
МАГИСТЪРСКИ ФАКУЛТЕТ
ДЕПАРТАМЕНТ „ИНФОРМАТИКА”
ПРОГРАМА „СОФТУЕРНИ ТЕХНОЛОГИИ В ИНТЕРНЕТ”

# МАГИСТЪРСКА ТЕЗА

НА
РАЛИЦА НИКОЛАЕВА КОЛЕВА
Фак. No: F23591

ТЕМА: РАЗРАБОТКА НА СИСТЕМА ЗА ГРАФИЧНО ИЗОБРАЗЯВАНЕ И ВИЗУАЛНО МАНИПУЛИРАНЕ НА ИЗКУСТВЕНИ НЕВРОННИ МРЕЖИ

НАУЧЕН РЪКОВОДИТЕЛ:.....
/ ИНЖ. ТОДОР БАЛАБАНОВ /

СОФИЯ
2012г.

# Съдържание:

# Увод

VitoshaTrade е софтуерен продукт на компанията ТБ Софт за прогнозиране на валутните пазари, предназначен за вграждане в информационно-търговската платформа MetaTrader. Прогнозите се извършват с помощта на генетични алгоритми и изкуствени невронни мрежи.

Към настоящия момент невронните мрежи биват представяни като матрица на съседство, които показват съществуващи или съответно несъществуващи връзки между отделните неврони. Подобно представяне затруднява манипулирането на мрежите изключително много поради факта, че потребителят не е в състояние да „начертае” в съзнанието си невронната мрежа по дадени матрици от единици и нули, особено при невронни мрежи с голям брой неврони.

С цел по-нататъшното усъвършенстване на софтуерния продукт VitoshaTrade, компанията възложи задача за проектиране и разработка на система за графично изобразяване и визуално манипулиране на изкуствени невронни мрежи, подпомагаща работата с мрежи от такъв тип.

Целта на настоящата магистърска теза е да се проектира и разработи информационна система за визуализиране и манипулиране на изкуствени невронни мрежи с потребителски дружелюбен интерфейс, с възможност за вграждане в уеб страница. За постигане на гореизложената цел са формулирани следните задачи:

1. Анализ на съществуващи системи за визуализация и манипулиране на изкуствени невронни мрежи.

2. Проучване на технологии, подходящи за разработване на информационна система от такъв тип.

3. Събиране и анализ на потребителските изисквания към бъдещата информационна система.

4. Изготвяне на диаграми и модели, подпомагащи разработката на системата.

5. Реализация на информационната система.

Дипломната работа се състои от увод, изложение, съдържащо три глави, заключение, използвана литература и четири приложения.

Първа глава съдържа обзор на предметната област и анализ на вече съществуващи системи, предназначени за визуално представяне и манипулиране на изкуствени невронни мрежи.

Във втора глава е представен анализ на алтернативни технологии за реализацията на проекта и обосновка на избора на технология.

Третата глава разкрива етапите на проектиране и реализация на информационната система. Проследени са събиране и анализ на потребителските изисквания към системата, изготвяне на Use case модел, привеждане в трета нормална форма и описание на базата от данни, програмна реализация на системата и анализ на качеството на програмния код.

Заключението съдържа обобщение на етапите по проектиране и разработка на приложението, използван софтуер и информация относно бъдещото развитие и планираните за системата промени.

В Приложение 1 са поместени предписанията към Use case модела, изготвен на базата на потребителските изисквания.

Приложение 2 съдържа основните успешни сценарии към Use case модела.

Приложение 3 представлява кратък наръчник на системния администратор, чиято отговорност е да инсталира приложението на сървър.

В Приложение 4 е представен програмният код на информационната системата, разработен изцяло от дипломанта.

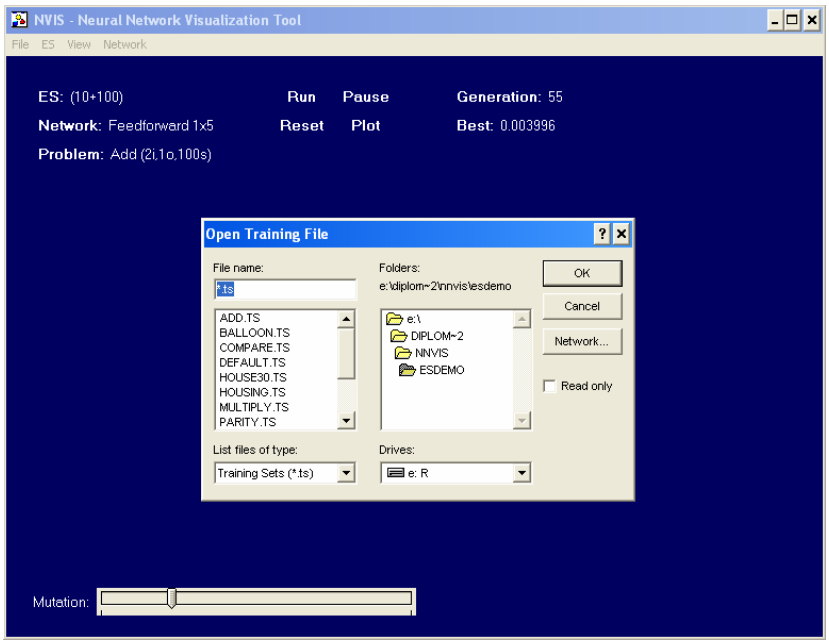
# ГЛАВА 1 Анализ на съществуващи системи за графично изобразяване и визуално манипулиране на изкуствени невронни мрежи

Съществуват различни системи, предлагащи възможност за изчертаване на изкуствени невронни мрежи и работа с тях. Някои от тях са научни разработки, други се предлагат на пазара като готов продукт.

## *1.1 N$^2$VIS: An Interactive Visualization Tool for Neural Networks*

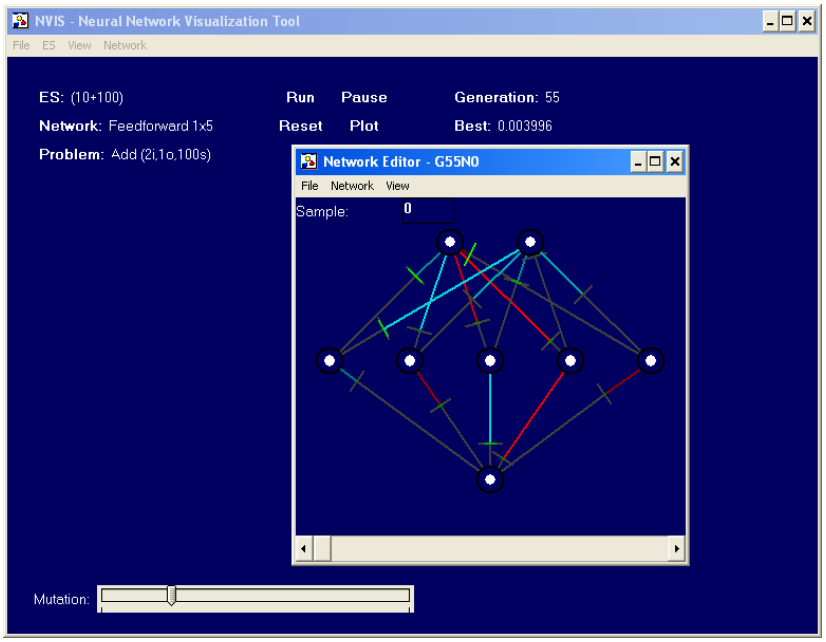
N$^2$VIS представлява интерактивен инструмент за визуализиране на изкуствени невронни мрежи. Посредством N$^2$VIS могат да бъдат изобразени топологията на дадена мрежа, теглата на връзките й, нивата на активиране на невроните. Към функционалностите на инструмента спада и възможността за интерактивно регулиране на тренировъчните параметри, както и изчертаването на екрана на резултатите от регулацията.

На Фиг.1.1 е представен основният екран на инструмента с прилежащия към него прозорец за отваряне на тренировъчен файл.




Фиг.1.1 Основен екран на инструментът за работа с невронни мрежи N$^2$VIS

Заредена изкуствена невронна мрежа е показана на Фиг.1.2.




Фиг.1.2 Изкуствена невронна мрежа, визуализирана в $N^2$VIS

Въпреки богатата функционалност, предлагана от $N^2$VIS, инструментът не е подходящ поради няколко причини, главните сред които са невъзможността за вграждане в уеб страница и използването му единствено под операционна система Microsoft Windows.

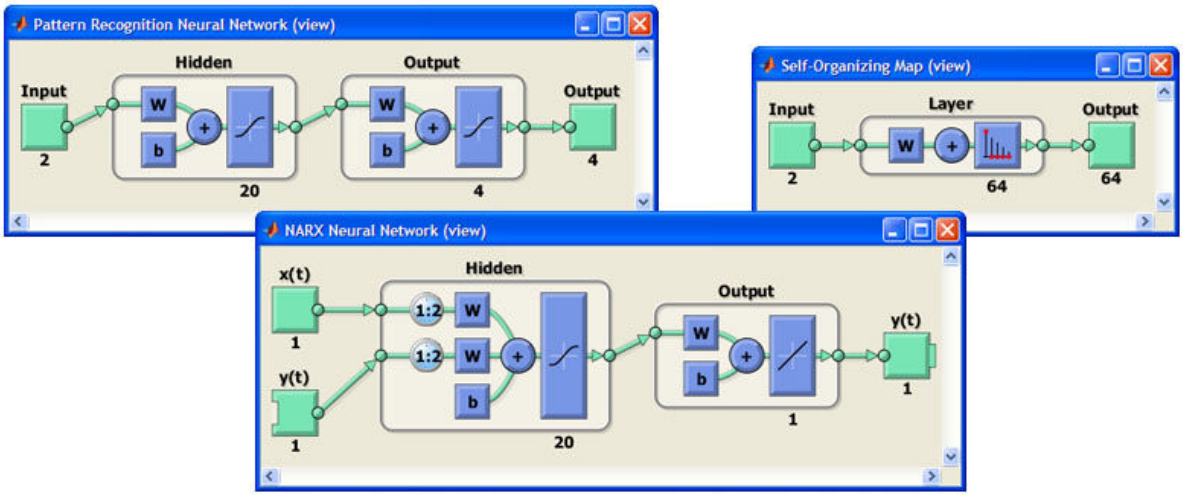
Използвани материали за $N^2$VIS са посочени в Използвана литература, източник [9].

## *1.2 Neural Network Toolbox™*

Софтуерът Neural Network Toolbox се предлага на пазара от международната корпорация MathWorks, Inc., специализирана в разработка на софтуер за математически изчисления. Последната версия на продукта е 7.0.2, излязла на 01 септември 2011г.

Neural Network Toolbox осигурява инструменти за проектиране, изпълнение, визуализиране и симулиране на невронни мрежи.

Фиг.1.3 представя три екрана с визуализирани изкуствени невронни мрежи.




Фиг.1.3 Екрани на Neural Network Toolbox, представящи изкуствени невронни мрежи

Основен недостатък на софтуера от гледна точка на потребителските изисквания е необходимостта от инсталиране на продукта на локална машина и невъзможността за вграждане в уеб страница.

Източници на информация за софтуерния продукт Neural Network Toolbox са посочени в Използвана литература, [14], [17].

Съществуват и други софтуери за изчертаване и манипулиране на изкуствени невронни мрежи, като NeuroEvolution Visualization Toolkit [18], представящ изкуствените невронни мрежи в браузър в SVG формат и разработеният на програмен език C# инструмент SharpNEAT [19]. Те, обаче, също не дават възможност за вграждане в уеб страница. Работят единствено на локален компютър, поради което не биха могли да удовлетворят нуждите на бъдещите потребители.

Системата за графично изобразяване и визуално манипулиране на изкуствени невронни мрежи, предложена като магистърска теза, е насочена към решението на проблема със затрудненията, с които потребителите се сблъскват по време на извършването на манипулации над невронните мрежи. Приложението е изцяло съобразено с изискванията на бъдещите му потребители.

# ГЛАВА 2 Избор на технология за реализация

Поради необходимостта информационната система за графично изобразяване и визуално манипулиране на изкуствени невронни мрежи да предоставя възможност за вграждането й в HTML страница, анализът на алтернативни технологии включва най-широко разпространените средства за разработка на интерактивно уеб съдържание. Разгледани са приложният софтуер Adobe® Flash®, алтернативата му Microsoft® Silverlight™ и JAVA™ аплетите.

## *2.1 Приложение на Adobe® Flash®*

Adobe Flash е платформа, предназначена за създаване на интерактивни мултимедийни приложения. Създадена е от компанията Macromedia през 1996 година и до 2005г. е позната под името Macromedia Flash. Софтуерът съчетава в себе си векторна и растерна графика, притежава богати възможности за възпроизвеждане на видео и аудио. Използва се за създаване на анимирани интернет страници, уеб игри и реклами, различни програми и презентации. Интерактивността на Flash приложенията се постига чрез използването на вградения скриптов език ActionScript.

