validací XML dokumentů, dalšími jazyky souvisejícími s XML – dotazovacím jazykem XPath a stylovacím jazykem XSLT. Opomenuty nejsou ani techniky zpracování XML dokumentů obecně i specificky pro programovací jazyk Java. Popsány jsou různé typy parserů lišící se přístupem jak k XML dokumentům, tak k jejich reprezentaci v paměti.

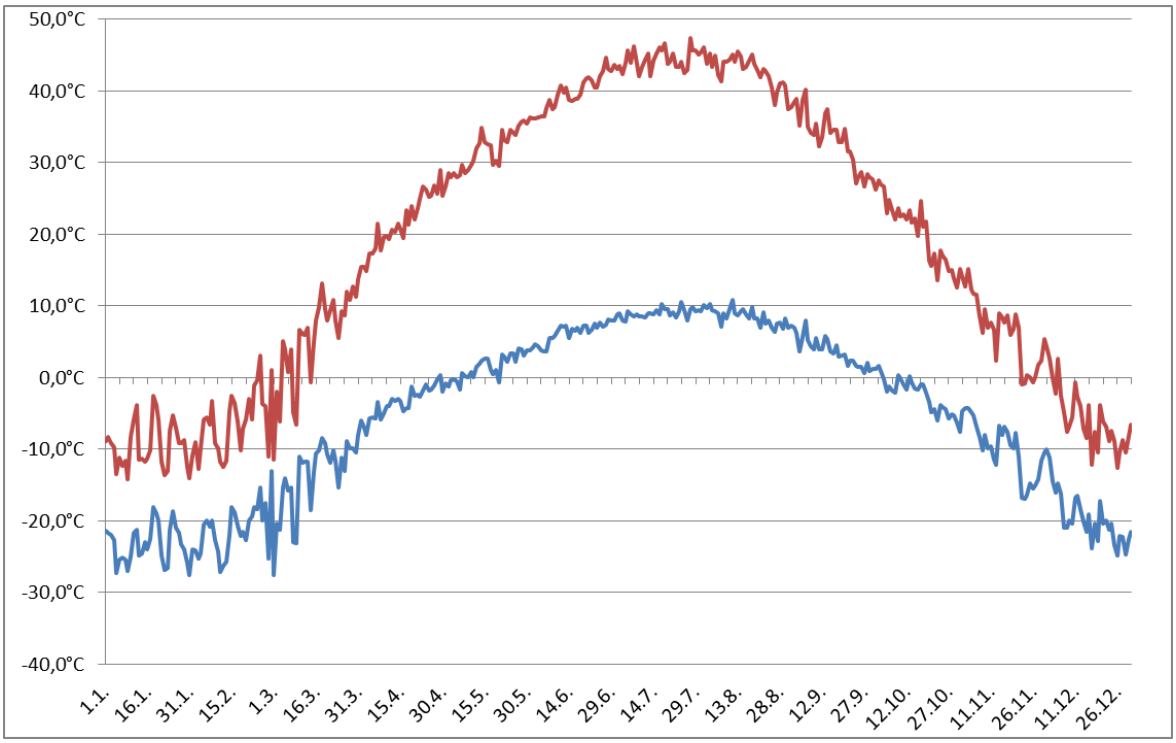
Praktická část se zaměřuje na konkrétní realizaci programového generátoru trendů, a tedy popisuje jak naprogramované třídy, jejich vzájemné vztahy a jejich účel, ale i algoritmy použité pro zarušení průběhu a postupy pro generování trendů v reálném čase.

2 Trendy

2.1 Definice a využití

Trend je určité směřování v čase, typickým příkladem trendu je vývoj ceny akcií na burze, jejich denní prodej nebo nákup, teplota, tlak, vlhkost, rosný bod a další. Trendem může být vývoj jakékoli sledované hodnoty v čase.

Vygenerovaný trend lze použít jako náhrada reálných vstupních dat pro aplikace, které zobrazují, analyzují, ukládají či jinak využívají vstupní data měnící se v čase. Body trendu je možno vygenerovat najednou či průběžně. Průběžné generování trendu odpovídá situaci, kdy trend neznáme dopředu a postupně získáváme nové hodnoty. Vygenerování celého trendu odpovídá situaci, kdy je nám trend dopředu známý – celý průběh trendu můžeme zpracovávat najednou.



Obrázek 1 - Teplotní trendy – Historická minima a maxima[2]

2.2 Programový generátor trendů

Účelem generátoru trendů je vytvořit na základě trendů zadaných lomenou čarou podrobnější trendy s možností zarušení průběhu. Např. vstupní XML soubor popisuje trend obsahující teploty naměřené v rozmezí jedné hodiny, a programový generátor trendů z něj vytvoří trend sestávající se z hodnot s odstupem jedné minuty. Takto vygenerovaný trend lze dále použít pro různé účely.

[2] Zdrojová data: http://old.chmi.cz/meteo/ok/extrklem.html

2.3 Zadávání a generování

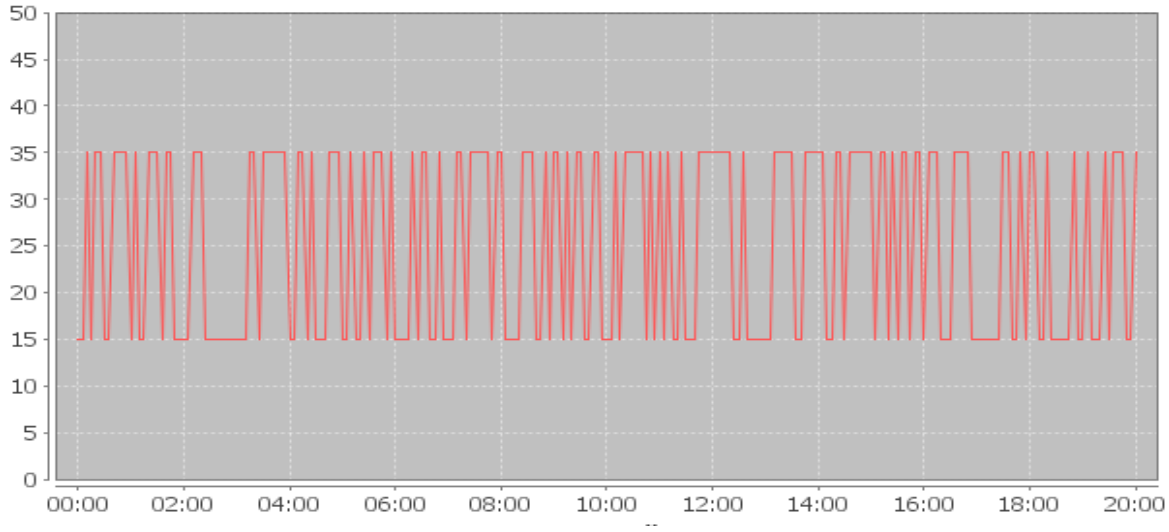
Trend je určen veličinou a jednotlivými body trendu. Body jsou zadány hodnotou a časem, kdy hodnota nastala. Body lze opatřit textovým popisem. Generování výstupního trendu bez rušení je doplněním bodů na spojnice mezi body vstupního trendu – tzv. lomené čáry. V případě rušení je hodnota každého dopočteného bodu trendu určena součtem vypočtené hodnoty a vygenerovaného rušení z daného rozdělení pravděpodobnosti. Rušení je realizováno jako pseudonáhodná veličina spadající do některého z rozdělení pravděpodobnosti.

2.4 Implementovaná rozdělení pravděpodobnosti

2.4.1 Alternativní

Náhodný pokus má jen dva možné výsledky a pravděpodobnost výsledků je stejná. Využití je v libovolné dvoustavové situaci – např. při hodu mincí jsou možnými výsledky panna a orel a pravděpodobnost obou výsledků je teoreticky shodná.

Konkrétní implementace: Funkce má 2 parametry reprezentující možné výsledky, návratová hodnota funkce je rovna jednomu z nich. Alternativní rozdělení návratových hodnot je zajištěno použitím funkce java.util.Random.nextBoolean().



Obrázek 2 - Alternativní rušení

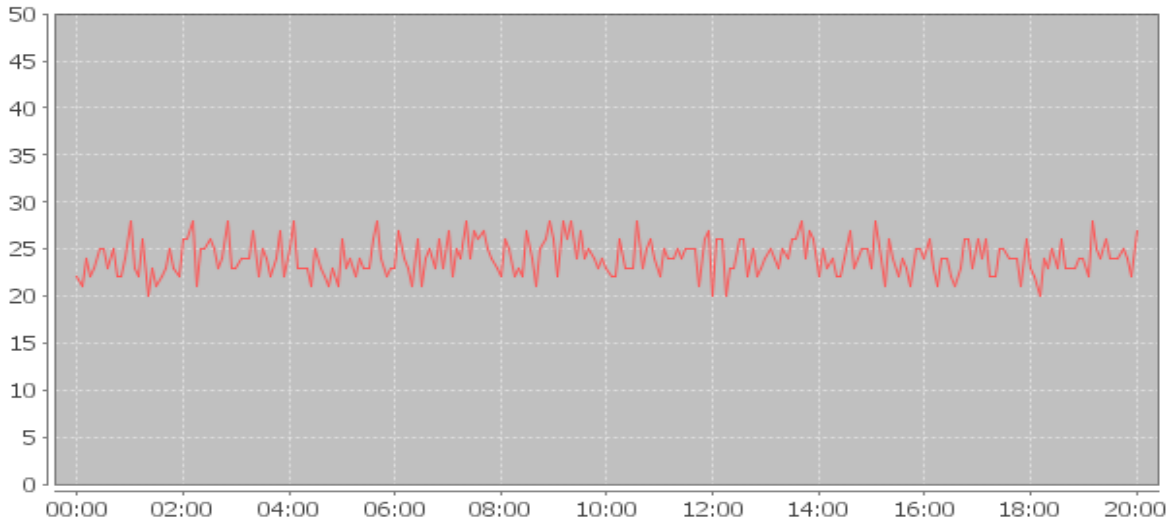
(Nezarušená hodnota 25, Minimum -10, Maximum 10)

2.4.2 Binomické

Řada náhodných pokusů délky X. Každý pokus má dva možné výsledky 0 a 1 a ve všech pokusech je pravděpodobnost hodnoty 1 stejná. Binomické rozdělení je rozdělení nové náhodné veličiny, která je definována jako počet těchto pokusů s výsledkem 1. Jedná se tedy o četnost výskytu náhodného jevu v X nezávislých pokusech.

Konkrétní implementace: Funkce má 3 parametry reprezentující minimální výsledek, maximální výsledek a pravděpodobnost jevu, návratová hodnota funkce je rovna

součtu minimálního výsledku a hodnoty z binomického rozdělení pro počet pokusů rovný rozdílu maximálního a minimálního výsledku.