Основни предимства на мултимедийните приложения, създадени с Adobe Flash, са гладката и приятна за окото анимация и възможността за създаване на сложни анимирани графики при по-ниско време за зареждане.

Въпреки предимствата на мултимедийната платформа Adobe Flash, тя не е подходяща за разработка на информационната система за графично визуализиране и манипулиране поради следните причини:

- Adobe Flash е продукт със затворен код;
- поддръжката му на различните операционни системи не е толкова добра, в сравнение с други технологии, в частност Java.

Източници на информация за Adobe Flash са посочени в Използвана литература, [11], [21].

## *2.2 Приложение на Microsoft® Silverlight™*


Silverlight представлява инструмент за разработване на интерактивни мултимедийни приложения. Създаден е от компанията Microsoft Corporation и е познат като алтернатива на платформата Adobe Flash поради сходните им характеристики и предназначение. Първата версия е разработена под кодово име Windows Presentation Foundation/Everywhere (WPF/E) през 2007г. Последната версия е Silverlight 5.

Приложенията, разработени на Silverlight, притежават следните предимства:

- съдържанието им е достъпно за търсещите машини;
- еднократно разработено приложение може да работи както в интернет браузър, така и като инсталирано на компютър приложение;
- притежават многоплатформеност и съвместимост с популярните браузъри;
- софтуерът за сървъра и клиента могат да бъдат разработени независимо един от друг.

За недостатъци на Silverlight се смятат:

- ниската популярност на технологията, особено в България;
- недостатъчно помощни материали и информация относно технологията и начина й на използване в сравнение с останалите по-стари и познати технологии.

Основен недостатък на Silverlight е, че продуктът е комерсиален.

Като недостатък може да се изтъкне и потенциалната възможност версия 5 да е последна версия на Silverlight, след която работата по платформата да бъде прекратена.

Информация за софтуера Silverlight е почерпана от източник [8], [16], [22] в Използвана литература.

## *2.3 JAVA™ аплети*


Java аплетът представлява компилирана програма на Java. Вгражда се като обект в HTML страница и се компилира от вградената в уеб браузъра Java виртуална машина (Java Virtual Machine). За първи път Java аплетите са представени през 1995г. с първата версия на езика Java.

Java аплетите притежават множество предимства пред останалите средства за разработка на интерактивно уеб съдържание, произтичащи главно от предимствата на самия език за програмиране Java.

- Java е добре познат език за програмиране с над 15 годишна история.
- Наличен е значителен обем информация относно технологията и начина й на използване. Съществуват огромен брой книги, помощни документи и Web страници, подпомагащи разработчиците.
- Разработката на приложения на Java не изисква допълнителни разходи, Java е с отворен код. Съществуват достатъчно добри безплатни и утвърдени инструменти за разработка на програми на Java.
- Вероятността Java да продължи да се използва във времето е голяма.
- Аплетите притежават платформена независимост, а поддръжката им е вградена във всеки наложил се в днешно време интернет браузър.
- Аплетите се интегрират лесно в уеб страници.
- Java аплетите предоставят мощни инструменти за постигане на интерактивност, като прихващане на събития.

За недостатъци на технологията се считат по-бавната скорост на машини с по-малка RAM памет и по-слаб процесор, както и по-семплия външен вид на аплетите, в сравнение с графичния потребителски интерфейс на Flash или Silverlight приложение. Тъй като системата за графично изобразяване и манипулиране на изкуствени невронни мрежи не цели високо бързодействие и красив външен вид, а стабилност, надеждност и сигурност, посочените недостатъци на Java аплетите не би следвало да се вземат предвид за конкретната разработка.

Поради предимствата, които Java аплетите притежават, тази технология е избрана за разработка на системата за визуално представяне и манипулиране на изкуствени невронни мрежи.

Източници на информация за Java аплети са посочени в Използвана литература, [2], [7], [13].

# ГЛАВА 3 Проектиране и реализация на информационната система

## *3.1 Събиране и анализ на потребителските изисквания*

При проведено интервю с бъдещи потребители на системата са отбелязани следните изисквания:

- интерактивна информационна система с възможност за вграждане в HTML страница;
- интуитивен и удобен за използване графичен потребителски интерфейс;
- защита на системата чрез удостоверение на потребителя;
- възможност за свързване към отдалечена MySQL база от данни;
- създаване на нови изкуствени невронни мрежи чрез въвеждане на валутна двойка, времеви период и брой неврони;
- зареждане на списък със съществуващите в системата невронни мрежи;
- изтриване на невронни мрежи от системата;
- възможност за търсене на конкретна мрежа посредством избор на валутна двойка и/или времеви период, върху който работи тя;
- визуализиране на топологията на избрана невронна мрежа на екрана с възможност за редакция на параметрите на невроните и връзките;
- промяна на координатите на невроните в мрежата чрез използване на техниката „издърпване и пускане” (“drag and drop”);
- запаметяване на промените в параметрите на невроните и връзките чрез обновяването им в базата от данни;
- задаване на цветове на различните типове неврони по избор на потребителя;
- избор на размер на радиуса на невроните в мрежата;
- възможност за визуализиране и скриване на номерата на невроните при изчертаване на мрежата на екрана;
- задаване големина на номерата на невроните по избор на потребителя;
- възможност за изчертаване на връзките между невроните в една мрежа в различни цветове, в зависимост от големината на теглата и/или активностите им;

- избор на фонов цвят на работното поле;
- надписи и съобщения на английски език и възможност за бърза и навременна смяна на езика в приложението при поискване от потребителя.

### *3.2 Изготвяне на Use case модел*

На базата на горепосочените потребителски изисквания е изготвен следният Use case модел.

### 3.2.1 Use case диаграма.


Фиг.3.1 Use case диаграма на системата за графично изобразяване и манипулиране на изкуствени невронни мрежи

### 3.2.2 Предписания

**Предписания за Use case “Създаване на нова изкуствена невронна мрежа”.**

1. Предпоставки.

Потребителят желае да добави нова изкуствена невронна мрежа в системата.

2. Предусловия.

Потребителят трябва да влезе в системата с коректно потребителско име и парола.

3. Стартиране на Use case.

Системата трябва да е в изправност. Не трябва да съществуват събития, които да възпрепятстват влизането на потребителя в нея (спиране на тока, прекъсната връзка с Интернет, хардуерен проблем с компютъра на потребителя).

4. Диалог.

- Потребителят въвежда коректни хост, порт, потребителско име и парола за връзка на системата към базата от данни.
- Системата инициализира начален екран за визуализиране на изкуствена невронна мрежа.
- Потребителят избира меню за въвеждане на нова изкуствена невронна мрежа.
- Системата визуализира екран, съдържащ падащи менюта за избор на валутна двойка и период, текстово поле за въвеждане на брой неврони в мрежата, както и бутони за зареждане на текущата информация в падащите менюта и запаметяване на нова изкуствена невронна мрежа.
- Потребителят натиска бутон за зареждане в падащите менюта на наличните в системата валутни двойки и времеви периоди.
- Системата обновява стойностите в падащите менюта.
- Потребителят избира валутна двойка и период, въвежда брой на невроните и натиска бутон за запаметяване на нова невронна мрежа.
- Системата запаметява информацията за новата изкуствена невронна мрежа и я визуализира на екрана.

5. Прекратяване на Use case.

Потребителят излиза от системата или избира друго меню.

6. След-условия.

Нововъведената изкуствена невронна мрежа е налична в системата и може да бъде манипулирана.

**Предписания за Use case “Изтриване на изкуствена невронна мрежа”.**

1. Предпоставки.

Потребителят желае да премахне изкуствена невронна мрежа от системата.

2. Предусловия.

Потребителят трябва да влезе в системата с коректно потребителско име и парола.

3. Стартиране на Use case.

Системата трябва да е в изправност. Не трябва да съществуват събития, които да възпрепятстват влизането на потребителя в нея (спиране на тока, прекъсната връзка с Интернет, хардуерен проблем с компютъра на потребителя).

4. Диалог.

- Потребителят въвежда коректни хост, порт, потребителско име и парола за връзка на системата към базата от данни.
- Системата инициализира начален екран за визуализиране на изкуствена невронна мрежа.
- Потребителят избира меню за изтриване на изкуствена невронна мрежа.
- Системата визуализира екран, съдържащ падащо меню за избор на идентификационен номер на невронна мрежа, както и бутони за зареждане на наличните идентификационни номера в падащото менюта и изтриване на мрежа.
- Потребителят натиска бутон за зареждане в падащото меню на наличните в системата идентификационни номера на невронни мрежи.
- Системата обновява стойностите в падащото меню.
- Потребителят избира идентификационен номер на мрежа и натиска бутон за изтриване.
- Системата премахва информацията за избраната невронна мрежа и се връща на екрана за изтриване на мрежи.

5. Прекратяване на Use case.

Потребителят излиза от системата или избира друго меню.

6. След-условия.

Избраната за премахване изкуствена невронна мрежа е изтрита от системата.

Останалата част от предписанията е разгледана в Приложение 1.

### 3.2.3 Основни успешни сценарии

**Сценарий на Use case “Създаване на нова изкуствена невронна мрежа”.**

1. Use case започва, когато потребителят влезе в прозореца за вход в системата.
2. Системата подканва потребителя да въведе хост, порт, потребителско име и парола.
3. Потребителят въвежда нужната информация. Ако хостът, портът, потребителското име и/или паролата са некоректни – стартира се поток за грешка E1.
4. Системата инициализира екран за визуализиране на изкуствени невронни мрежи.
5. Потребителят избира меню за създаване на нова изкуствена невронна мрежа. Ако потребителят избере друго меню, стартира се алтернативен поток A1.
6. Системата инициализира екран, съдържащ падащи менюта за избор на валутна двойка и период, текстово поле за въвеждане на брой неврони в мрежата, както и бутони за зареждане на текущата информация в падащите менюта и запаметяване на нова изкуствена невронна мрежа.
7. Потребителят натиска бутон за зареждане в падащите менюта на наличните в системата валутни двойки и времеви периоди.
8. Системата обновява стойностите в падащите менюта.
9. Потребителят избира валутна двойка и период, въвежда брой на невроните и натиска бутон за запаметяване на нова невронна мрежа. Ако потребителят не избере валутна двойка и/или времеви период и/или не въведе брой на невроните в новата невронна мрежа, стартира се поток на грешка E2.
10. Системата запаметява информацията за новата изкуствена невронна мрежа и я визуализира на екрана.
11. Use case приключва.

Алтернативен поток A1.

1. Инициализира се избраният от потребителя екран.

2. Use case приключва.

Поток на грешка E1.

1. Системата уведомява потребителя, че въведените идентификационни данни са невалидни.

2. Use case приключва.

Поток на грешка E2.

1. Системата уведомява потребителя, че необходимата информация не е въведена.

2. Use case приключва.

**Сценарий на Use case “Изтриване на изкуствена невронна мрежа”.**

1. Use case започва, когато потребителят влезе в прозореца за вход в системата.

2. Системата подканва потребителя да въведе хост, порт, потребителско име и парола.

3. Потребителят въвежда необходимата информация. Ако хостът, портът, потребителското име и/или паролата са некоректни – стартира се Поток на грешка E1.

4. Системата инициализира екран за визуализиране на изкуствени невронни мрежи.

5. Потребителят избира меню за изтриване на съществуваща в системата изкуствена невронна мрежа. Ако потребителят избере друго меню, стартира се алтернативен поток A1.

6. Системата визуализира екран, съдържащ падащо меню за избор на идентификационен номер на невронна мрежа, както и бутони за зареждане на наличните идентификационни номера в падащото менюта и изтриване на мрежа.

7. Потребителят натиска бутон за зареждане в падащото меню на наличните в системата идентификационни номера на невронни мрежи.

8. Системата обновява стойностите в падащото меню.

9. Потребителят избира идентификационен номер на мрежа и натиска бутон за изтриване. Ако не е избран идентификационен номер, стартира се поток на грешка E2.

10. Системата изтрива информацията за избраната изкуствена невронна мрежа и се връща на екрана за изтриване.

11. Ако потребителят желае да премахне друга мрежа от системата, се преминава към т. 9.

12. Use case приключва.

Алтернативен поток А1.

1. Инициализира се избраният от потребителя екран.

2. Use case приключва.

Поток на грешка Е1.

1. Системата уведомява потребителя, че въведените идентификационни данни са невалидни.

2. Use case приключва.

Поток на грешка Е2.