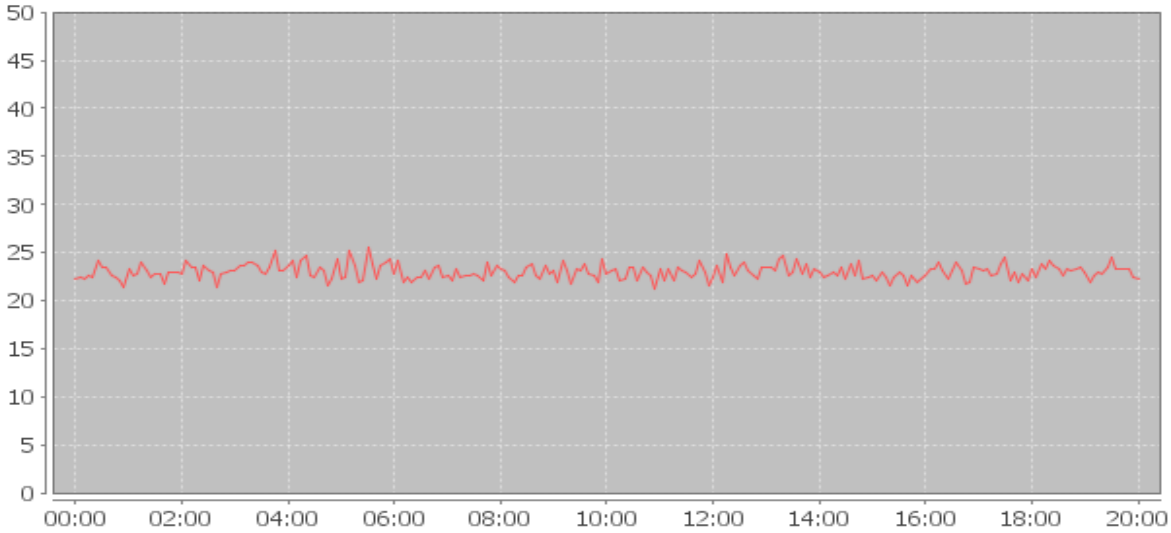


Obrázek 3 - Binomické rušení

(Nezarušená hodnota 25, Minimum -5, Maximum 15, Odchylka 0.2)

2.4.3 Gamma

Gamma rozdělení se obvykle používá pro výpočty pojistné škody, dešťových srážek, pro zkoumání proměnných, které mohou mít asymetrické rozdělení, případně v geologii při analýze obsahu některých kovů v rudních blocích. Zdrojem algoritmu pro výpočet hodnoty z gamma rozdělení je Vladislav Vyshemirsky[3].



Obrázek 4 - Gamma rušení

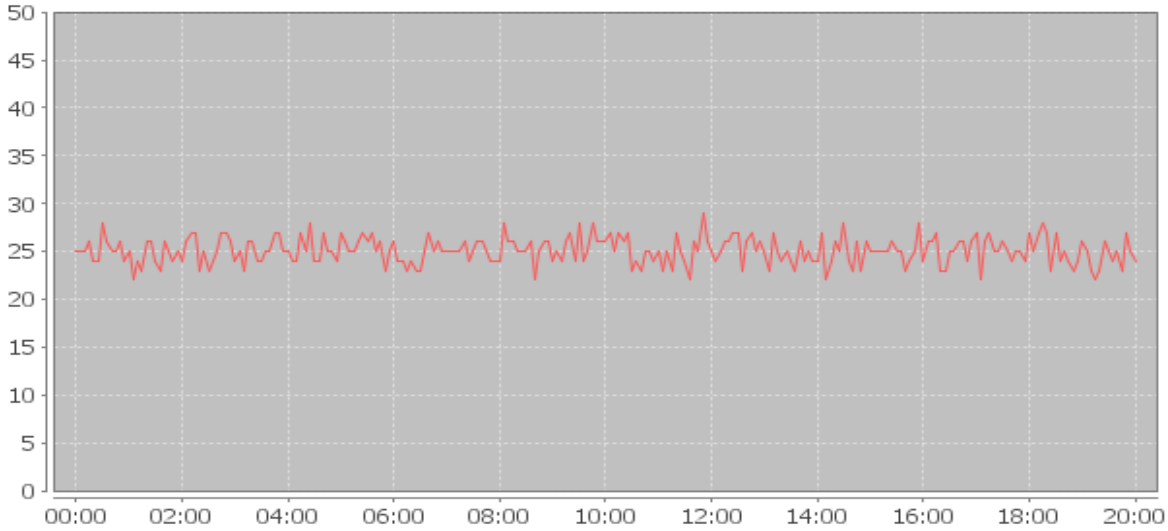
(Nezarušená hodnota 25, Minimum -5, Maximum 15, Odchylka 0.2)

[3] http://vyshemirsky.blogspot.com/2007/11/sample-from-gamma-distribution-in-java.html

2.4.4 Hypergeometrické

Hypergeometrické rozdělení má výsledek řady náhodných pokusů délky X, ve které je výsledek následujícího pokusu závislý na pokusu předcházejícím. Typickým příkladem je výběr prvků bez vracení.

Konkrétní implementace: Funkce má 4 parametry reprezentující minimální hodnotu, maximální hodnotu, počet prvků a počet vyhovujících prvků. Počet výběrů prvků je určen jako rozdíl maximální a minimální hodnoty. Návratová hodnota je rovna součtu minimální hodnoty a počtu vybraných vyhovujících prvků s hypergeometrickým rozdělením.[4]



Obrázek 5 - Hypergeometricke rušení

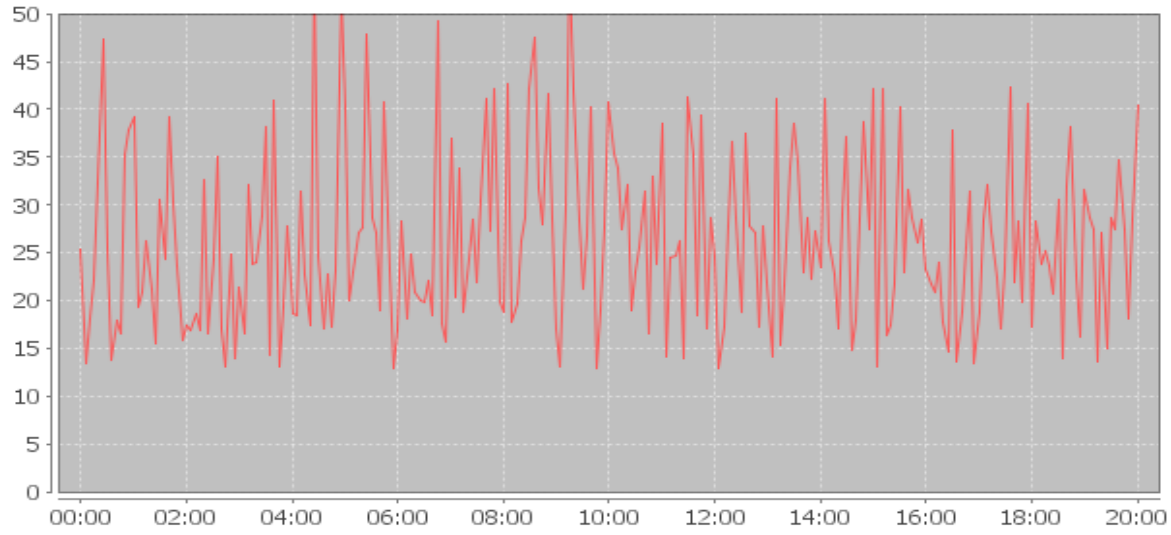
(Nezarušená hodnota 25, Minimum -5, Maximum 10, Prvků 30, Vyhovujících 10)

2.4.5 Normální

Normální rozdělení je velmi důležité, protože se nejčastěji vyskytuje, mnoho jiných rozdělení se mu blíží a řada jiných rozdělení se jím dá nahradit.[5] Normálního (Gaussova) rozdělení nabývají náhodné chyby, některé fyzikální a technické veličiny či náhodné veličiny v ekonomii. Normální rozdělení je symetrické kolem střední hodnoty, hustota má tvar zvonu s maximem ve střední hodnotě, okraje rozdělení konvergují k ose x.

Konkrétní implementace: Funkce má 3 parametry reprezentující střední hodnotu, odchylku a pro zpřesnění výpočtu přepínač sin/cos. Návratová hodnota je vypočtena pomocí Box-Müllerovy transformace[6] za pomoci vygenerovaných hodnot z rovnoměrného rozdělení pravděpodobnosti.

[4] http://homen.vsb.cz/~oti73/cdpast1/KAP04/PRAV4.HTM
[5] http://www.fs.vsb.cz/books/SystAnal/texty/10.htm
[6] http://www.aiaccess.net/English/Glossaries/GlosMod/e_gm_box_muller.htm

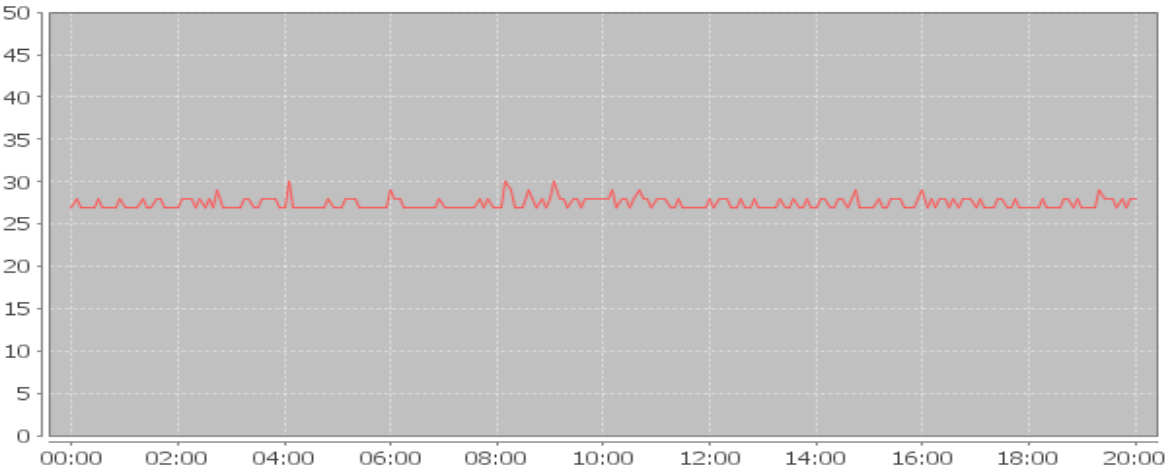


Obrázek 6 - Normální rušení

(Nezarušená hodnota 25, Střed 0, Odchylka 10)

2.4.6 Poissonovo

Poissonovo rozdělení je tzv. rozdělení řídkých jevů, vyjadřuje počet jevů s malou pravděpodobností výskytu v určitém časovém/objemovém kvantu. Často se využívá pro aproximaci binomického rozdělení při velkém počtu pokusů a malé pravděpodobnosti výskytu jevu (při více jak třiceti pokusech). Poissonovo rozdělení mají i odhady příchodů zákazníků, počet automobilů projíždějících konkrétním místem, počet telefonních hovorů v call centru, počet mutací v úseku DNA či počet branek během fotbalového zápasu.[7] Zdrojem algoritmu pro výpočet hodnoty z Poissonova rozdělení je Donald E. Knuth[8].



Obrázek 7 - Poissonovo rušení

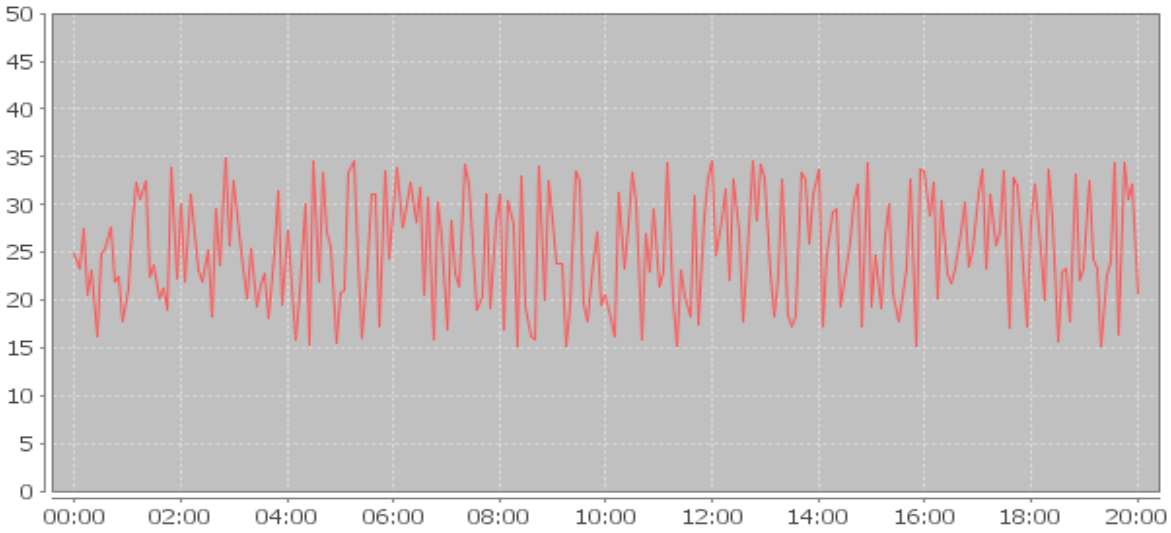
(Nezarušená hodnota 25, Minimum 2, Odchylka 0.5)

[7] http://en.wikipedia.org/wiki/Poisson_distribution#Generating_Poisson-distributed_random_variables
[8] http://stackoverflow.com/questions/1241555/algorithm-to-generate-poisson-and-binomial-random-numbers

2.4.7 Rovnoměrné

Rovnoměrného rozdělení nabývají například chyby při zaokrouhlování čísel, chyby při odečítání údajů z lineárních měřicích přístrojů, doby čekání na uskutečnění jevu opakujícího se v pravidelných intervalech.[9] Náhodný pokus má X možných výsledků, které jsou všechny stejně pravděpodobné. Hodnota rušení R z intervalu omezeného dolní mezí D a horní mezí H je rovna D + (H - D) * <0,1> .