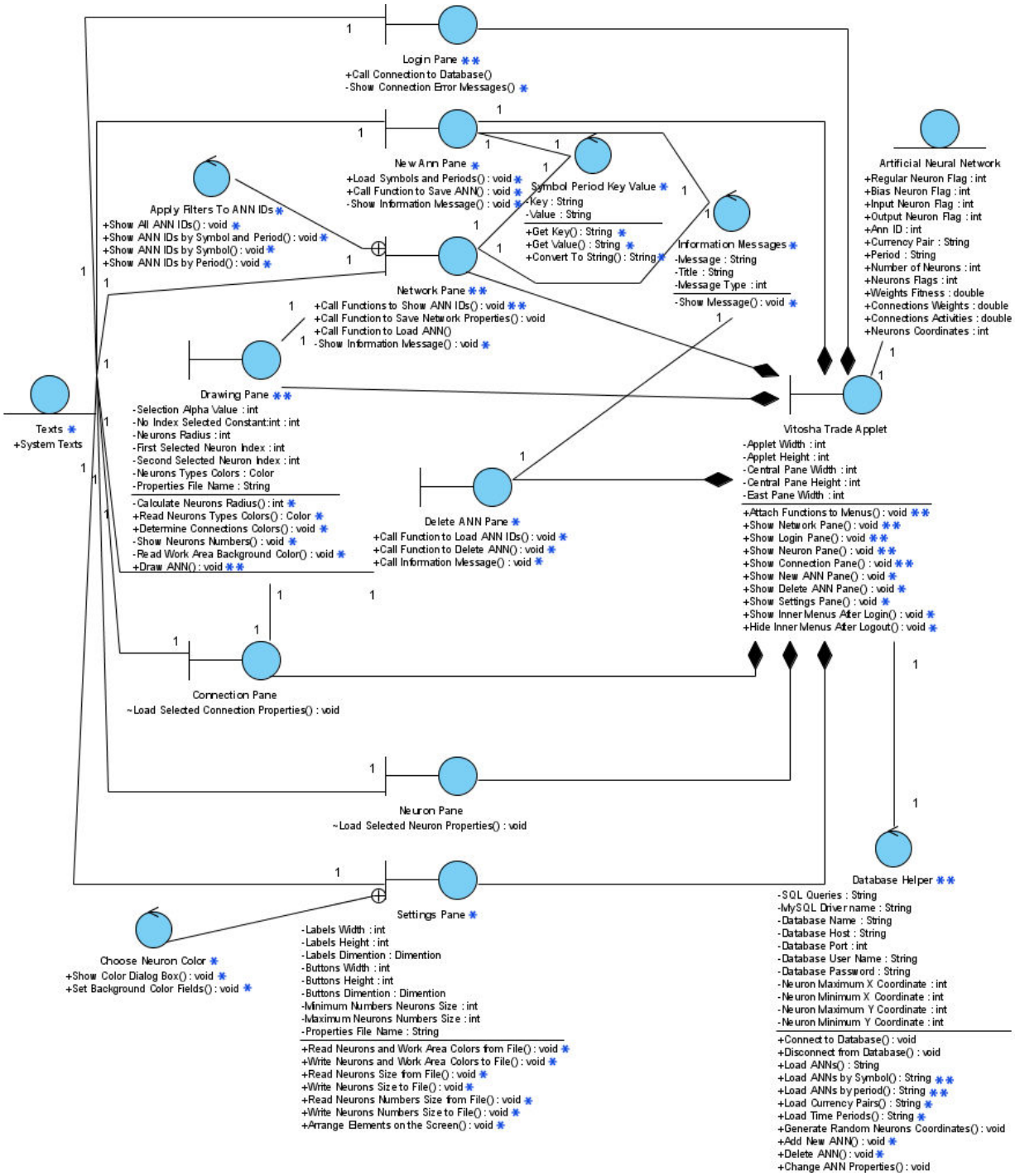
1. Системата уведомява потребителя, че е необходимо да избере идентификационен номер на невронна мрежа.

2. Use case приключва.

Останалата част основни успешни сценарии е разгледана в Приложение 2.

### *3.3 Изготвяне на Class диаграма*

За подпомагане разработката на системата за визуализиране и манипулиране на изкуствени невронни мрежи е изготвена следната Class диаграма (виж Фиг.3.2).




Фиг.3.2 Class диаграма на системата за графично изобразяване и манипулиране на изкуствени невронни мрежи

* Класовете и методите, обозначени с една синя звезда, са изцяло разработени от дипломанта.

** Класовете и методите, обозначени с две сини звезди, са частично разработени от дипломанта.

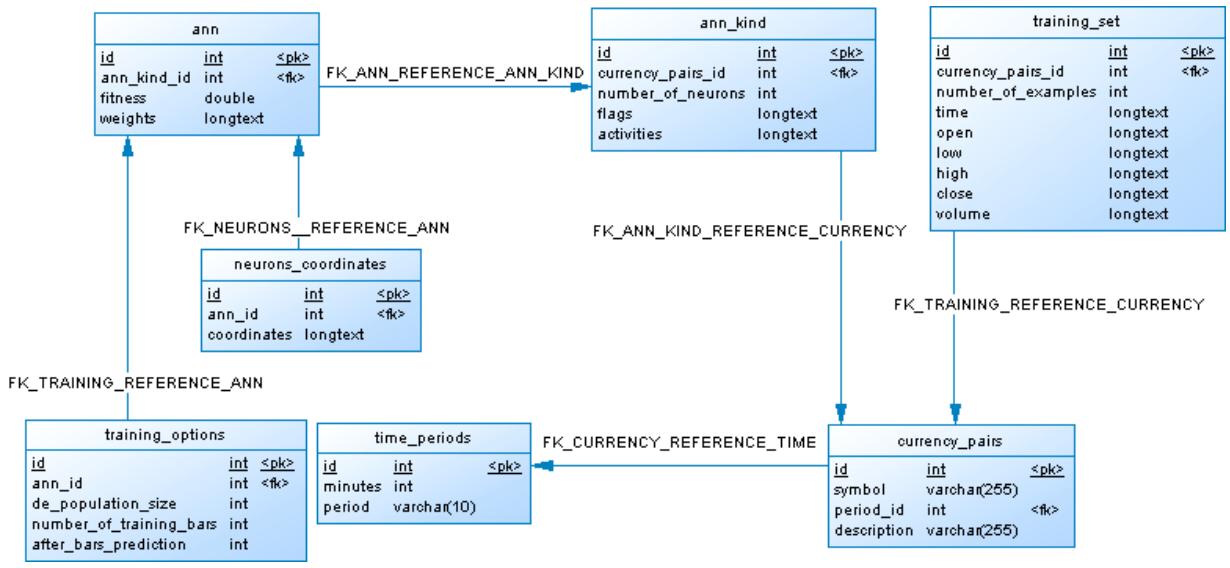
### *3.4 База от данни*

Поради лошата структура на съществуващата база от данни се наложи привеждането ѝ в трета нормална форма.

## 3.4.1 Физически и концептуален модел на базата от данни

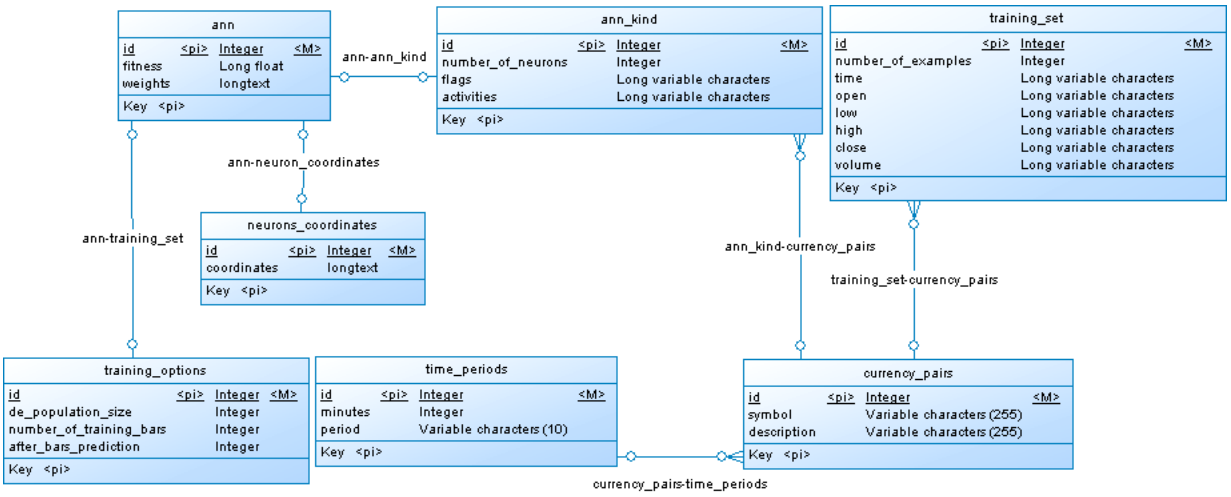
Използваната база от данни е релационна. Тя е индексирана по главните ключове на всяка една от таблиците. Поради неголемия брой таблици и неголемия размер на базата от данни, не са предвидени вторични индекси за подобряване производителността на системата.

Физическият модел на базата от данни е представен на Фиг.3.3.




Фиг.3.3 Физически модел на базата от данни

Концептуалният модел на базата от данни е представен на Фиг.3.4.




Фиг.3.4 Концептуален модел на базата от данни

## 3.4.2 Описание на релациите в базата от данни

За реализиране на функционалността на системата са използвани пет от седемте релации в базата от данни, както следва:

- ann
- ann_kind
- neurons_coordinates
- currency_pairs
- time_periods

Релация **ann** съдържа информация за невронни мрежи. Към отношението се добавят, променят и изтриват кортежи съответно от екраните за добавяне, промяна на параметри и изтриване на изкуствени невронни мрежи.

Релацията се състои от 4 атрибута, както следва (виж Таблица 3.1):

Таблица 3.1

| **ann** | | | | |
|---|---|---|---|---|
| ИМЕ НА АТРИБУТ | ТИП | ГЛАВЕН КЛЮЧ | ЧУЖД КЛЮЧ | ИНФОРМАЦИЯ |
| id | int | V | Чужд ключ за релация neurons_coordinates. | Уникален идентификационен номер на невронна мрежа. Допълнително: auto_increment. |
| ann_kind_id | int | | Чужд ключ от релация ann_kind. | Уникален идентификационен номер на типа на невронната мрежа |
| fitness | double | | | Реално число, показващо ефективността на невронната мрежа. |
| weights | longtext | | | Символен низ, съдържащ теглата на връзките на невронната мрежа, разделени с интервал. |

Релация **ann_kind** е предназначена за съхранение на типовете невронни мрежи. Към нея се добавят, променят и изтриват кортежи съответно при добавяне, промяна на параметри и изтриване на невронни мрежи.

Степента на отношението е 5 (виж Таблица 3.2).

Таблица 3.2

| **ann_kind** | | | | |
|---|---|---|---|---|
| ИМЕ НА АТРИБУТ | ТИП | ГЛАВЕН КЛЮЧ | ЧУЖД КЛЮЧ | ИНФОРМАЦИЯ |
| id | int | V | Чужд ключ за релация ann. | Уникален идентификационен номер на тип на невронна мрежа. Допълнително: auto_increment. |
| currency_pairs_id | int | | Чужд ключ от релация currency_pairs. | Уникален идентификационен номер на валутна двойка за съответния тип невронна мрежа. |
| number_of_neurons | int | | | Цяло число, показващо броя неврони в невронната мрежа. |
| flags | longtext | | | Символен низ, съдържащ типовете неврони в целочислен формат, разделени |

| | | | | |
|---|---|---|---|---|
| | | | | с интервал. |
| activities | longtext | | | Символен низ, съдържащ активностите на неврони в целочислен формат, разделени с интервал. |

Отношението **neurons_coordinates** съдържа координатите на невроните в дадена изкуствена невронна мрежа. Към релацията се прибавят и изтриват кортежи съответно при добавяне и премахване на изкуствени невронни мрежи.

Отношението притежава 3 на брой атрибута (виж Таблица 3.3).

Таблица 3.3

| **neurons_coordinates** | | | | |
|---|---|---|---|---|
| ИМЕ НА АТРИБУТ | ТИП | ГЛАВЕН КЛЮЧ | ЧУЖД КЛЮЧ | ИНФОРМАЦИЯ |
| id | int | V | | Уникален идентификационен номер на запис с координати на неврони. Допълнително: auto_increment. |
| ann_id | int | | Чужд ключ от релация ann. | Уникален идентификационен номер на невронна мрежа. |
| coordinates | longtext | | | Символен низ, съдържащ координатите на невроните в целочислен формат, разделени с интервал. |

Релацията **currency_pairs** съдържа валутните двойки, върху които работят изкуствените невронните мрежи. Информацията в отношнието е зададена предварително и не е възможна промяна през системата за визуализиране и манипулиране на изкуствени невронни мрежи.

Релацията е от степен 4 (виж Таблица 3.4).

Таблица 3.4

| currency_pairs | | | | |
|---|---|---|---|---|
| ИМЕ НА АТРИБУТ | ТИП | ГЛАВЕН КЛЮЧ | ЧУЖД КЛЮЧ | ИНФОРМАЦИЯ |
| id | int | V | Чужд ключ за релация ann_kind. | Уникален идентификационен номер на валутна двойка. Допълнително: auto_increment. |
| symbol | varchar (255) | | | Символен низ с дължина 255 символа, съдържащ име на валутна двойка. |
| period_id | longtext | | Чужд ключ от релация time_periods. | Уникален идентификационен номер на времеви период.. |
| description | varchar (255) | | | Кратко описание на съответната валутна двойка. |

Отношението **time_periods** е предназначено за съхранение на информация за времевите периоди, върху които работят невронните мрежи. Информацията в релацията е заложена предварително и не е възможна промяна през системата.

Релацията се състои от 3 атрибута, както следва (виж Таблица 3.5.):

Таблица 3.5

| time_periods | | | | |
|---|---|---|---|---|
| ИМЕ НА АТРИБУТ | ТИП | ГЛАВЕН КЛЮЧ | ЧУЖД КЛЮЧ | ИНФОРМАЦИЯ |
| id | int | V | Чужд ключ за релация currency_pairs | Уникален идентификационен номер на времеви период. Допълнително: auto_increment. |
| minutes | int | | | Времеви период в минути, върху който работят изкуствените невронни мрежи. |
| period | varchar (10) | | | Символен низ с дължина 10 символа, съдържащ името на времевия период. |

## *3.5 Програмна реализация на системата за графично изобразяване и визуално манипулиране на изкуствени невронни мрежи*

Системата е реализирана под формата на Java аплет, като за изграждане на потребителския интерфейс е използван стандартният набор от инструменти в графичната библиотека „Swing”.

Разработеният аплет включва няколко панела, които се сменят в процеса на работа със системата, работна среда за изчертаване и манипулиране на изкуствени невронни мрежи и лента с менюта за навигация между отделните екрани на приложението и задаване на допълнителни настройки.

Системата съдържа следните 16 класа, като 2 от тях представляват вътрешни класове. Последващите описания включват методи и атрибути, създадени изцяло от дипломанта.

**VitoshaTradeApplet**

Класът наследява `JApplet`. В него са интегрирани всички екрани под формата на класове-наследници на `JPanel`.

- `public NewAnnPane newAnnPane = new NewAnnPane(this)` създава инстанция на класа `NewAnnPane`, която дава възможност на потребителя да въведе нова невронна мрежа в системата.
- `public DeleteAnnPane deleteAnnPane = new DeleteAnnPane(this)` създава обект от клас `DeleteAnnPane`, чрез който потребителят може да премахне невронна мрежа от системата.
- `public SettingsPane settingsPane = new SettingsPane(this)` създава инстанция на класа `SettingsPane`, която дава възможност на потребителя да промени настройките на системата.

В класа е включена и лента с менюта, съдържаща следните елементи:

- `private JMenu networkMenu = new JMenu(Texts.MENU_NETWORK)` съдържа подменюта за визуализиране на екрани за създаване на нова изкуствена невронна мрежа, изчертаване на избрана невронна мрежа, изтриване на невронна мрежа.

- `private JMenuItem newNetworkItem = new JMenuItem(Texts.MENU_ITEM_NEW)` визуализира екран за въвеждане на информация за нова изкуствена невронна мрежа в системата. При натискането на този елемент от менюто се извиква метод `showNewAnnPane()`, който премахва текущите панели и зарежда екрана за създаване на нова изкуствена невронна мрежа.
- `private JMenuItem loadNetworkItem = new JMenuItem(Texts.MENU_ITEM_LOAD)` визуализира екран за избор на невронна мрежа за изчертаване. При натискане на този елемент от менюто се извиква функцията `showNetworkPane()`, която премахва текущите панели и зарежда панелите съответно за избор на невронна мрежа и работна среда за изчертаване и манипулиране на избраната такава.
- `private JMenuItem deleteNetworkItem = new JMenuItem(Texts.MENU_ITEM_DELETE)` визуализира екран за изтриване на изкуствена невронна мрежа от системата. При натискане на този елемент от менюто се извиква методът `showDeleteAnnPane()`, който премахва текущите панели и визуализира панел за изтриване на невронна мрежа.
- `private JMenu toolsMenu = new JMenu(Texts.MENU_TOOLS)` съдържа подменюта за допълнителни настройки при визуализиране на изкуствена невронна мрежа, както и на работната среда.
- `private JMenuItem settingsItem = new JMenuItem(Texts.MENU_ITEM_SETTINGS)` отваря екран с настройки към изкуствените невронни мрежи. При натискане на този елемент от менюто се извиква функцията `showSettingsPane()`, премахваща текущите панели и визуализираща панел с наличните настройки.
- `private JMenu meshMenu = new JMenu(Texts.MENU_MESH)` представлява подменю на `toolsMenu`. То дава възможност за избор на начин на изчертаване на връзките между невроните в дадена изкуствена невронна мрежа. Към него са включени четири подменюта с възможност за единствен избор чрез радио бутон. За целта е използвана група от бутони, включваща четирите подменюта – `private ButtonGroup meshItemsGroup = new ButtonGroup()`.
- `public JRadioButtonMenuItem meshActivitiesItem = new JRadioButtonMenuItem(Texts.MENU_ITEM_ACTIVITIES)` е първото подменю на `meshMenu`, като при изборът му изчертана невронна мрежа се изчертава наново, следвайки конкретни правила за визуализиране на връзките между невроните според активността на всяка една връзка.

- `public JRadioButtonMenuItem meshWeightsItem = new JRadioButtonMenuItem(Texts.MENU_ITEM_WEIGHTS)` е второто подменю на `meshMenu`, като при изборът му изчертана невронна мрежа се изчертава наново, следвайки конкретни правила за визуализиране на връзките между невроните според теглото на всяка една връзка.
- `public JRadioButtonMenuItem meshBothItem = new JRadioButtonMenuItem(Texts.MENU_ITEM_BOTH)` е третото подменю на `meshMenu`, като при изборът му изчертана невронна мрежа се изчертава наново, следвайки конкретни правила за визуализиране на връзките между невроните според активността и теглото на всяка една връзка, взети заедно.
- `public JRadioButtonMenuItem meshSolidItem = new JRadioButtonMenuItem(Texts.MENU_ITEM_SOLID)` е четвъртото подменю на `meshMenu`, като при изборът му изчертана невронна мрежа се изчертава наново. Връзките между невроните се визуализират в плътен цвят, независимо от активностите и теглата им.
- `public JCheckBoxMenuItem numberingItem = new JCheckBoxMenuItem(Texts.MENU_ITEM_NUMBERING)` представлява подменю на `toolsMenu`. То бива избрано или не чрез квадрат за отметка. При запълване на квадрата за отметка, към всеки неврон от изчертана невронна мрежа на екрана се визуализира и неговият уникален идентификационен номер.

В класа `VitoshaTradeApplet` са включени следните методи.

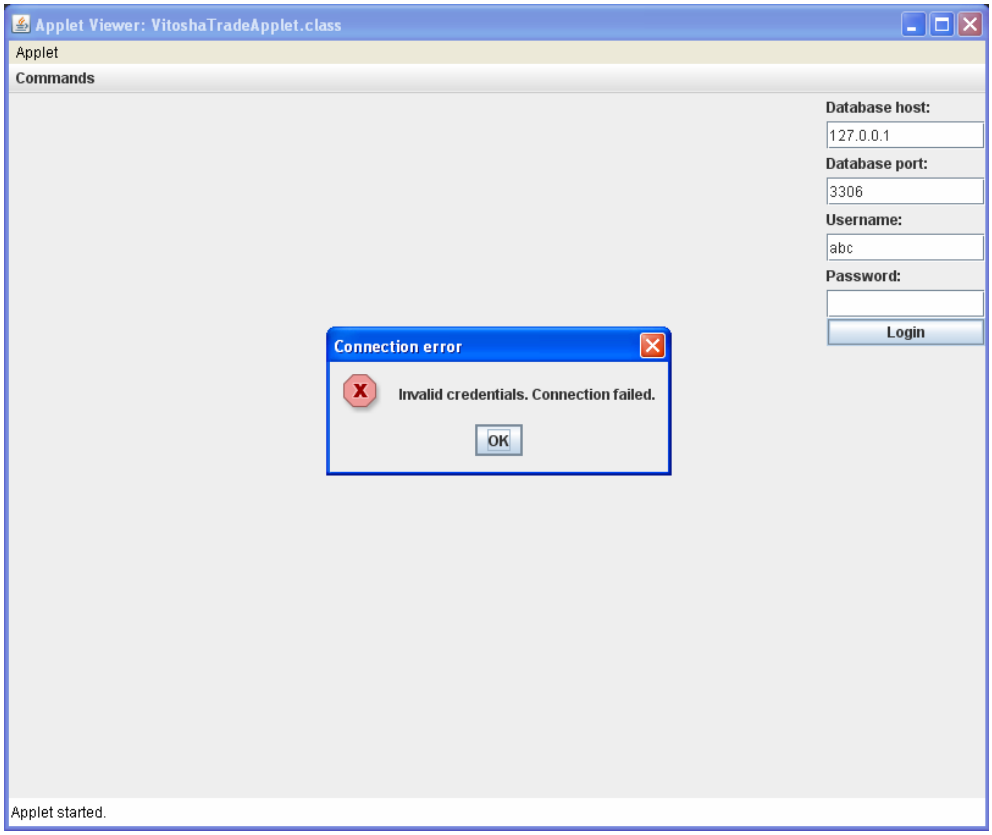
- `public void showNewAnnPane()` визуализира на екрана панел за въвеждане на информация за нова изкуствена невронна мрежа, след като премахне всички останали панели от аплета.
- `public void showDeleteAnnPane()` показва на екрана панел за изтриване на изкуствена невронна мрежа, като преди това премахва от аплета всички други панели.
- `public void showSettingsPane()` добавя в аплета панел за промяна на настройките на системата, след премахване на останалите налични панели.
- `public void showMenuBarAfterLogin()` се изпълнява при вход в системата. В метода се осъществява прибавяне към лентата с менюта на менютата, които могат да бъдат използвани от потребителя само и единствено след идентификацията му от страна на системата. Радио бутонът на елемента `meshSolidItem`, принадлежащ към менюто `meshMenu`, бива автоматично избран по подразбиране.

- `public void hideMenuBarAfterLogout()` се изпълнява при изход от системата. Извършва се премахване от лентата с менюта на менютата, предназначени за работа със системата единствено след идентификация на потребител.

**LoginPane**

Клас `LoginPane` наследява `JPanel`. Той визуализира екран за вход в системата. Съдържа следния метод, който се използва при некоректни идентификационни данни, въведени от страна на потребителя:

- `private void showConnectionError()`, в който се създава обект от клас `InformationMessages` и съответно се показва съобщение за грешка. Екранът за вход и прилежащото към него съобщение за грешка в случай на въведени неправилни идентификационни данни е показан на Фиг.3.5.