Konkrétní implementace: Funkce má 2 parametry reprezentující minimální a maximální možnou hodnotu, návratová hodnota funkce je z rozmezí určeného těmito parametry, rovnoměrné rozdělení návratových hodnot je zajištěno použitím funkce java.util.Random.nextDouble() která vrátí hodnotu z intervalu od 0 do 1 včetně z rovnoměrného rozdělení pravděpodobnosti.



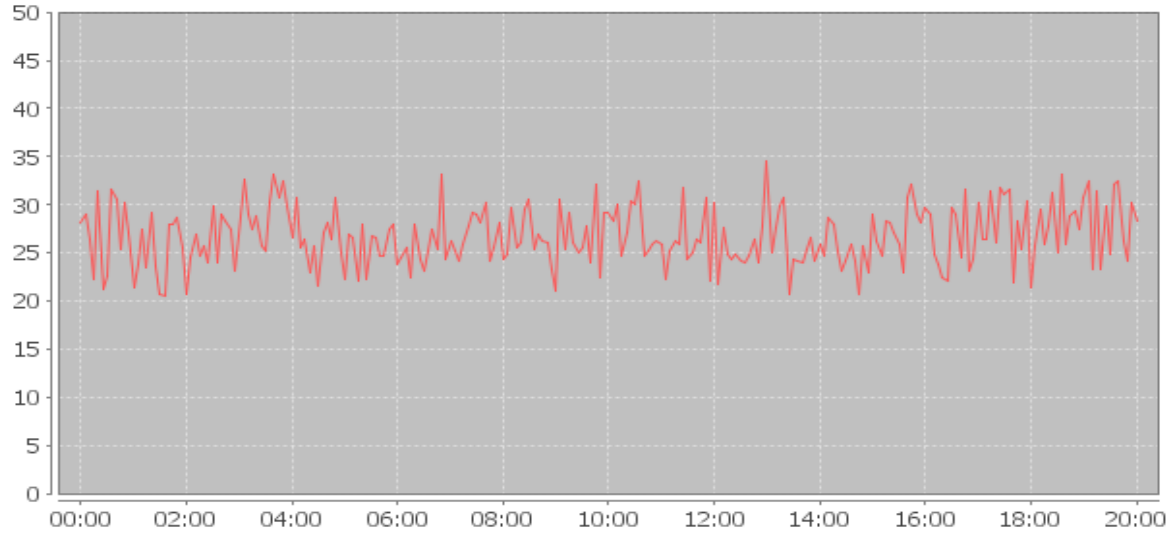
Obrázek 8 - Rovnoměrné rušení

(Nezarušená hodnota 25, Minimum -10, Maximum 10)

2.4.8 Trojúhelníkové

Typickým využitím je popis populace na základě neúplných či nedostatečně rozsáhlých údajů. Je založena na znalosti minima, maxima a odhadu modální hodnoty. Je užitečná v případech, kde jsou vztahy mezi veličinami známé, ale přesná data jsou vzácná/drahá. Používá se jako náhrada Beta distribuce v PERT, CPM a dalších nástrojích pro správu projektů. Zdroj algoritmu pro výpočet hodnoty z trojúhelníkového rozdělení je Brighton Webs Ltd.[10]

[9] http://www.fs.vsb.cz/books/SystAnal/texty/09.htm
[10] http://www.brighton-webs.co.uk/distributions/triangular.asp



Obrázek 9 - Trojúhelníkové rušení

(Nezarušená hodnota 25, Minimum -5, Maximum 10, Střed 0)

3 XML

3.1 Obecně

XML – rozšiřitelný značkovací jazyk – je jedním z nejdůležitějších standardů pro výměnu strukturovaných dat. XML popisuje data nezávisle na platformě ve formě čitelné i lidem. Této univerzálnosti dosahuje použitým formátem dat, jedná se o textové informace a tedy XML soubor je souborem textovým. Textem jsou jak značky, tak samotná data, a součástí souboru je hlavička informující o použitém kódování znaků.

3.2 Struktura XML souboru

XML soubor tvoří stromovou strukturu začínající deklarací (druhá verze jazyka, kódování znaků) s jedním kořenovým elementem a libovolným počtem vnořených elementů. Každý element je ohraničený tagy a musí být uzavřený (u prázdných elementů je možno využít zkrácené formy). Element může obsahovat další atributy – hodnota atributu je uzavřená do uvozovek. V XML souboru je nutné důsledně dodržovat nepřekrývání tagů (nekřížení). Název elementu musí začínat písmenem či podtržítkem; povoleny jsou: písmena, číslice, pomlčky, tečky, podtržítka; zakázány jsou: mezery, při použití jmenných prostorů se používá dvojtečka.

```
<?xml version="1.0"?>
<vzkaz>
  <komu>Tobiáš</komu>
  <od>Hanuko</od>
  <nadpis>Připomínka</nadpis>
  <text>Nezapomeň vyvenčit psa!</text>
</vzkaz>
```

Obrázek 10 - XML soubor

3.3 Porovnání s binárními formáty

Oproti binárním formátům netrpí XML formát omezeným rozsahem čísel, k uchovávání záporných čísel nepotřebuje doplňkový kód a při ukládání čísel odpadají starosti s pořadím bajtů v závislosti na big-endian/little-endian na různých platformách (závisí jak hardware, tak software).

XML je otevřeným formátem, značky jsou doplnitelné podle potřeby, mají pevnou gramatiku (stejně jako HTML) proto lze oproti binárním formátům snadno kontrolovat strukturu a částečně i obsah XML souborů. Tomu odpovídá i skutečnost, že název značky popisuje význam uložených dat (dle autora dokumentu), což je hlavním rozdílem oproti HTML, kde značka určuje, jak se obsah zobrazí. Binární formáty nejsou srozumitelné bez detailní znalosti vnitřní struktury formátu.

Nevýhodou je výrazně větší velikost XML souboru oproti binárnímu souboru při ukládání stejných dat, z toho plyne omezení použití XML formátu pro data s rozsáhlými bitovými sekvencemi – není vhodný pro ukládání audiovizuálních dat, také není optimální pro značně obsáhlé dokumenty (stovky MB). Dle Herouta se jedná o řádový rozdíl v neprospěch XML souborů.

3.4 DTD, XSD schémata

„Pomocí schémových jazyků (též schémat) se definuje nový značkovací jazyk.“ (HEROUT, 2007) Význam schémat spočívá v omezení možností XML podle našich požadavků. Tedy v XML souboru mohou být pouze elementy popsané v příslušném schématu. Tato omezení snižují pravděpodobnost chyb, umožňují lepší kontrolu dat nad rámec prostého formátování dokumentu a usnadňují další práci s daty obsaženými v souboru.

3.4.1 DTD

DTD – definice typu dokumentu – jde o první schémový jazyk, v dnešní době již vytlačován novějšími jazyky, nemá nejlepší pověst – „Je považován za největší chybu ve vývoji XML.“ (HEROUT, 2007) Jedinou výhodou DTD je množství aplikací, které tento jazyk podporují, nevýhody převažují a tento jazyk není vhodný pro datově orientované dokumenty. Mezi nevýhody patří zcela odlišná syntaxe, která znesnadňuje kontrolu správnosti DTD souborů, nemožnost popsat datové typy a absence podpory jmenných prostorů.

```
<!ELEMENT vzkaz (komu, od, nadpis, text)>
<!ELEMENT komu (#PCDATA)>
<!ELEMENT od (#PCDATA)>
<!ELEMENT nadpis (#PCDATA)>
<!ELEMENT text (#PCDATA)>
```

Obrázek 11 - DTD schéma

3.4.2 XSD

XSD – definice schématu XML – jedná se o významný a široce podporovaný schémový jazyk. Oproti DTD dodržuje jazyk XSD konvence XML, což umožňuje snadnou validaci, umožňuje definici vlastních datových typů a podporuje jmenné prostory. Nevýhodou jazyku XSD je složitá specifikace a skutečnost, že u jednodušších XML dokumentů může přesáhnout velikost definičního XSD souboru velikost XML souboru. Obrázek 12 - XSD schéma je XSD schéma popisující XML soubor na Obrázek 10 - XML soubor.

```xml
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="vzkaz">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="komu" type="xs:string"/>
      <xs:element name="od" type="xs:string"/>
      <xs:element name="nadpis" type="xs:string"/>
      <xs:element name="text" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Obrázek 12 - XSD schéma

Konkurenty jazyku XSD jsou především RNG a částečně jazyk Schematron. Jazyk Schematron je v současnosti spíše doplněním jazyků XSD a RNG, jeho síla spočívá v možnosti kontroly obsahu elementů za pomoci jazyka XPath.