Фиг.3.5 Екран за вход и прилежащото към него съобщение за грешка

**NewAnnPane**

Класът е наследник на `JPanel`. Чрез него потребителят има възможност да въведе нова изкуствена невронна мрежа в системата. За позициониране на елементите на екрана е използван GridLayout с двадесет и пет реда и една колона. В класа са включени десет компонента, както следва:

- падащо меню `private JComboBox networkSymbol = new JComboBox()`, в което се зареждат наличните символи на валутни двойки (например EURUSD за валутна двойка Евро-Долар);
- падащо меню `private JComboBox networkPeriod = new JComboBox()`, показващо наличните времеви периоди, върху които работят изкуствените невронни мрежи (например M1 за едноминутен времеви период, M5 за петминутен времеви период и т.н.);
- текстово поле private `JTextField networkNumberNeurons = new JTextField()`, служещо за въвеждане от страна на потребителя на брой неврони в мрежата;
- бутон `private JButton save = new JButton()` за запаметяване на новата невронна мрежа;
- бутон `private JButton refresh = new JButton()` за обновяване на информацията в падащите менюта;
- три етикета JLabel, които се добавят при инициализирането на падащите менюта и текстовото поле в конструктора и пояснява данните към тях;
- два етикета без текст, служещи за създаване на разстояние между компонентите;

Клас `NewAnnPane` съдържа три анонимни класа:

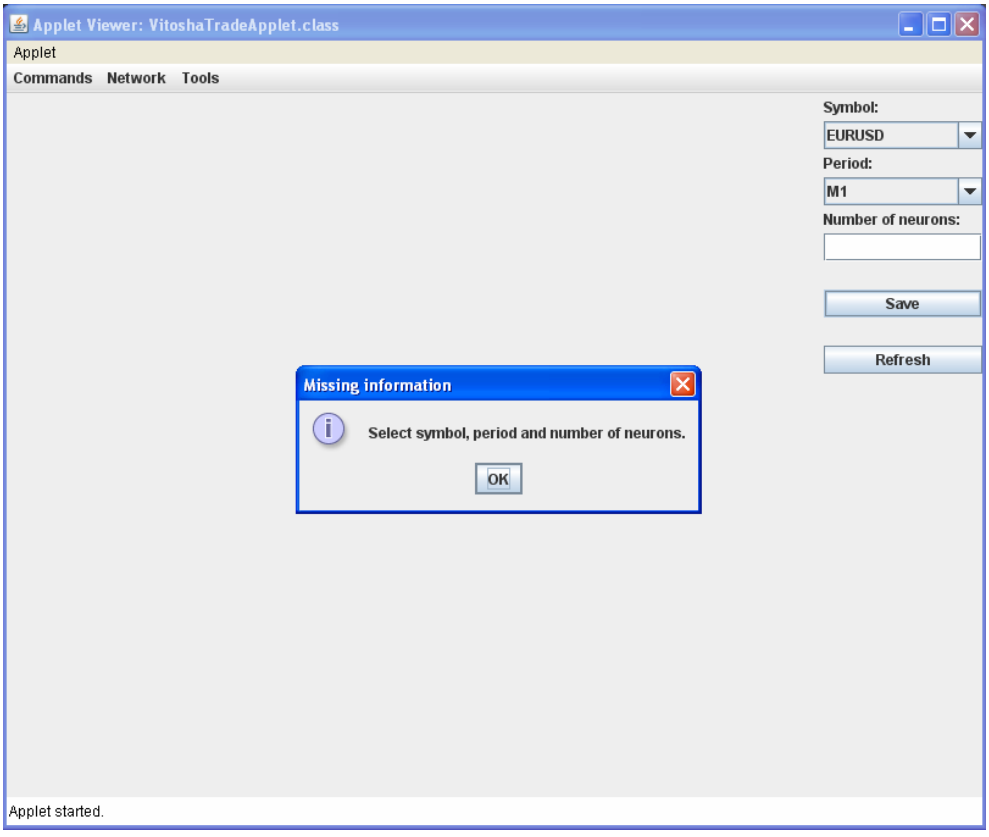
- клас `new KeyAdapter()`, съдържащ метод `public void keyTyped(KeyEvent event)`, чиято роля е да не позволи на потребителя да въведе в полето за брой на неврони символ, различен от цифра;
- клас `new ActionListener()`, прикачен към бутона за запаметяване, съдържащ метод `public void actionPerformed(ActionEvent event)`, който взима въведената от потребителя информация и с помощта на инстанция от класа `DatabaseHelper` запаметява информацията за новата невронна мрежа в базата от данни;

- клас `new ActionListener()`, прикачен към бутона за обновяване на информацията в падащите менюта, съдържащ метод `public void actionPerformed(ActionEvent event)`, който с помощта на инстанция от класа `DatabaseHelper` опреснява информацията.

Класът съдържа и следния метод, който се използва при некоректни данни, избрани към новата изкуствена невронна мрежа:

- `private void showInformationMessage()`, в който се създава инстанция на клас `InformationMessages` и съответно се показва съобщение за грешка.

На Фиг.3.6 е представен екранът за въвеждане на изкуствена невронна мрежа и прилежащото към него съобщение за грешка при некоректно избрана информация. В конкретния случай потребителят не е въвел брой на невроните в мрежата.




Фиг.3.6 Екран за въвеждане на изкуствена невронна мрежа и прилежащо съобщение за грешка

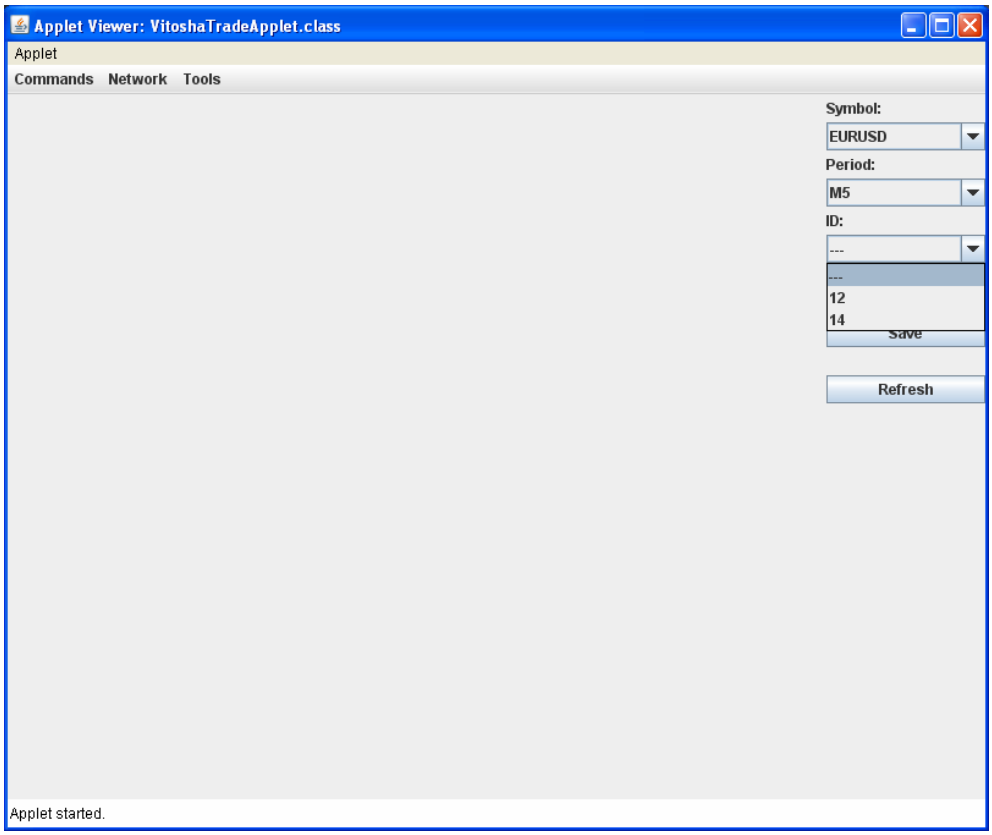
**NetworkPane**

Класът е наследник на `JPanel`. Чрез него потребителят има възможност да избере уникален идентификационен номер на изкуствена невронна мрежа и да я зареди в работното поле за извършване на понататъшни манипулации.

Към вече съществуващия клас е добавен един вътрешен клас:

- `private class SetFiltersBySimbolAndPeriod implements ItemListener`, съдържащ метод `public void itemStateChanged(ItemEvent event)`. Класът служи за прилагане на филтър и съответно търсене на идентификационен номер на изкуствена невронна мрежа според избрана валутна двойка и / или период. Така, при избор на валутна двойка и / или времеви период от съответните падащи менюта, в падащото меню за идентификационни номера на невронни мрежи динамично се зареждат само номера на мрежи, притежаващи избраните параметри.

Фиг.3.7 представя екрана за избор на идентификационен номер на невронна мрежа с приложен филтър към символ на валутна двойка и времеви период.

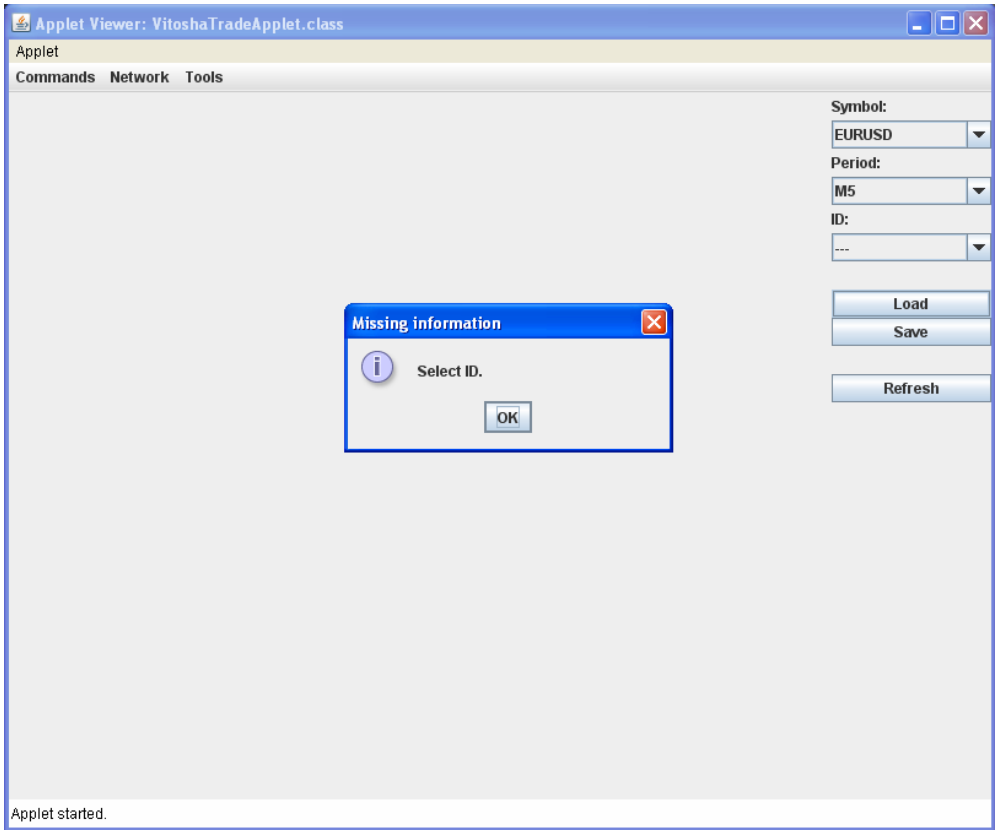



Фиг.3.7 Екран за избор на номер на невронна мрежа с избрани полета за филтър

Към вече съществуващия клас е добавен един метод, а именно:

- private void showInformationMessage(), в който се създава инстанция на клас `InformationMessages` и съответно се показва съобщение за грешка.

Екранът за избор на изкуствена невронна мрежа с прилежащо към него съобщение за грешка е показан на Фиг.3.8. В конкретния случай е натиснат бутон за изчертаване на мрежа, без да е избран идентификационен номер от падащото меню.




Фиг.3.8 Екран за избор на идентификационен номер на невронна мрежа и прилежащо съобщение за грешка

**DeleteAnnPane**

Класът е наследник на `JPanel`. Той дава възможност на потребителя да избира и изтрива изкуствени невронни мрежи от системата. За позициониране на елементите на екрана е използван `GridLayout` с двадесет и пет реда и една колона. В класа са включени шест компонента, както следва:

- падащо меню `private JComboBox networkId = new JComboBox()`, в което се зареждат уникалните идентификационни номера на наличните в системата изкуствени невронни мрежи;
- бутон `private JButton delete = new JButton()` за изтриване на избрана невронна мрежа;
- бутон `private JButton refresh = new JButton()` за обновяване на информацията в падащите менюта;
- един етикет `JLabel`, който се добавя при инициализирането на падащото меню в конструктора и пояснява данните към него;
- два етикета без текст, служещи за създаване на разстояние между компонентите;

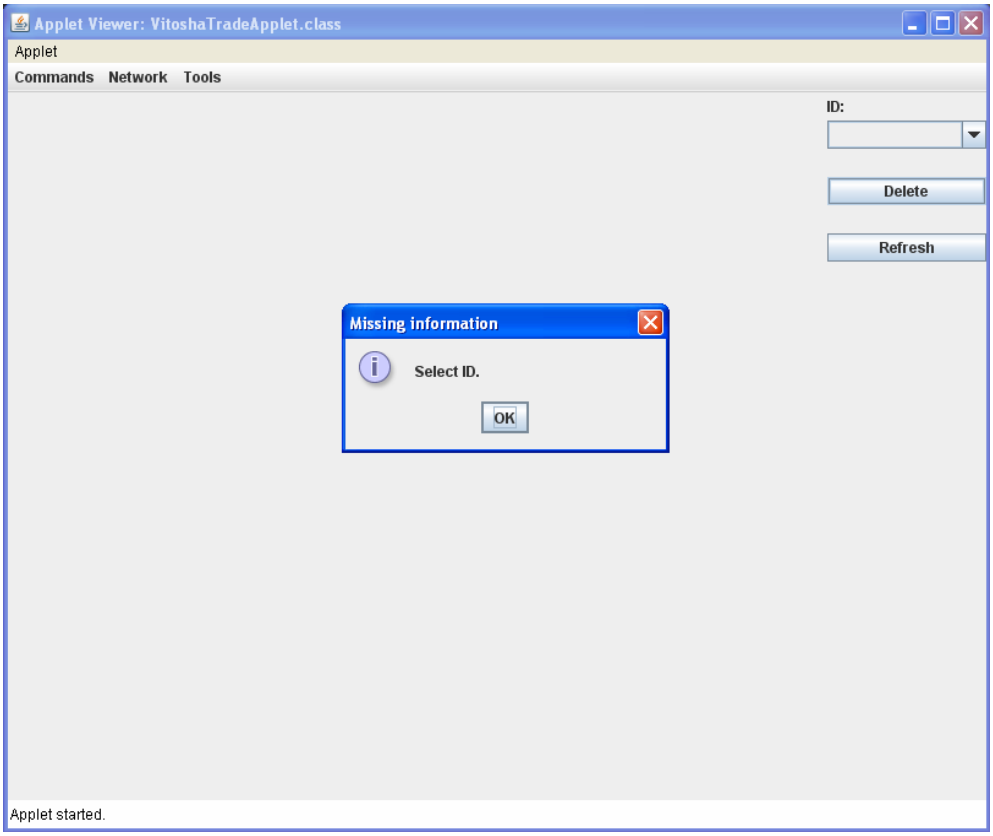
Клас `NewAnnPane` съдържа два анонимни класа:

- клас `new ActionListener()`, прикачен към бутона за премахване на невронна мрежа от системата, съдържащ метод `public void actionPerformed(ActionEvent event)`, който взима избрания от потребителя уникален идентификационен номер на мрежа и с помощта на инстанция от класа `DatabaseHelper` изтрива информацията за избраната мрежа от базата данни;
- клас `new ActionListener()`, прикачен към бутона за обновяване на информацията в падащото меню, съдържащ метод `public void actionPerformed(ActionEvent event)`, който с помощта на метода `loadAllAnnIds()` опреснява информацията.