```xml
<element xmlns="http://relaxng.org/ns/structure/1.0"
  name="vzkaz">
  <attribute name="komu">
    <text/>
  </attribute>
  <element name="od">
    <text/>
  </element>
  <element name="nadpis">
    <text/>
  </element>
  <element name="text">
    <text/>
  </element>
</element>
```

Obrázek 13 - RNG schéma

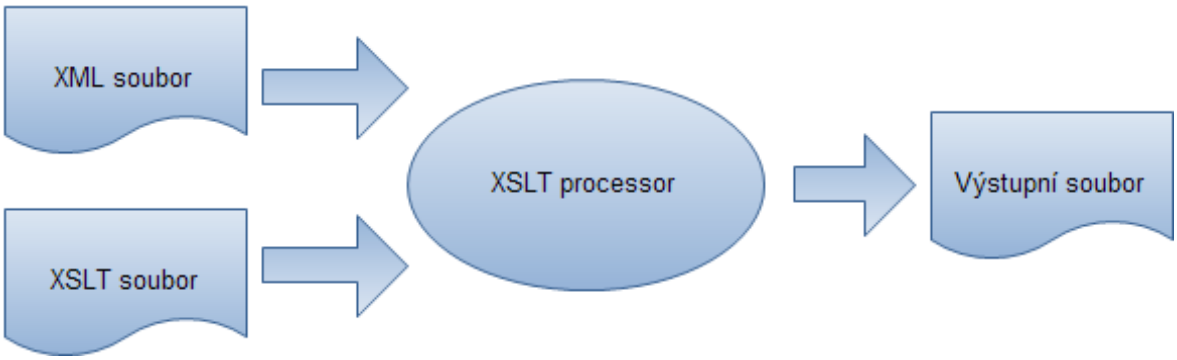
3.5 XPath

Dotazovací jazyk XPath se používá k identifikaci částí XML souboru. Hlavními kritérii, která XPath používá při odkazování, jsou pozice, relativní pozice, typ a obsah elementu/atributu. Výrazy XPath mohou odkazovat např. na první výskyt elementu, na element obsahující zadaný řetězec nebo na konkrétní atribut. Výrazy XPath nacházejí

uplatnění při vyhledání a výběru konkrétních elementů ze vstupního souboru, které mají být dále zpracovány – využívá se při XSL transformaci.

3.6 XSLT

Jednou z výhod XML je důsledné oddělení obsahu od zobrazení, o tom, jak se data reprezentovaná v souboru zobrazí, XML nic neříká, o vzhled se starají stylové jazyky. XSLT je v podstatě šablonou stylu, kterou XSLT procesor porovná s elementy XML souboru. V případě, že je nalezena shoda (pomocí XPath), je zapsaná příslušná část vstupního souboru ve formě určené šablonou do výstupního souboru. Tímto způsobem lze vstupní XML soubor převést do různých formátů např. XML soubor s jinou strukturou, HTML soubor, PDF, RTF, prostý text a další.



Obrázek 14 - XSLT

Tato nezávislost umožňuje dvě zajímavé aplikace – můžeme pro jeden XML soubor mít více transformačních šablon a vhodnou volit podle požadované formy výstupu, a tedy zobrazení přizpůsobit účelu (webové stránky, tisk, PDF, přenosná zařízení, uložení,…), nebo můžeme pro více XML souborů mít stejnou šablonu a tímto způsobem sjednotit jejich zobrazení.

3.7 Validace XML souborů

XML soubor lze snadno ověřovat nezávisle na programu, který bude soubor dále zpracovávat. Toto umožňuje kontrolu dat externím nástrojem a tedy i možnost jednodušší datové vrstvy aplikace, která XML soubor načítá. Odpovědnost za kontrolu dat má jejich poskytovatel, pokud XML soubor neprojde validací, je oprava žádána po poskytovateli – toto je výhodné pro strukturovanou výměnu informací mezi aplikacemi.

Úrovně validace jsou:

1) správná strukturovanost (nekřížení značek, jejich ukončení atd.)
2) správná množina definovaných značek (v DTD/XSD schématu) ve správném pořadí
3) správnost datových typů (XSD schéma)

3.8 Návrh vlastních schémat

Pro návrh schémat trendů jsem se rozhodl využít jazyk XSD, jehož podpora je ve vývojovém prostředí NetBeans (6.9.1) zajištěna pomocí doplňku „XML Schema and WDSL“, a jenž mi umožní mapování do tříd pomocí JAXB – viz část JWSDP, StAX, JAXB.

Obrázek 15 – Schéma pro vstupní trend – design A

Schéma pro vstupní trendy (viz příloha A) jsem navrhl s ohledem na požadované vlastnosti programového generátoru trendu. Vstupní XML soubor popisuje <1,∞) trendů,

tyto trendy se skládají z 2 a více bodů, údaje o požadované výstupní četnosti bodů a názvu trendu. Dále obsahuje vstupní soubor údaje o jednotce, ve které jsou zadané hodnoty bodů, volitelně údaje o rušení a údaje pro zobrazení. Rušení je definováno pomocí typu rušení a vstupních parametrů nutných pro vygenerování daného rušení. Element Sestava umožňuje seskupení více trendů do jedné výstupní sestavy. (Obrázek 1 znázorňující teplotní trendy odpovídá dvěma seskupeným trendům se stejnou jednotkou ve stejné časové výseči.)

Obrázek 16 - Schéma pro výstupní trend – design B

Pro výstupní trendy jsem navrhl dvě schémata – první využívá hierarchický model vhodný pro znázornění celého trendu (viz příloha B), druhé je zaměřeno na dynamický výstup bodů trendu (viz příloha C). Schéma hierarchického modelu je velmi podobné schématu vstupních trendů, liší se pouze absencí údajů o rušení. Výstupní XML soubor podle dynamického schématu se skládá z jednoho bodu a volitelně z údajů popisujících trend (stejné jako u hierarchického schématu). To, do kterého trendu bod patří, pokud neobsahuje informace o trendu, určují atributy s informacemi o sestavě a názvu.

Obrázek 17 - Schéma pro výstupní trend - design C

3.9 Práce s XML soubory

Výhodou formátu XML je snadné ověření XML dokumentů – pokud je při načítání XML dokumentu validací zjištěna chyba (úroveň validace závisí např. na dostupnosti XSD souboru), s dokumentem se dále nepracuje, odpovědnost za data v XML dokumentu náleží subjektu, který XML dokument vytvořil. Toto umožňuje značné zjednodušení datové vrstvy aplikace proti aplikacím využívajícím jiné způsoby načítání dat.

4 XML a Java

4.1 Parsery

Parser je nástroj, který načte a rozdělí XML dokument na jednotlivé části – elementy, atributy, hodnoty. Parsery v libovolném programovacím jazyku by měli umožňovat načtení XML dokumentu ze souboru nebo vstupního proudu. Při načítání by mělo automaticky docházet ke kontrole správné struktury dokumentu, volitelně i k validaci proti schématu. Během čtení by měl zpracovat nejen části dokumentu ale i pomocná data z XML dokumentu (např. komentáře). Parser by měl poskytovat rozhraní pro přístup k načtenému dokumentu tzv. infosetu – dále se pracuje přímo s elementy, atributy a hodnotami nikoliv s jejich pouhou textovou reprezentací.

Parsery lze rozdělit podle rozhraní na 2 skupiny. První jsou parsery s proudovým zpracováním, druhá jsou parsery se stromovou reprezentací. Mezi parsery s proudovým zpracováním patří SAX a StAX. Stromovou reprezentaci využívají DOM a JAXB.

4.2 JAXP, JDOM

JAXP a JDOM jsou rozhraní parserů pro Javu.

JAXP je začleněno do Java Core API, definuje rozhraní pro práci s XML dokumenty, sjednocuje používání existujících parserů a umožňuje stejný způsob práce pro odlišné způsoby zpracování XML dokumentů. „Využíváme-li rozhraní JAXP, jsme odstíněni od drobných odlišností parserů a jejich verzí.“ (HEROUT, 2007)

JDOM není začleněno do Java Core API, zjednodušuje obecné DOM rozhraní, je navržen přímo pro Javu a tedy používá její typické vlastnosti (kolekce, přetěžování metod). JDOM dokáže číst ze SAX i DOM zdrojů a také odesílat výsledky jako DOM strom či SAX proud.

4.3 SAX

Rozhraní pro SAX jsou součásti Java Core API, běžně se využívá varianta odstíněná přes JAXP. SAX zpracovává XML dokument proudově, tedy pracuje s částmi XML dokumentu – při jejich načtení dochází k událostem, program na tyto události reaguje odpovídajícími metodami. V SAX není možnost návratu k již načteným částem dokumentu v průběhu parsování, v případě potřeby je nutno znovu zahájit parsování od začátku.

Výhodou SAX parserů je nízká paměťová náročnost a obvykle i vyšší rychlost při zpracovávání XML dokumentů, to umožňuje načtení rozsáhlých XML dokumentů (takových, se kterými mohou mít DOM parsery potíže).

4.4 DOM

Rozhraní pro DOM jsou součásti Java Core API, stejně jako u SAX se běžně využívá varianta odstíněná přes JAXP. DOM načítá celý XML dokument do paměti ve formě stromová reprezentace, jednotlivé uzly stromu jsou objekty (tzv. nody). Typy uzlů jsou uvedeny níže v Tabulka 1. Nevýhodou DOM jsou vysoké paměťové nároky a pomalejší načítání větších XML dokumentů. Výhodou je možnost opakovaného přístupu k již načteným částem – „… při zpracování údajů můžeme libovolně vracet, přeskakovat nepotřebné části, tj. obecné se pohybovat po stromu.“ (HEROUT, 2007). V paměti lze XML dokument upravit a pomocí DOM zapsat do souboru.

Tabulka 1 - Typy DOM uzlů[11]

Typ uzlu	Popis	Potomci
Document	Reprezentuje celý dokument	Element (jeden), ProcessingInstruction, Comment, DocumentType
DocumentFragment	Reprezentuje odlehčenou část dokumentu - fragment	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
DocumentTypev	Rozhraní pro entity definované v dokumentu	-
ProcessingInstruction	Instrukce pro zpracování	-
EntityReference	Reprezentuje referenci na entitu	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
Element	Reprezentuje jeden element	Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference
Attr	Reprezentuje jeden atribut	Text, EntityReference
Text	Text v elementu či atributu	-
CDATASection	Reprezentuje v dokumentu sekci CDATA, nezpracovává ji parser	-
Comment	Komentář	-
Entity	Reprezentuje jednu entitu	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
Notation	Notace deklarovaná v DTD	-

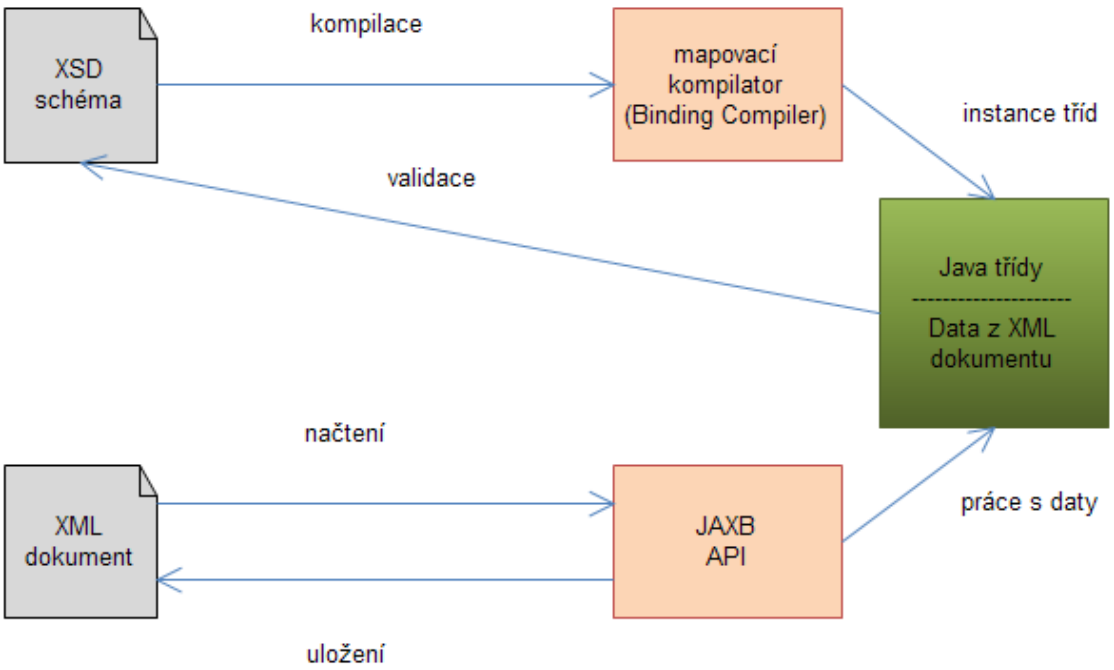
[11] Překlad z tutoriálu http://www.w3schools.com/dom/dom_nodetype.asp

4.5 JWSDP, StAX, JAXB

JWSDP obsahuje důležité technologie pro webové služby, mezi nimi se mimo jiné nacházejí i StAX a JAXB. Od verze Javy 1.6 se tyto technologie nacházejí již v Java Core API.

StAX (dříve také znám jako JSR 173, Zephyr či SJSXP) je rozhraní pro sekvenční čtení a zápis XML dokumentů. StAX nabízí dvě možnosti jak přistupovat k XML dokumentům – kurzorově a událostně. Jedná se o velmi rychlé a jednoduché rozhraní, které umí XML dokumenty jak načítat tak vytvářet. Stejně jako SAX může být využíván pro načtení rozsáhlých XML dokumentů, ale práce s textovým obsahem je jednoduší. StAX načítá XML dokument na vyžádání (pamatuje si poslední pozici) – na rozdíl od SAX, kde je nutné načíst celý dokument. StAX nevaliduje XML dokumenty a případné uvedené schéma ignoruje. Jeho jednoduchost je výhodou i při zápisu XML dokumentů – pomocí StAX je možné vytvořit XML dokumenty značně větší než pomocí DOM.

JAXB se liší od dříve zmíněných parserů. XML dokument je načten a přemapován do odpovídajících tříd – využívají se detailní informace z XSD schéma popisujícího dokument k automatickému vygenerování odpovídajících tříd. JAXB je vhodné pokud dopředu známe XSD schéma, XML dokument je strukturovaný (případně s opakujícími se informacemi), dokument je potřeba načítat a upravovat a pokud dokument není příliš velký. Výhodou JAXB přístupu je skutečnost, že po načtení dokumentu (unmarshall) již pracujeme odstínění od XML pouze s třídami v Javě. Načtený objekt lze snadno upravovat (např. setry a getry) a validovat. Opačný proces se nazývá marshall, jedná se o převedení dat z tříd zpět do XML dokumentu.

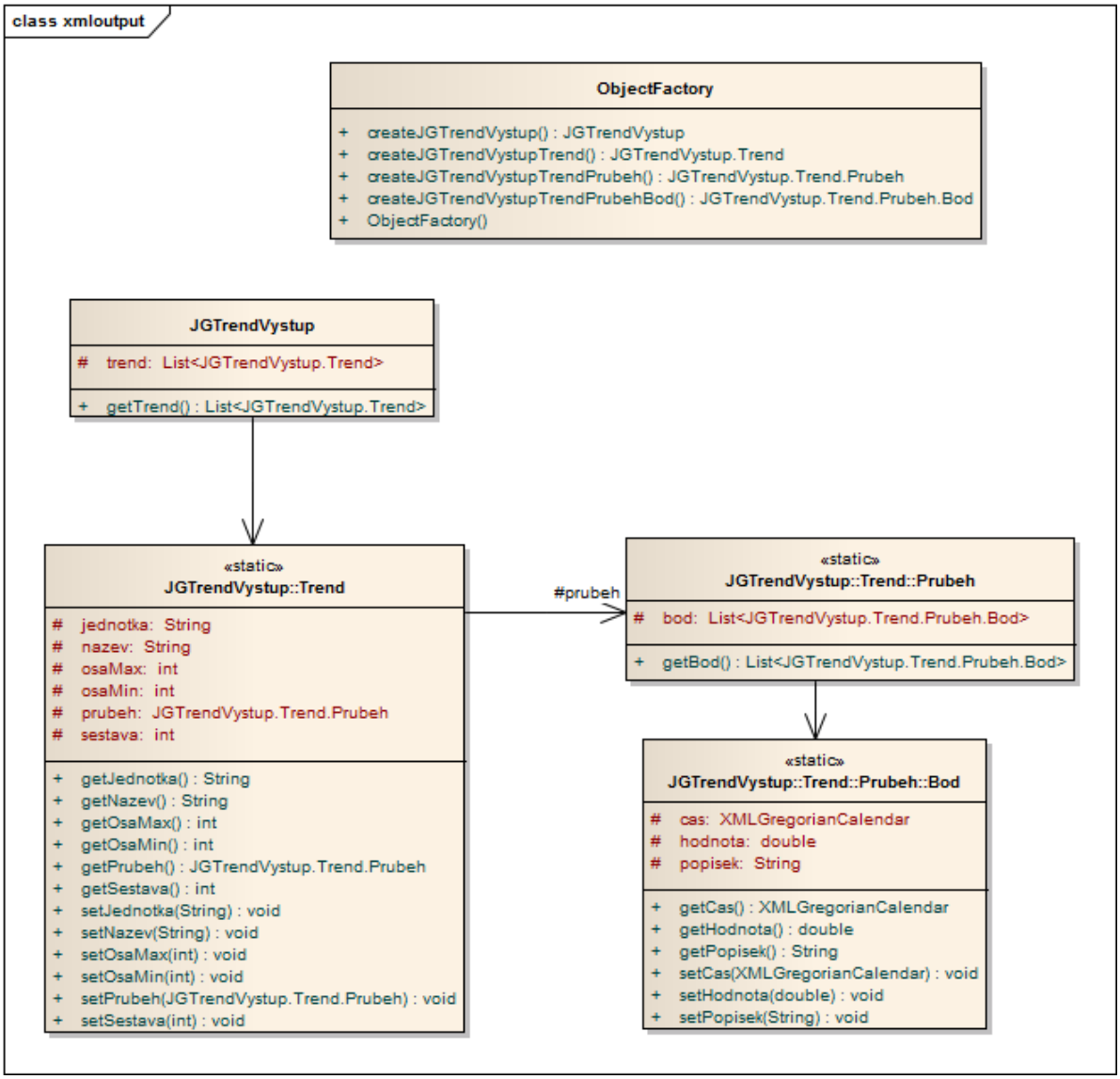


Obrázek 18 - JAXB

Generování tříd z XSD schéma má na starosti schema compiler (XJC), tento převod je jednorázový, dále se pracuje s takto vytvořenými třídami. Níže uvedená Tabulka 2 zachycuje některé příklady mapování prvků z XSD schéma

Tabulka 2 - XJC prvky

XSD prvek	Java prvek
xsd:string	java.lang.String
xsd:dateTime	javax.xml.datatype.XMLGregorianCalendar
xsd:double	double
xsd:unsignedInt	long
xsd:boolean	boolean
xsd:anySimpleType	java.lang.Object/ java.lang.String
xsd:decimal	java.math.BigDecimal



Obrázek 19 – XJC převod z XSD schéma pro výstupní trendy (Příloha B)

Konkrétní implementace: Pro zpracování XML dokumentů jsem zvolil JAXB na základě toho, že byly splněny podmínky, při kterých je použití JAXB výhodné – XML dokumenty jsou strukturované, aplikace je bude načítat, upravovat a ukládat, a jejich XSD schémata jsou známá dopředu. Obrázek 19 ukazuje, jak vypadají vygenerované třídy pro dříve výstupní schéma (grafická reprezentace schématu viz Obrázek 16). Součástí vygenerovaných tříd je i třída ObjectFactory – jejím účelem je tvoření nových instancí mapovaných tříd. Třídy jsem samozřejmě vygeneroval i pro vstupní schéma. Tyto vytvořené třídy využívám ve vlastních třídách, jejichž popis následuje v části Implementace.

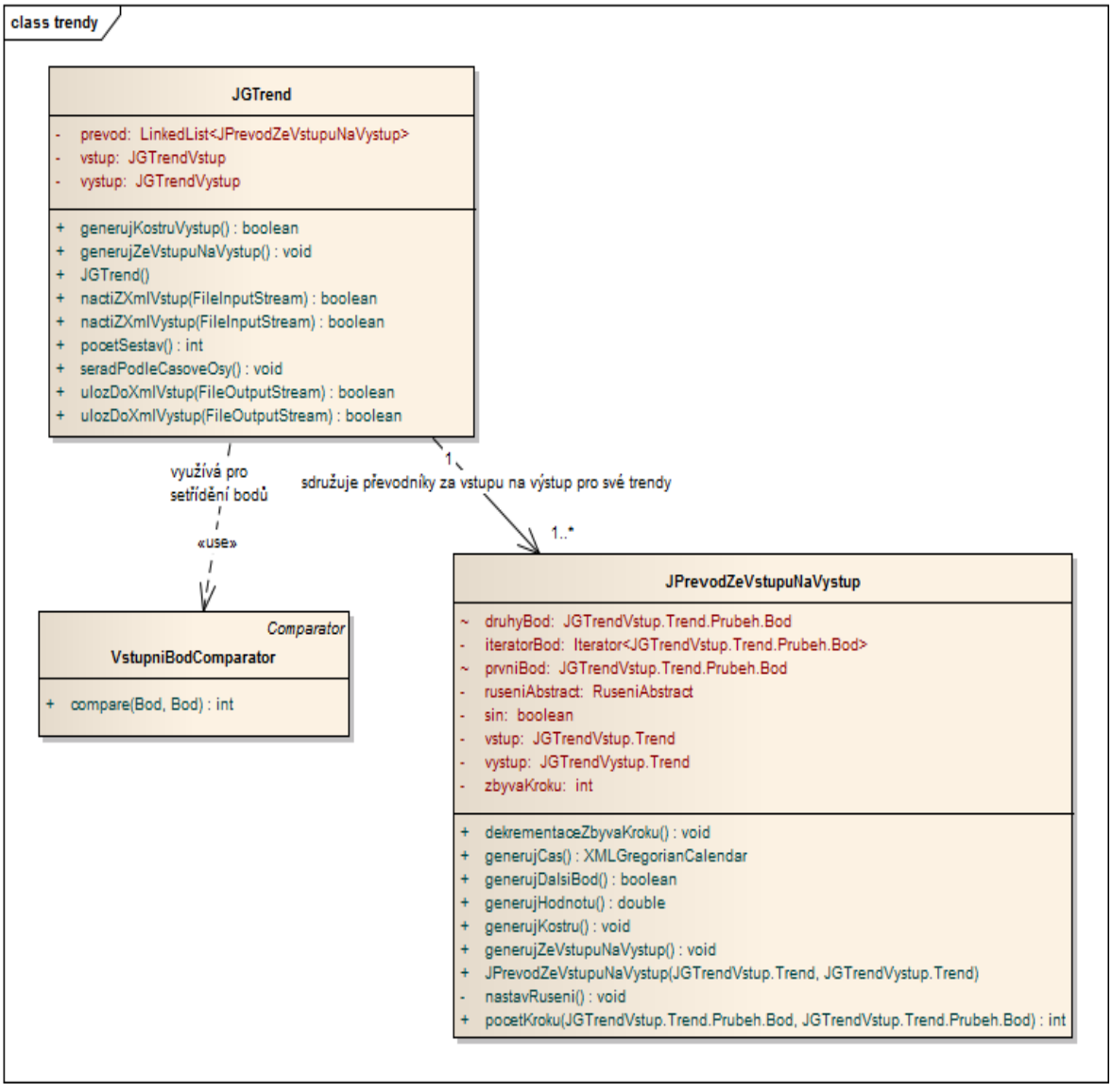
5 Implementace

Programový generátor trendů jsem implementoval dle zásad objektově orientovaného programování za pomoci vývojových iteračních cyklů. Třídy jsou seskupeny v balíčcích a jejich metody jsou okomentovány dle pravidel javadoc.

5.1 Balíčky tříd a jejich vztahy

5.1.1 Trendy

Balíček trendy (odpovídá první iteraci – převod vstupního trendu na výstupní) obsahuje třídy JGTrend, VstupniBodComparator a JPrevodZeVstupuNaVystup.



Obrázek 20 - Trendy a převodník

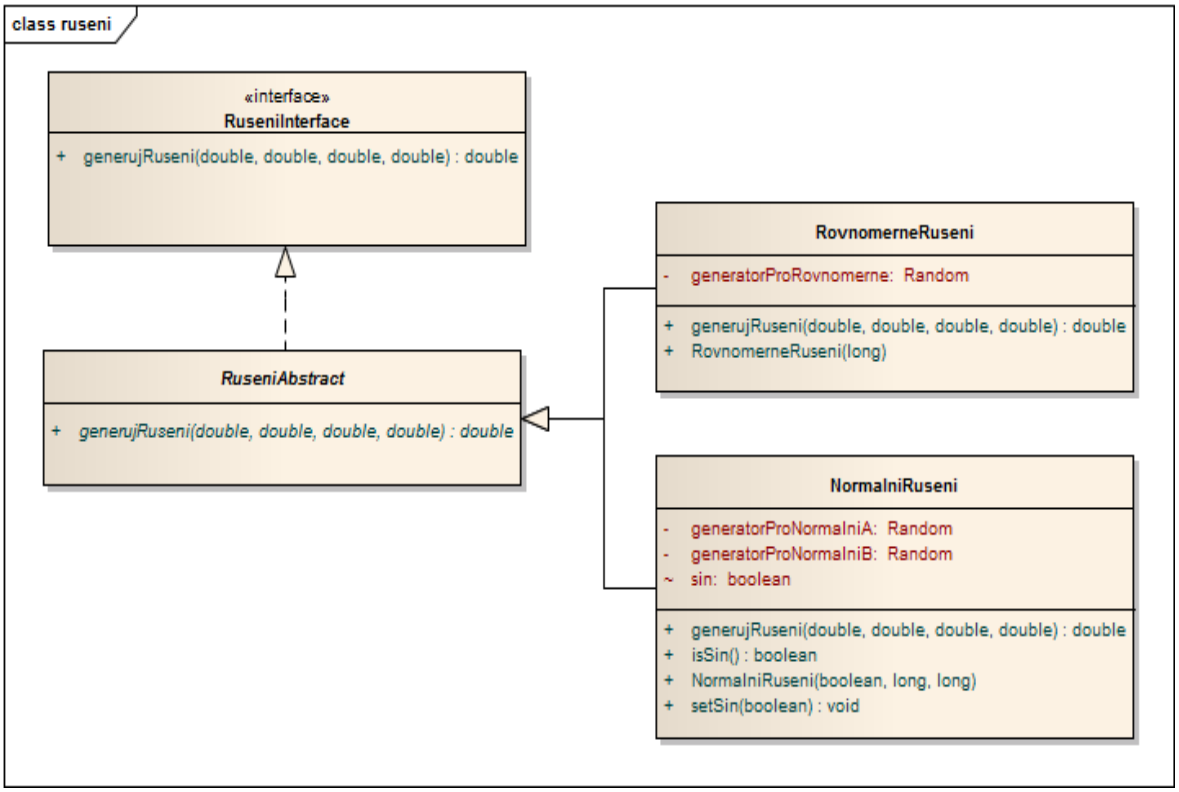
Třída JGTrend se stará o interakce s vstupním a výstupním XML souborem, vytváří pro jednotlivé trendy převodníky (instance třídy JPrevodZeVstupuNaVystup) a obsahuje proceduru pro převedení trendů.

Třída JPrevodZeVstupuNaVystup se stará o samotný převod jednoho trendu včetně rušení. Převod probíhá postupně - mezi jednotlivými vstupními body je vygenerován odpovídající počet výstupní bodů se zadanou četností, jejich hodnota je dopočtena ze spojnic hodnot mezi body a případné hodnoty rušení z daného rozdělení pravděpodobnosti.

Třída VstupniBodComparator slouží k setřídění kolekce vstupních bodů podle času.

5.1.2 Rušení

Balíček ruseni obsahuje interface RuseniInterface, abstraktní třídu RuseniAbstract, která implementuje RuseniInterface, a potomky třídy RuseniAbstract – pro jednotlivá rušení – RovnomerneRuseni, NormalniRuseni, AlternativniRuseni a další.



Obrázek 21 - Rušení

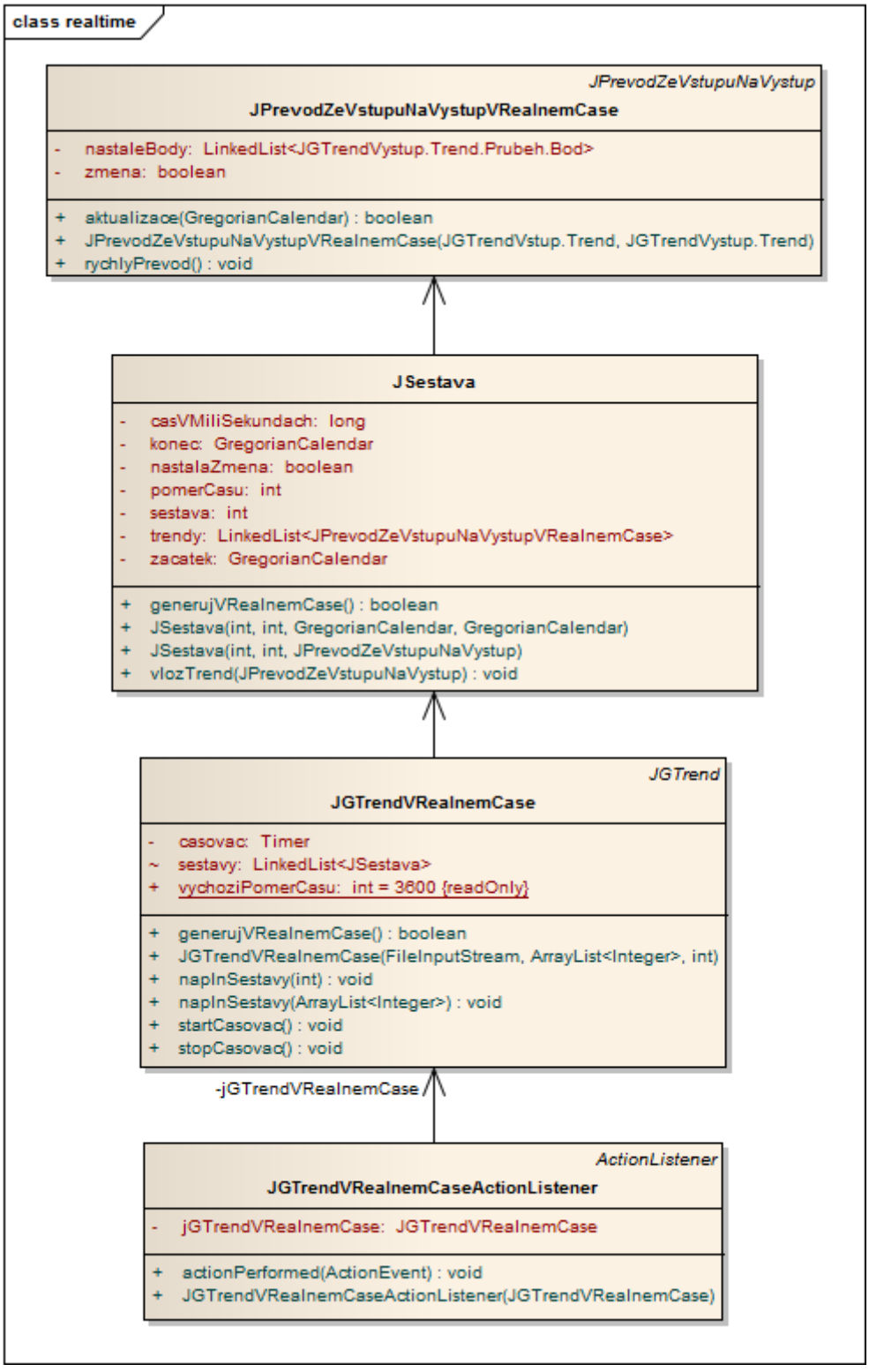
RuseniInterface obsahuje pouze funkci generujRuseni(), která vygeneruje hodnotu spadající do rozdělení pravděpodobnosti dle konkrétního implementovaného rozdělení.

Abstraktní třída RuseniAbstract umožňuje vybrat správnou třídu (jednoho z potomků) za běhu programu.

Třídy AlternativniRuseni, BinomickeRuseni, GammaRuseni, NormalniRuseni, HypergeometrickeRuseni, PoissonovoRuseni, RovnomerneRuseni a TrojuhelnikoveRuseni jsou potomky třídy RuseniAbstract, pro vygenerování rušení obsahují instance třídy Random a v zděděné funkci generujRuseni() algoritmus pro dané rozdělení.

5.1.3 Realný čas