Класът съдържа и следните два метода:

- `public void loadAllAnnIds()`, зареждащ информацията в падащото меню за идентификационни номера на невронни мрежи;
- `private void showInformationMessage()`, в който се създава инстанция на клас `InformationMessages` и съответно се показва съобщение за грешка. Възможна грешка е натискането на бутона за изтриване, без да е избрана стойност от падащото меню.

На Фиг.3.9 е представен екранът за премахване на изкуствена невронна мрежа и прилежащото към него съобщение за грешка.




Фиг.3.9 Екран за премахване на изкуствена невронна мрежа и прилежащо съобщение за грешка

**SettingsPane**

Класът е наследник на `JPanel`. В него е реализирана функционалност за промяна на настройките на системата. Настройките се четат и записват във външен текстов файл, чието име е зададено като константа в класа. За позициониране на елементите на екрана е използван `GridBagLayout`. В класа са включени 33 визуални компонента, както следва:

- единадесет етикета `JLabel` с подравнен в дясно текст, пояснящи всяка една от настройките;
- девет текстови полета `JTextField`, които се запълват автоматично с избраните цветове съответно за отделните типове неврони и фонов цвят на работното поле;
- падащо меню `private JComboBox chooseNeuronsSize = new JComboBox()`, позволяващо на потребителя да избере големина на радиус на невроните в изкуствена невронна мрежа;

- падащо меню `private JComboBox chooseNeuronsNumbersSize = new JComboBox()`, съдържащо стойности за големина на идентификационни номера на невроните в изкуствена невронна мрежа;
- девет на брой бутона `JButton` за избор на цвят съответно на отделните типове неврони и фонов цвят на работното поле, като към всеки един бутон е прикачен `new ChooseNeuronColorListener()`, управляващ избора на цветове;
- бутон `private JButton save = new JButton(Texts.LABEL_BUTTON_SAVE_CAPS)`, отговарящ за запаметяването на настройките във файл;
- бутон `private JButton close = new JButton(Texts.LABEL_BUTTON_CLOSE_CAPS)`, при натискането на който екранът за промяна на настройки се затваря, без да се запаметяват промени.

В `SettingsPane` се съдържа един вътрешен клас:

- `private class ChooseNeuronColorListener implements ActionListener`, съдържащ метод `public void actionPerformed(ActionEvent event)`. Класът служи за отваряне на диалогов прозорец за избор на цвят. Диалоговият прозорец се отваря при натискане на бутон за избор на цвят на тип неврон или фон на работното поле. Цветът по подразбиране в прозореца е текущият цвят на избрания тип неврон/фон на работното поле. След като потребителят избере желания цвят и потвърди, съответният `JTextField` придобива за фон посочения цвят.

Налични са два анонимни класа:

- клас `new ActionListener()` е прикачен към бутона за запаметяване и съдържа метод `public void actionPerformed(ActionEvent event)`, който взима стойностите на настройките и ги запаметява във файл;
- клас `new ActionListener()` се използва от бутона за затваряне на екрана и съдържа метод `public void actionPerformed(ActionEvent event)`, който затваря панела с настройки и визуализира екрана за избор на изкуствена невронна мрежа за изчертаване.

Класът `SettingsPane` съдържа следните седем метода.

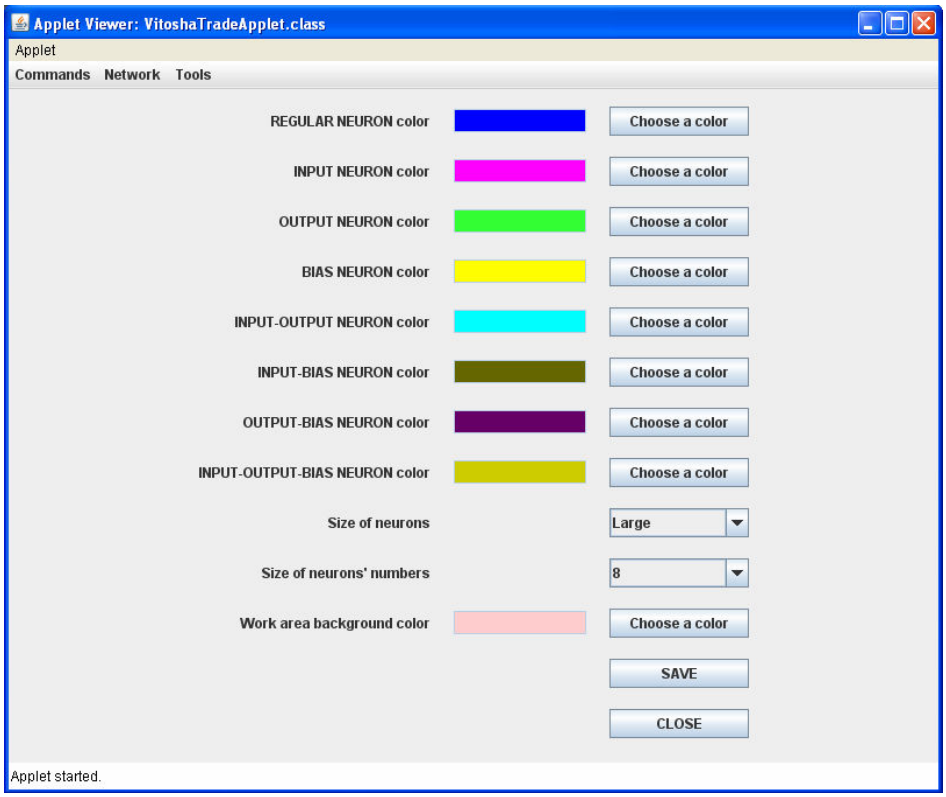
- Метод `public void readNeuronsTypesAndWorkAreaBackgroundColor(JTextField showNeuronColorField, String colorProperty)` се извиква за полетата, визуализиращи цветовете на отделните типове неврони и фона на работното поле. В метода се

осъществява четене от външен файл на съответната двойка „ключ-стойност” за конкретното поле за цвят, след което за фон на полето се задава прочетеният цвят. Аргументът `JTextField showNeuronColorField` съдържа името на текстовото поле, на което ще бъде зададен фон. Аргументът `String colorProperty` съдържа името (ключа от двойката „ключ-стойност” във файла със свойства) на параметъра, чиято стойност представлява съответният цвят във формат RGB.

- Метод `public void writeNeuronsTypesAndWorkAreaBackgroundColor(JTextField showNeuronColorField, String colorProperty, Properties properties)` се извиква в анонимния клас `new ActionListener()`, прикачен към бутона за запаметяване. Методът записва цветовете на типовете неврони и фона на работното поле във файл, във формат RGB. Аргументът `showNeuronColorField` съдържа името на полето, което е оцветено в съответния цвят. Аргументът `colorProperty` съдържа името (ключа от двойката „ключ-стойност” във файла със свойства) на параметъра, чиято стойност представлява съответният цвят във формат RGB. Аргументът `properties` съдържа обект от клас Properties и съдържа всички двойки „ключ-стойност” от външния файл, съдържащ стойностите за настройките на системата.
- Метод `public void readNeuronsSize()` се извиква в конструктора на класа `SettingsPane`. В него се осъществява прочитане на текущата стойност за настройката на големината на радиуса на невроните, след което автоматично в падащото меню за избор на големина на радиус на невроните се избира текущата стойност.
- Метод `public void writeNeuronsSize(Properties properties)` се извиква в анонимния клас `new ActionListener()`, прикачен към бутона за запаметяване. Методът записва във файл избраната стойност от падащото меню за големина на радиус на неврон. Аргументът `properties` приема обект от клас `Properties` и съдържа всички двойки „ключ-стойност” от съдържащият стойностите за настройките на системата външен файл.
- Метод `public void readNeuronsNumbersSize()` се извиква в конструктора на класа `SettingsPane`. Използва се за прочитане на текущата стойност за настройката на предпочитаната големина на идентификационните номера на невроните, след което автоматично в падащото меню за избор на големина на номерата се избира текущата стойност.

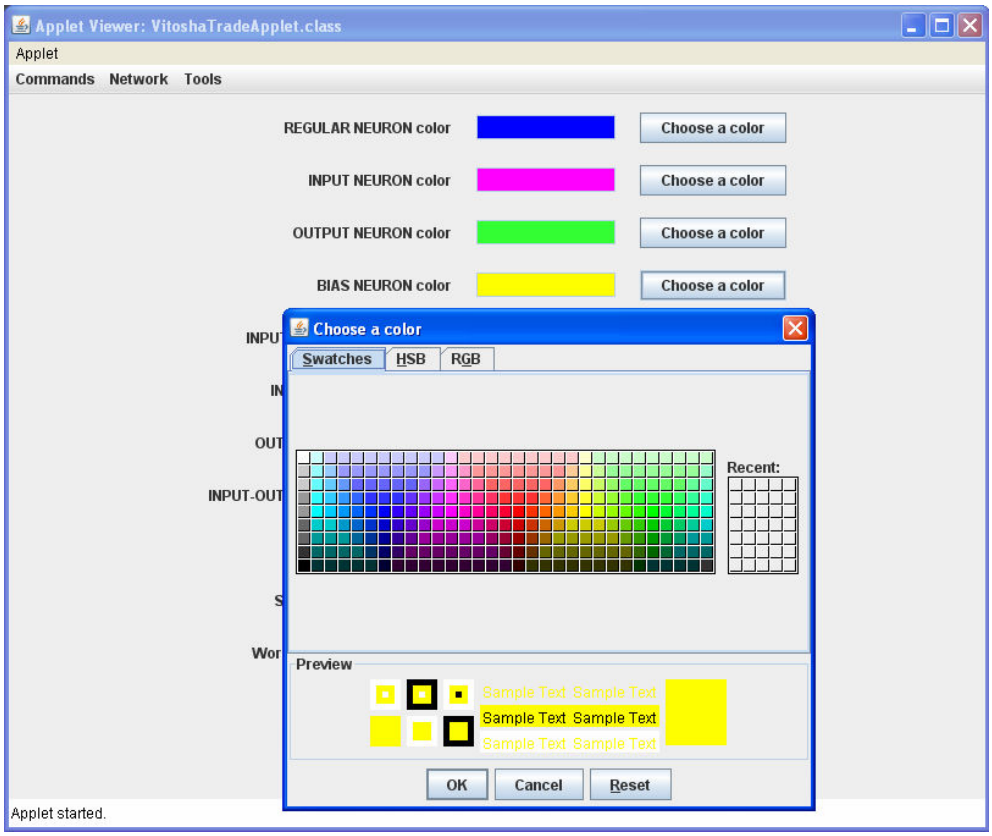
- Метод `public void writeNeuronsNumbersSize(Properties properties)` се извиква в анонимния клас `new ActionListener()`, прикачен към бутона за запаметяване. Методът записва избраната стойност от падащото меню за големина на номер на неврон във файл. Аргументът `properties` приема обект от клас `Properties` и в него са „заредени” всички двойки „ключ-стойност” от съдържащият стойностите за настройките на системата външен файл.
- Метод `public void arrangeElements(JComponent component, int gridX, int gridY, boolean enabled)` се използва за разполагане на визуалните елементи на екрана по подходящ и лесен за разбиране начин. Аргументът `JComponent component` приема компонентът, който ще бъде разположен в панела. Аргументът `int gridX` приема координатата на компонента по оста X. Аргументът int gridY приема координатата на компонента по оста Y. Аргументът boolean enabled определя дали компонентът е активен и потребителят може да работи с него или не (например бутоните са активни, а полетата с цветове не са).