Balíček realtime (odpovídá druhé iteraci – převod vstupu na výstup postupně v čase) obsahuje třídy JGTrendVRealnemCaseActionListener, JGTrendVRealnemCase, JPrevodZeVstupuNaVystupVRealnemCase a JSestava.



Obrázek 22 - Převod trendu v reálném čase

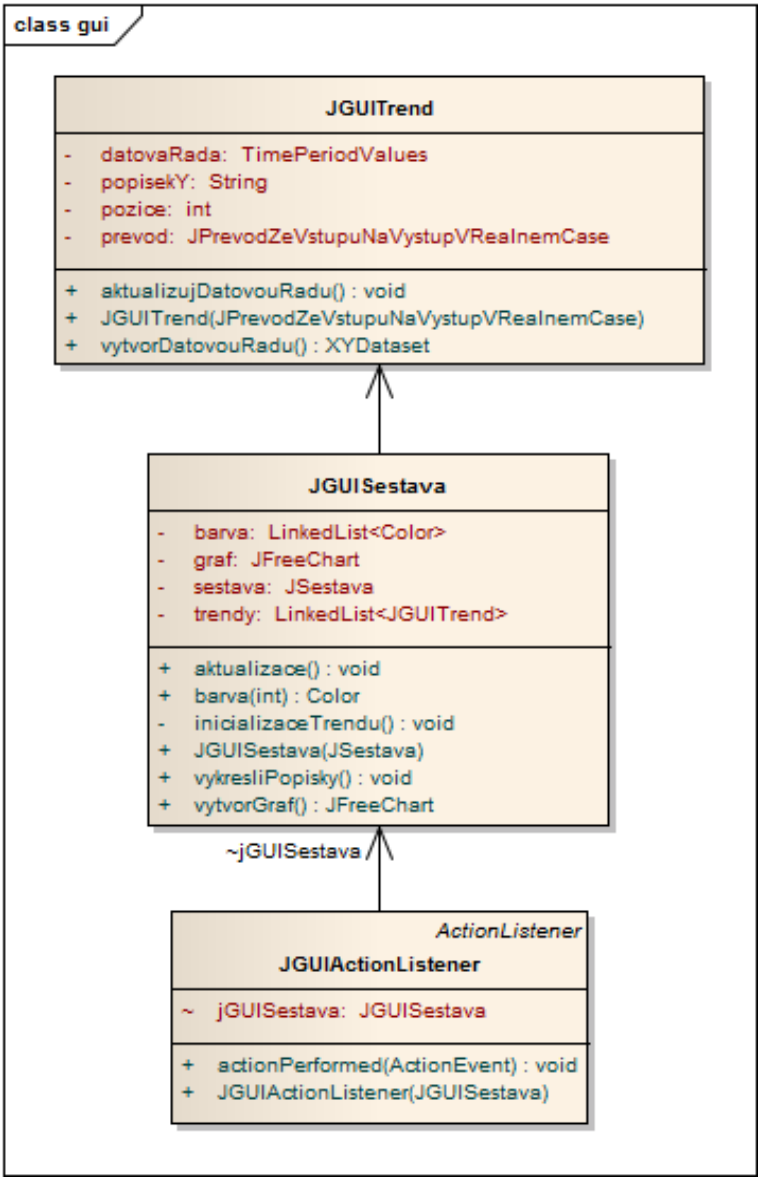
Třída JGTrendVRealnemCase je potomkem třídy JGTrend, rozšiřuje její možnosti o seskupení trendů do sestav (instancí JSestava) a o generování výstupních trendů v čase. K tomu využívá časovač (instanci třídy Timer); procedury startCasovac() a stopCasovac() a posluchač událostí časovače JGTrendVRealnemCaseActionListener. Posluchač událostí je asociován s instancí JGTrendVRealnemCase, ve své proceduře actionPerformed() volá funkci generujVRealnemCase() asociované instance a zastavuje časovač při dosažení konce všech sestav. Funkce generujVRealnemCase() generuje body pro uplynulý časový úsek pro všechny sestavy, její návratová hodnota je rovna true dokud nejsou sestavy u konce.

Třída JSestava seskupuje trendy (resp. Jejich převodníky pro reálný čas) náležící do stejné sestavy (instance třídy JPrevodZeVstupuNaVystupVRealnemCase), obsahuje atributy potřebné pro převod v reálném čase – počátek a konec trendů v sestavě; poměr času; aktuální čas v průběhu převodu. Stejně jako třída JGTrendVRealnemCase obsahuje funkci generujVRealnemCase(), která generuje body pro uplynulý časový úsek pro všechny trendy sestavy, její návratová hodnota je rovna true dokud nejsou sestavy u konce.

Třída JPrevodZeVstupuNaVystupVRealnemCase je potomkem třídy JPrevodZeVstupuNaVystup, rozšiřuje její možnosti o převod trendu v reálném čase. Toho je dosaženo LinkedListem nastalých bodů trendu a atributem obsahujícím informaci o změně. Samotný převod realizuje funkce aktualizace(), jejímž parametrem je kalendářní údaj – místo v trendu ke kterému má být trend vygenerován; návratová hodnota je rovna true dokud není trend u konce.

5.1.4 Graf

Balíček gui (odpovídá třetí iteraci – postupné zobrazení převedeného trendu pomocí knihovny JFreeChart) obsahuje třídy JGUIActionListener, JGUISestava a JGUITrend.



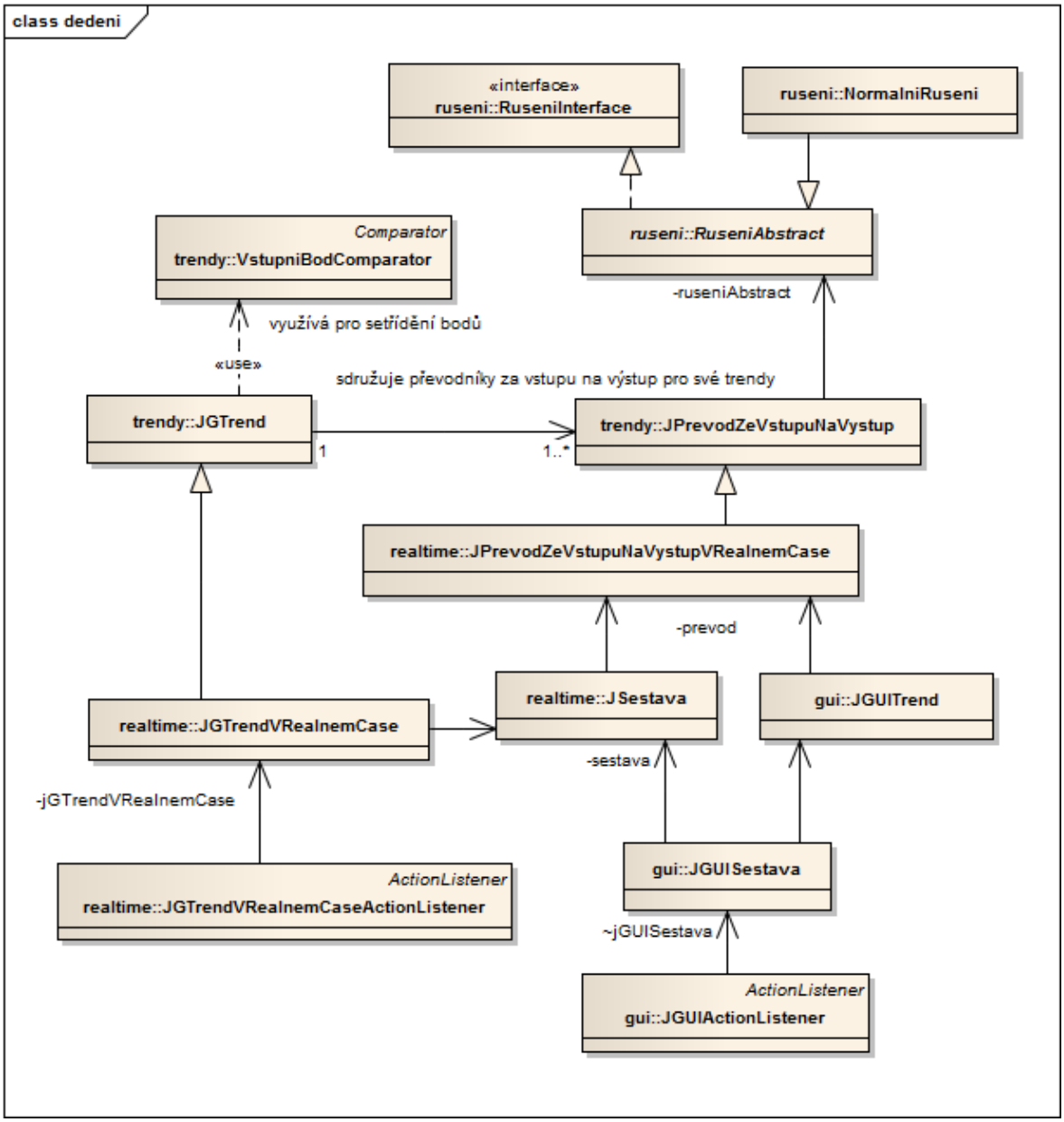
Obrázek 23 - Vytvoření grafu z trendů

Třída JGUISestava se stará o převedení trendů sestavy (instance třídy JGUITrend) na grafy (instance třídy JFreeChart). Funkce vytvorGraf() kompletuje datové řady jednotlivých trendů (instance třídy TimePeriodValues) do jednoho grafu, trendy jsou odlišeny barevně včetně legendy; trendy mají vlastí osy a popisky veličin. Aktualizaci grafu v reálném čase obstarává posluchač událostí časovače JGUIActionListener. Posluchač událostí JGUIActionListener naslouchá stejnému časovači jako posluchač událostí JGTrendVRealnemCaseActionListene; je asociován s instancí JGUISestava; ve své proceduře actionPerformed() volá proceduru aktualizace() asociované instance. Procedura aktualizace() doplní datové řady trendů o nově vygenerované body vzniklé pomocí funkce aktualizace() třídy JPrevodZeVstupuNaVystupVRealnemCase.

Třída JGUITrend převádí nastalé body trendu získané z převodníku (instance JPrevodZeVstupuNaVystupVRealnemCase) do formy datové řady.

5.1.5 Vzájemné vztahy

Pro lepší orientaci popisuje níže uvedený obrázek (Obrázek 24) asociace a dědičnosti výše zmíněných tříd.



Obrázek 24 - Vzájemné vztahy tříd

5.2 JFreeChart

JFreeChart je knihovna pro tvorbu grafů napsaná v jazyce Java, dostupná pod LGPL licencí. Tato knihovna implementuje mnoho typů grafů, grafy podporují Swingové komponenty a převod na vektorovou či bitmapovou grafiku. JFreeChart lze

použít v aplikacích, appletech, servletech a JSP.[12] Aktuální a tedy i použitou verzí knihovny je 1.0.13, dále je nutno použít knihovnu JCommon (verze 1.0.16).

Konkrétní implementace: Použil jsem graf typu TimeSeriesChart – typ grafu používaný pro kolekce dat reprezentující čas (osa X) a hodnotu (osa Y). V případě sestavy s více trendy obsahuje vytvořený graf odpovídající počet Y os pro hodnoty bodů v trendech. Jednotlivé trendy jsem odlišil barevně a orientaci usnadňuje legenda. Pro zobrazení grafu jsem využil Swingové komponenty. Zobrazení v reálném čase je řešeno přidáním nově nastalých bodů do kolekce zdrojových dat pro graf.

5.3 Generování v reálném čase

První důležitou hodnotou je odstup bodů v trendu – je určen ve vstupním XML dokumentu. Druhou důležitou hodnotou je poměr reálného času proti času generovaného trendu – tento poměr určuje proměnná pomerCasu v třídě JSestava pro všechny trendy zahrnuté v dané sestavě.

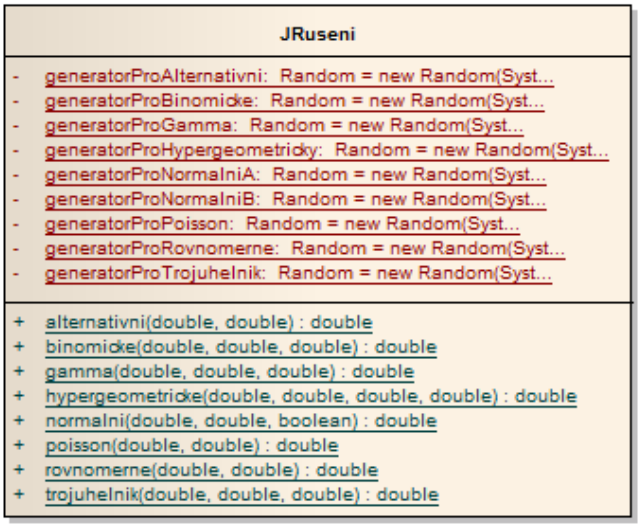
V navržené podobě je minimální odstup mezi body v trendu 1 sekunda, výchozí přepočet pro sestavy je 1 sekunda reálného času rovna 3600 sekundám (1 hodině) v sestavě. V současné podobě nelze pracovat v řádu menším, než jsou sekundy, granularita časovače je 1 milisekunda, takže případná modifikace je možná, pro práci v nanosekundách by bylo nutno implementovat nanosekundový časovač s nižší granularitou.

Časovače trendů jsou použity nejen pro JGTrendVRealnemCaseActionListener starající se o pravidelné generování bodů, ale i pro JGUIActionListener mající na starosti aktualizaci výstupního grafu zachycujícího průběh trendů.

5.4 Statické a dynamické třídy rušení, singleton

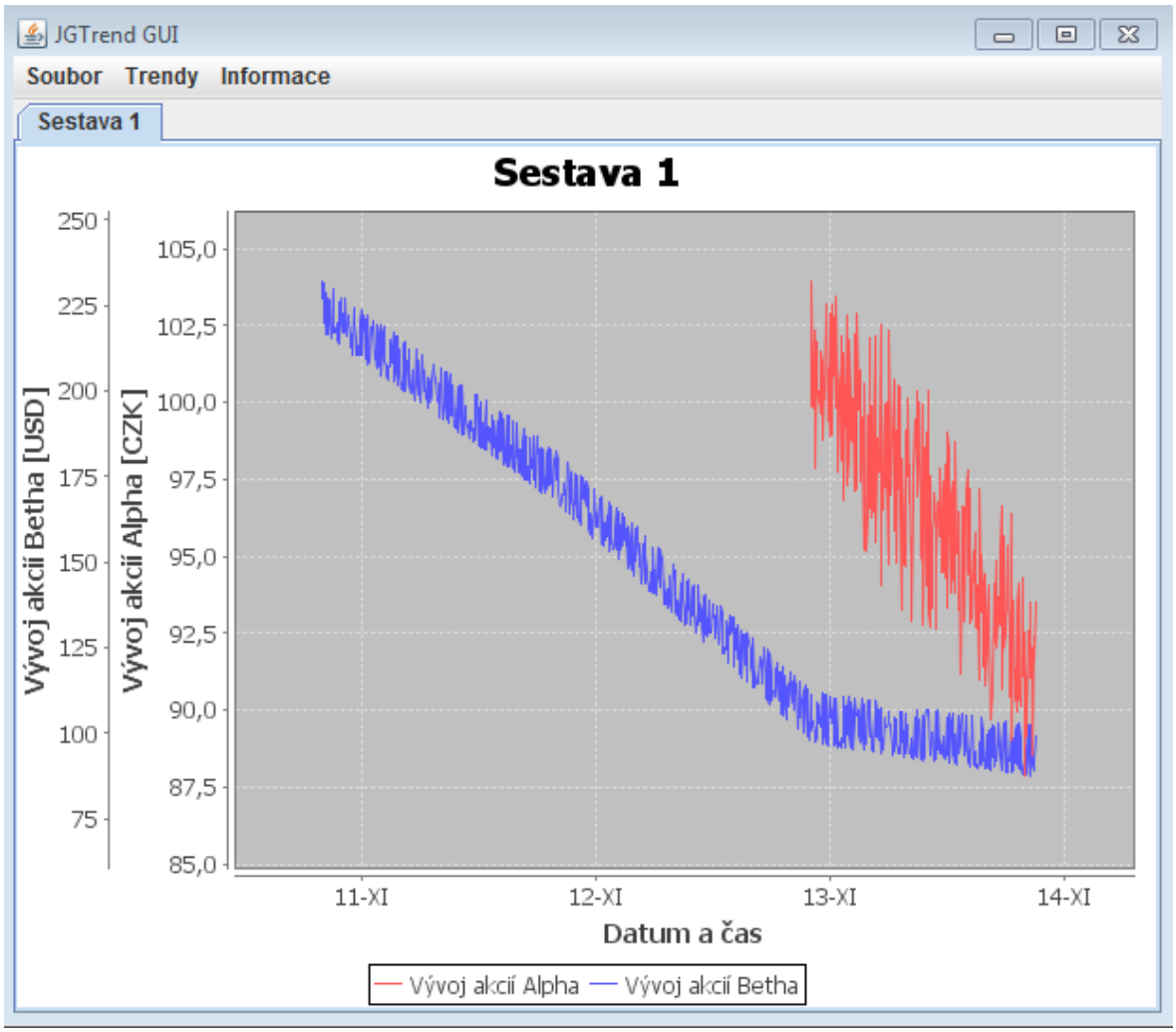
V části Implementovaná rozdělení pravděpodobnosti jsou popsány algoritmy pro generování hodnot z těchto rozdělení. Původně jsem implementoval tyto algoritmy do statických metod třídy JRuseni (viz Obrázek 25). Výhodou byla možnost využívat metody bez potřeby instance této třídy. Později jsem se rozhodl přiklonit k dynamickým třídám a využití polymorfismu vzhledem k lepším možnostem znovupoužití zdrojového kódu a snadnější rozšiřitelnosti o další rozdělení pravděpodobnosti. Toho je dosaženo pomocí abstraktní třídy implementující rozhraní obsahující metodu pro získání hodnoty rušení (viz Obrázek 21). Další zvažovanou možnosti byl návrhový vzor singleton, ale vzhledem k jeho problematické implementaci v jazyce Java jsem od této možnosti upustil[13].

[12] http://www.jfree.org/jfreechart/
[13] http://www.javaworld.com/javaworld/jw-04-2003/jw-0425-designpatterns.html



Obrázek 25 - Statická třída JRuseni

5.5 Swingová aplikace



Obrázek 26 - Okno aplikace

Pro implementaci GUI jsem použil Swing v kombinaci s JFreeChart knihovnou pro vykreslení průběhu trendů. Hlavním ovládacím prvkem je aplikační menu obsahující položky seřazené dle kategorií. V nabídce Soubor se nacházejí metody pro načtení a uložení XML dokumentů, a pro ukončení aplikace. V nabídce Trendy jsou metody pro spuštění a zastavení generování trendu, pro generování celého trendu najednou, dále metoda pro transformaci vstupního trendu pomocí XSLT do HTML dokumentu a jeho následné zobrazení. V nabídce Trendy je též dynamicky generovaná struktura informující o aktuálních nastaveních poměrů časů pro jednotlivé sestavy a obsahující metodu pro zobrazení dialogu umožňujícího změnit tyto poměry v průběhu generování. V nabídce Informace jsou metody pro zobrazení informací o aplikaci a pro zobrazení nápovědy.

Samotné vykreslení průběhu grafu je realizováno na swingovou komponentu JTabbedPane, pro sestavy ze vstupního trendu je vytvářen odpovídající počet záložek, každá z nich obsahuje právě jeden JFreeChart graf zachycující průběh generovaných trendů. Graf je generován třídou JGUISestava z balíčku gui.

Swingová aplikace je umístěna v balíčku gui.swing. Součástí balíčku jsou kromě hlavní třídy JGTrendGUI i třídy PomerCasuDialog, InformaceDialog, XSLTDialog a XMLFileFilter. PomerCasuDialog je třída implementující dialog pro změnu poměru reálného času ke generovanému v sestavách, třída InformacDialog zobrazuje informace o aplikaci, třída XSLTDialog má na starosti XSLT transformaci XML dokumentu a jeho zobrazení ve formě HTML a třída XMLFileFilter slouží pro odfiltrování nevyhovujících souborů při načítání – akceptovány jsou pouze soubory s příponou xml.

6 Závěr

Tato bakalářská práce měla za cíl realizaci softwarového generátoru časových trendů podle scénářů zadaných lomenou čarou.

V teoretické části jsem popsal pojem trend, věnoval jsem pozornost popisu procesu generování hodnot a s tím souvisejícím rušením a tedy jsem popsal jednotlivá rozdělení pravděpodobnosti včetně algoritmů. Dále jsem v teoretické části shrnul informace o XML formátu a k němu přidružením technologiím jako jsou schémové jazyky (DTD, XSD, RNG), dotazovací jazyk XPath, transformační jazyk XSLT. Neopomněl jsem ani obecné informace o práci s XML dokumenty a jejich validaci. Posléze jsem zpracoval přehled integrace technologie XML do jazyka Java, který zahrnuje popis parserů a rozhraní implementovaných v Javě (SAX, DOM, StAX, JAXB, JAXP, JDOM).

V praktické části jsem realizoval programový generátoru trendů splňující kritéria zadání, jeho vývoj probíhal v iteračních cyklech s využitím technik z OOP ve vývojovém prostředí NetBeans 6.9.1. Součástí realizace jsou XSD schémata pro XML dokumenty trendů, zdrojové kódy v jazyce Java tvořící dohromady Java aplikaci postavenou nad Swingem a JFreeChartem a dokumentace k aplikaci. Aplikace generuje trendy v reálném čase a zobrazuje jejich průběh. V implementaci generování v reálném čase je prostor pro případné zjemnění granularity při případném využití nanosekundových časovačů. Byť nad rámec zadání, považuji za neúspěch nefunkčnost XSLT transformace v aplikaci i přes skutečnost že ve vývojovém prostředí je XSL šablona validována a otestována nad stejným XML dokumentem.

Osobním přínosem této práce pro mě byla možnost projít vývojovým cyklem aplikace v jazyce Java. Dále jsem se seznámil s velkým množstvím nových technologií, např. XSD, XSLT, XPath, JAXB a měl jsem možnost aplikovat znalosti a postupy z oblasti OOP.

Literatura

GUNDERLOY, Mike. 2007. *Z kodéra vývojářem : Nástroje a techniky pro opravdové programátory.* Brno : Computer Press, 2007. ISBN 978-80-251-1517-6.

HEROUT, Pavel. 2007. *Java a XML*. České Budějovice : Kopp, 2007. ISBN 978-80-7232-307-4.

PAGE-JONES, Meilir. 2001. *Základy objektově orientovaného návrhu v UML.* Praha : Grada, 2001. ISBN 80-247-0210-X.

PATTON, Ron. 2002. *Testování softwaru.* Brno : Computer Press, 2002. ISBN 80-7226-636-5.

PECINOVSKÝ, Rudolf. 2007. *Návrhové vzory : 33 vzorových postupů pro objektové programování.* rno : Computer Press, 2007. ISBN 978-80-251-1582-4.

Příloha A – XSD schéma pro vstupní trendy

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://xml.netbeans.org/schema/XMLInput"
  xmlns:tns="http://xml.netbeans.org/schema/XMLInput"
  elementFormDefault="qualified">
  <xsd:element name="JGTrend-vstup">
    <xsd:complexType>
      <xsd:sequence maxOccurs="unbounded">
        <xsd:element name="Trend">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Nazev" type="xsd:string"/>
              <xsd:element name="OsaMin" type="xsd:int"/>
              <xsd:element name="OsaMax" type="xsd:int"/>
              <xsd:element name="Jednotka" type="xsd:string"/>
              <xsd:element name="Cetnost" type="xsd:int"/>
              <xsd:element name="Sestava" type="xsd:int"/>
              <xsd:element name="Ruseni" minOccurs="0">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="Typ" type="xsd:string"/>
                    <xsd:element name="Min" minOccurs="0"
type="xsd:double"/>
                    <xsd:element name="Stred" minOccurs="0"
type="xsd:double"/>
                    <xsd:element name="Max" minOccurs="0"
type="xsd:double"/>
                    <xsd:element name="Odchylka" minOccurs="0"
type="xsd:double"/>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
              <xsd:element name="Prubeh">
                <xsd:complexType>
                  <xsd:sequence maxOccurs="unbounded" minOccurs="2">
                    <xsd:element name="Bod">
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="Cas" type="xsd:dateTime"/>
                          <xsd:element name="Hodnota" type="xsd:double"/>
                          <xsd:element name="Popisek" type="xsd:string"/>
                        </xsd:sequence>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Příloha B – XSD schéma pro výstupní trendy 1. varianta

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://xml.netbeans.org/schema/XMLOutput"
  xmlns:tns="http://xml.netbeans.org/schema/XMLOutput"
  elementFormDefault="qualified">
  <xsd:element name="JGTrend-vystup">
    <xsd:complexType>
      <xsd:sequence maxOccurs="unbounded" minOccurs="1">
        <xsd:element name="Trend">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Nazev" type="xsd:string">
              </xsd:element>
              <xsd:element name="OsaMin" type="xsd:int"/>
              <xsd:element name="OsaMax" type="xsd:int"/>
              <xsd:element name="Jednotka" type="xsd:string">
              </xsd:element>
              <xsd:element name="Sestava" type="xsd:int"/>
              <xsd:element name="Prubeh">
                <xsd:complexType>
                  <xsd:sequence minOccurs="2" maxOccurs="unbounded">
                    <xsd:element name="Bod">
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="Hodnota" type="xsd:double">
                          </xsd:element>
                          <xsd:element name="Popisek" type="xsd:string"/>
                          <xsd:element name="Cas" type="xsd:dateTime"/>
                        </xsd:sequence>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Příloha C – XSD schéma pro výstupní trendy 2. varianta