Екранът за промяна на настройки е представен на Фиг.3.10.




Фиг.3.10 Екран за промяна на настройки

На Фиг.3.11 е представен екранът за промяна на настройки с прилежащ към него диалогов прозорец за избор на цвят.




Фиг.3.11 Екран за промяна на настройки с прилежащ към него диалогов прозорец за избор на цвят

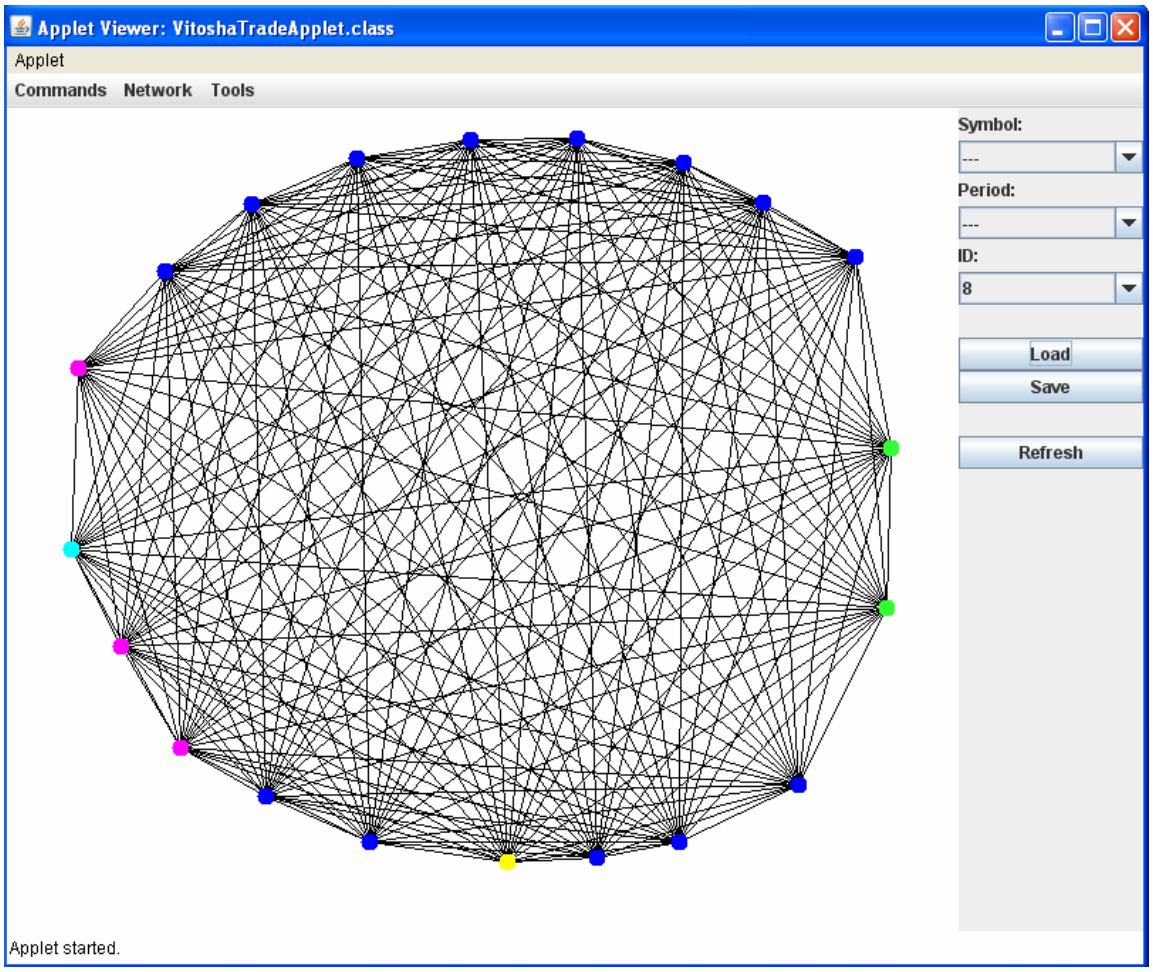
**DrawingPane**

Класът `DrawingPane` е наследник на `JPanel`. Той отговаря за изчертаването на избрана изкуствена невронна мрежа в работното поле и дава възможност на потребителя да я манипулира по лесен и удобен начин.

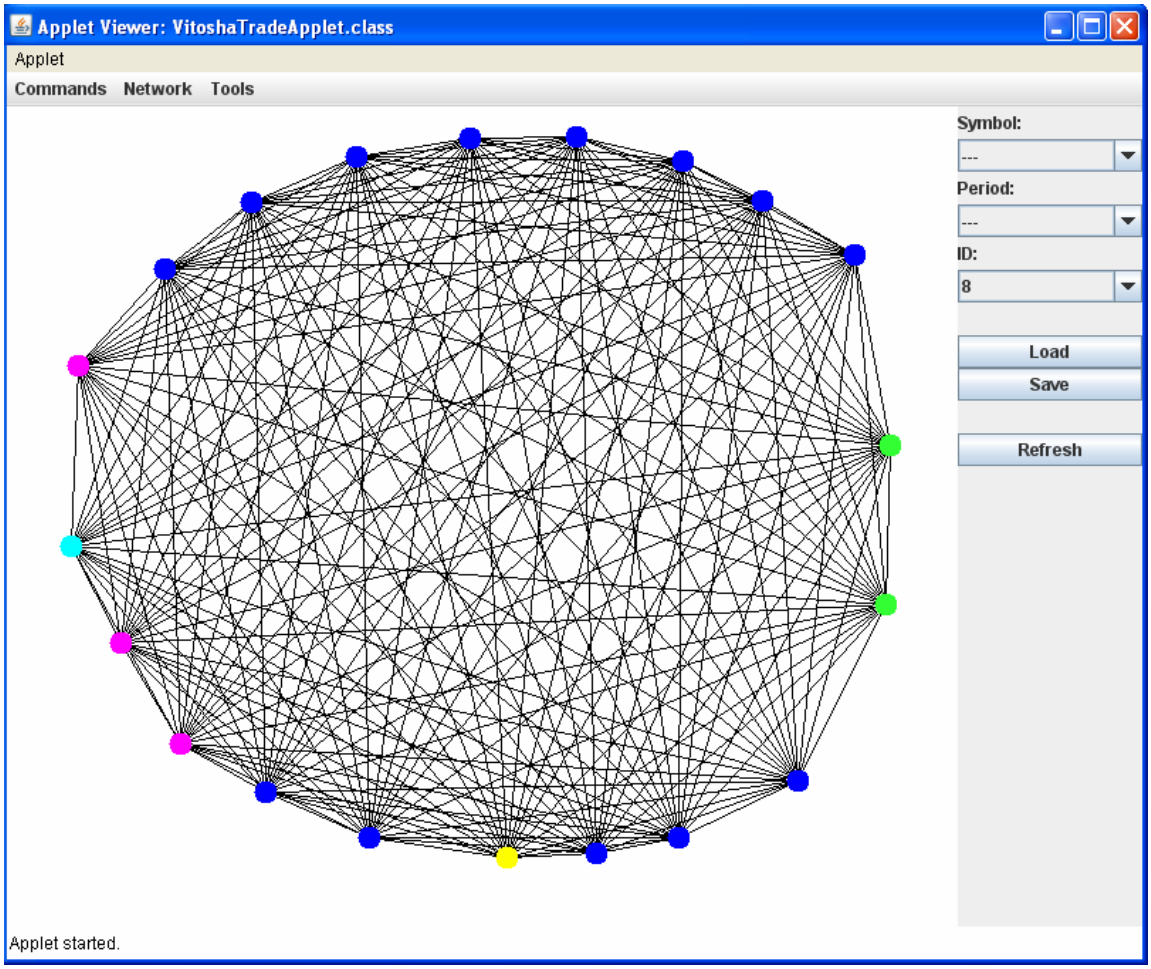
Към класа са добавени следните пет метода.

- Метод `private int calculateNeuronRadius()` отговаря за изчисляването на радиуса на невроните според зададената от потребителя настройка. Методът се извиква от друг метод в класа – `public void paint(Graphics g)` и връща радиуса в целочислен формат. Фиг.3.12 представя екран с изчертана изкуствена невронна мрежа, като радиусът на

невроните е най-малкият наличен в системата. Фиг.3.13 представя екран с изчертана изкуствена невронна мрежа, като радиусът на невроните е най-големият наличен в системата.



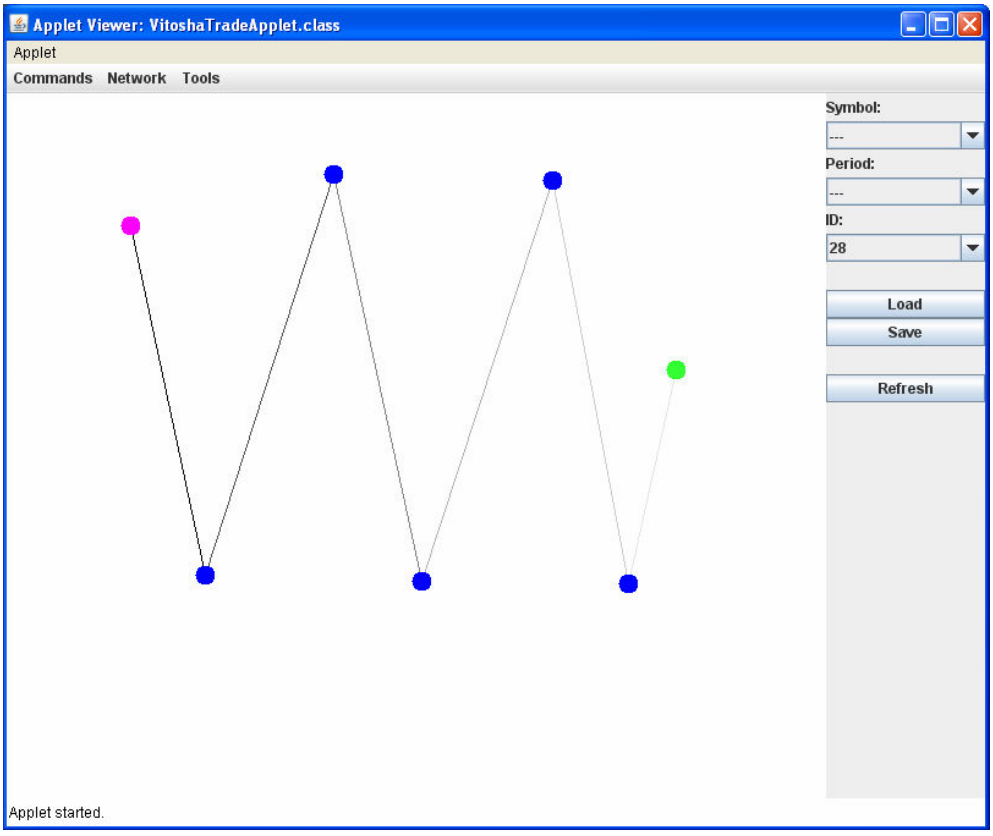

Фиг.3.12 Изкуствена невронна мрежа с неврони с най-малък размер на радиус




Фиг.3.13 Изкуствена невронна мрежа с неврони с най-голям размер на радиус

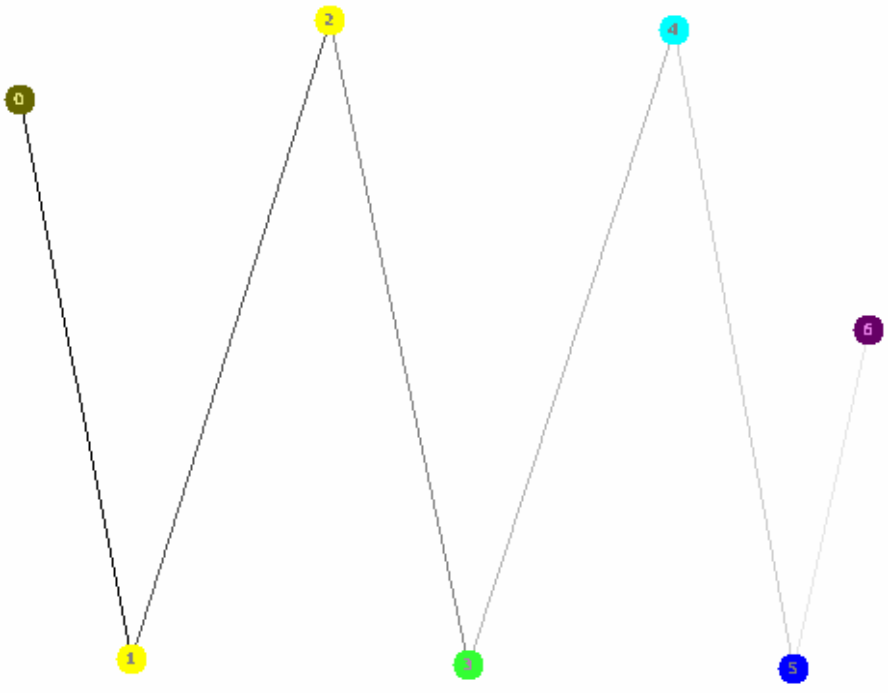
- Метод `public Color readNeuronColor(String colorProperty)` осъществява четене от външен файл на цвета на всеки един неврон, в зависимост от неговия тип. Методът се използва в друг метод в класа, а именно `public void paint(Graphics g)` и връща цвета на конкретния неврон. Аргументът `String colorProperty` съдържа името (ключа от двойката „ключ-стойност” във файла със свойства) на параметъра, чиято стойност представлява съответният цвят във формат RGB.
- Метод `public void selectConnectionsColorByChosenMesh(int firstNeuronIndex, int secondNeuronIndex, Graphics g)` изчертава на екрана връзките между невроните в зависимост от избрания от меню `meshMenu` начин за изчертаване на връзки. За визуализирането на връзките при избрана опция „изчертаване според активностите на