```
<?xml version="1.0" encoding="windows-1250"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://xml.netbeans.org/schema/XMLOutputPrubeznyBod"
  xmlns:tns="http://xml.netbeans.org/schema/XMLOutputPrubeznyBod"
  elementFormDefault="qualified">
  <xsd:element name="JGTrend-vystup-prubezny">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Trend" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="OsaMin" type="xsd:int"/>
              <xsd:element name="OsaMax" type="xsd:int"/>
              <xsd:element name="Jednotka" type="xsd:string"/>
              <xsd:element name="Název" type="xsd:string"/>
              <xsd:element name="Sestava" type="xsd:int"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="Bod">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Hodnota" type="xsd:double"/>
              <xsd:element name="Popisek" type="xsd:string"/>
              <xsd:element name="Cas" type="xsd:dateTime"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="sestava" type="xsd:int" use="required"/>
      <xsd:attribute name="název" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Příloha D – Popis instalace, nastavení vývojového prostředí

Vývojové prostředí

NetBeans IDE 6.9.1 s JDK 6.

V menu Tools -> Libraries přidat do Class Libraries knihovnu JCommon – jcommon-1.0.16 a knihovnu JFreeChart – jfreechart-1.0.13.

Pro práci s XML, XSD a XSLT v menu Tools -> Plugins nainstalovat plugin XML Schema and WSDL (verze 1.3 pro NetBeans IDE 6.9.1, Build 201011082200).

Aplikace

Pro spuštění aplikace je potřeba JRE 6.

Na přiloženém CD jsou umístěny jak instalační soubory pro vývojové prostředí a knihovny, tak zdrojové kódy aplikace, javadoc nápověda a XML/XSD/XSLT dokumenty.