връзките”, „изчертаване според теглата на връзките” и „изчертаване според активностите и теглата едновременно”, се използва формула, която изчислява нюанс на сивия цвят според активностите и теглата на връзките. Така връзките биват начертани в сивата цветова гама, в зависимост от техните параметри. Връзки с по-големи активности/тегла се изчертават в по-тъмен нюанс на сивия цвят, а с по-малки – в по-светъл. Аргументът `int firstNeuronIndex` приема идентификационния номер на началния неврон на връзката. Аргументът `int secondNeuronIndex` приема идентификационния номер на крайния неврон на връзката. Аргументът `Graphics g` приема графичния контекст, в който се изчертава избраната изкуствена невронна мрежа. На Фиг.3.14 е представен екран с изчертана изкуствена невронна мрежа, като връзките между невроните са визуализирани според техните активности.

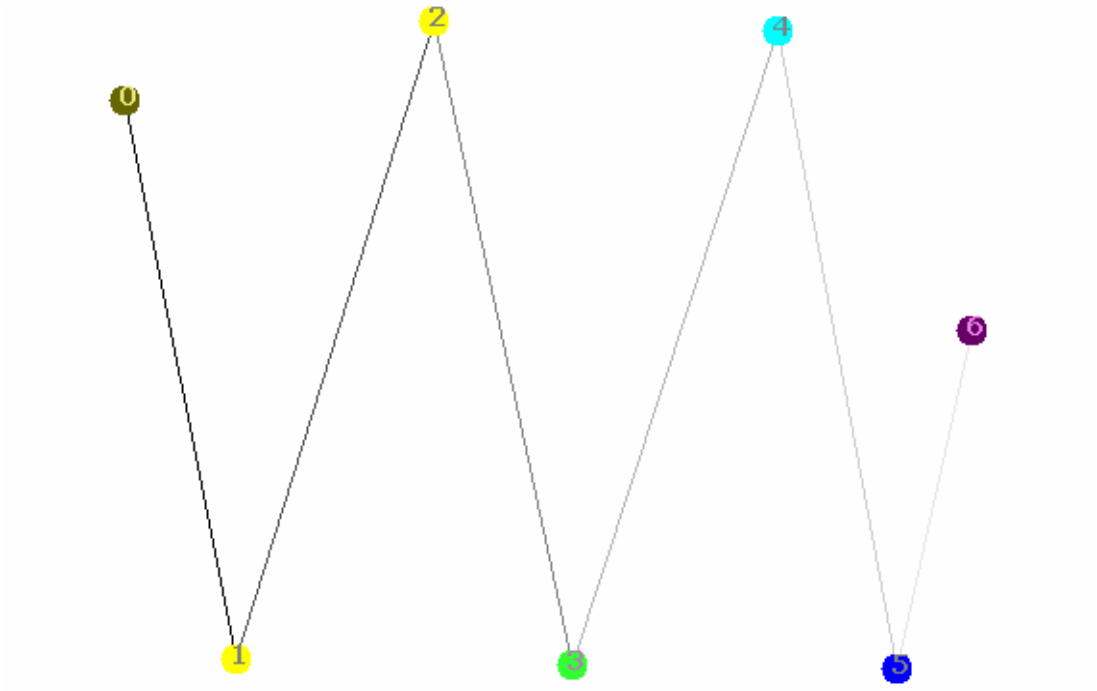



Фиг.3.14 Връзки на изкуствена невронна мрежа, изчертани в различен нюанс на сивия цвят според активностите им

- Метод `private void showNeuronNumbers(Graphics g, int neuronId, Color neuronColor, int neuronIdXCoordinate, int neuronIdYCoordinate)` чете от външен файл зададената големината на идентификационните номера на невроните, след което визуализира номерата на екрана. Методът се извиква в друг метод в класа – `public void paint(Graphics g)`. Аргументът `Graphics g` приема графичния контекст, в който се изчертава избраната изкуствена невронна мрежа. Аргументът `int neuronId` съдържа идентификационния номер на поредния изобразен неврон. Аргументът `Color neuronColor` приема цвета на неврона, нужен за определяне на цвета на номера на неврона така, че да бъде видим. Аргументите `int neuronIdXCoordinate, int neuronIdYCoordinate` приемат съответно координатата на номера на неврона по оста X и координатата на номера на неврона по оста Y. На Фиг.3.15 е представен екран с графично изобразена изкуствена невронна мрежа, чиито неврони са номерирани с най-малкия позволен размер на номерата. На Фиг.3.16. е изобразен екран с изчертана в работното поле изкуствена невронна мрежа, чиито неврони са номерирани с най-големия наличен в системата размер на номерата.

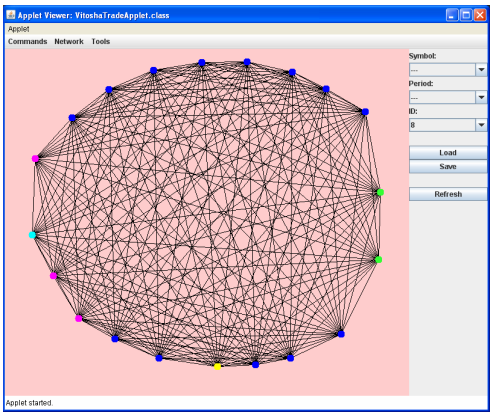



Фиг.3.15 Неврони в изкуствена невронна мрежа, номерирани с най-малък размер на номерата




Фиг.3.16 Неврони в изкуствена невронна мрежа, номерирани с най-голям размер на номерата

- Метод `private void selectBackgroundColor(Graphics g)` прилага фонов цвят на работното поле, като преди това го прочита от външния файл, съдържащ настройките на системата. Извиква се от друг метод на класа - `public void paint(Graphics g)`. Аргументът `Graphics g` приема графичния контекст, в който се изчертават изкуствени невронни мрежи. На Фиг.3.17 е изобразен екранът за избор и изчертаване на невронна мрежа, като работното поле е с избран от потребителя фонов цвят.




Фиг.3.17 Работно поле с избран от потребителя фонов цвят

**SymbolPeriodKeyValue**

Класът служи за задаване на двойки стойности в падащи менюта, като първата стойност е скрита (обикновено тя е идентификационен номер на втората стойност), а втората се изобразява в падащото меню. Прилага се към падащи менюта `JComboBox` по следния начин (следващият пример въвежда в падащо меню с времеви периоди идентификационен номер на период и символ на период): `networkPeriod.addItem(new SymbolPeriodKeyValue(listPeriod[i][0], listPeriod[i][1]))`. При избор на стойност от страна на потребителя, тя може да бъде присвоена на променлива по следния начин: `int selectedPeriod = ((SymbolPeriodKeyValue) networkPeriod.getSelectedItem()).getKey();`.

Посредством използването на двойки стойности в падащо меню потребителят избира измежду познатите му стойности, а като избрана такава системата разпознава съответния идентификационен номер .

Класът съдържа три на брой метода.

- Метод `public String getValue()` връща видимата в падащото меню стойност.
- Метод `public int getKey()` връща невидимия идентификационен номер, принадлежащ на стойността в падащото меню.
- Метод `public String toString()` преобразува видимата в падащото меню стойност в символен низ.

**InformationMessages**

Класът се грижи за инициализиране на диалогови прозорци, съдържащи съобщения с информация. Обикновено се използва за уведомяване на потребителя при некоректно въведена от него информация, а именно: при грешни идентификационни данни на екрана за вход, некоректен идентификационен номер при изтриване на изкуствена невронна мрежа, при неприемливи стойности в полетата при създаване на нова мрежа.

При извикването си класът получава текста на съобщението, заглавието на диалоговия прозорец и типа на съобщението („информация”, „грешка” и т.н.).

В класа се съдържа един метод, а именно:

- `void showMessage()`, визуализиращо на екрана диалогов прозорец със съобщение и бутон „ОК”.

**DatabaseHelper**

Класът е отговорен за осъществяване на връзка към базата от данни и извличане на информация от нея. В класа са дефинирани всички необходими за правилното функциониране на системата SQL заявки.

Към класа са добавени следните четири метода.

- Метод `public String[][] loadCurrencyPairs()` предоставя списък с наличните в базата от данни валутни двойки. Връща масив от символни низове, съдържащ символите на валутните двойки.
- Ролята на метода `public String[][] loadPeriods()` е да извлече от базата от данни информация за времевите периоди, върху които работят изкуствените невронни мрежи. Връща двумерен масив, съдържащ двойки „идентификационен номер/символ” за наличните периоди.
- Метод `public int saveNewAnn(String annSymbol, int annPeriod, int annNumberNeurons, String annWeights, int annFitness, String annFlags)` въвежда в базата от данни информация за нова изкуствена невронна мрежа, въведена от потребител. Аргументът `String annSymbol` приема стойност валутната двойка за новата мрежа. Аргументът `int annPeriod` приема идентификационния номер на времевия период, върху който работи новата мрежа. Аргументът `int annNumberNeurons` взима броя на невроните в изкуствената невронна мрежа. Аргументът `String annActivitiesWeights` съдържа активностите и теглата на връзките, разделени с интервал. При създаване на нова мрежа активностите и теглата имат еднакви стойности, поради което в базата от данни се записва една и съща стойност за активности и тегла. Аргументът `int annFitness` приема стойност по подразбиране, показваща ефективността на невронната мрежа. Аргументът `String annFlags` съдържа флаговете (типовете) на невроните в мрежата, разделени с интервал.
- Метод `public void deleteAnn(int id)` премахва от базата данни информация за изкуствена невронна мрежа с идентификационен номер аргумента на метода `int id`.

**Texts**

Клас `Texts` съхранява всички текстове, съобщения и надписи, използвани в аплета, включително текстовете на менютата. Всички текстове са константи и могат да бъдат използвани без предварително да бъде създаден обект от класа.

Класът не съдържа методи, вътрешни и анонимни класове.

Общият брой редове програмен код, съставен от дипломанта, е 2414, като в него се включват и коментарите.

### *3.6 Анализ на качеството на програмния код*

За установяване на качеството на програмния код на аплета са разгледани и анализирани няколко основни метрики. В анализа са включени класовете и методите, разработени от дипломанта.

Разгледаните метрики, отнасящи се до класовете в системата, са:

- брой редове код;
- сложност на методите на класа;
- дълбочина на дървото на наследяване;
- липса на кохезия между методите.

Метриките, разгледани на ниво „метод”, са следните:

- брой редове код;
- брой параметри;
- дълбочина на вложените блокове;
- цикломатична сложност на МакКейб.

**Брой редове код на ниво „клас”**

Посочената метрика включва броя на редовете код, като празните редове и коментарите не се взимат предвид. В случай че даден клас съдържа вътрешни или анонимни класове, същите не са разгледани отделно.

В Таблица 3.6 е представен броят на редовете код за всеки клас.

Таблица 3.6

| **Брой редове код на ниво „клас”.** | |
|---|---|
| **ИМЕ НА КЛАС** | **БРОЙ РЕДОВЕ КОД** |
| SettingsPane | 319 |
| NewAnnPane | 109 |
| Texts | 74 |
| DeleteAnnPane | 64 |
| InformationMessages | 17 |
| SymbolPeriodKeyValue | 17 |

Общоприетият максимален брой на редове код в един клас е 750. Всички класове в приложението са в границата и не превишават максималната стойност на метриката.

**Сложност на методите на класа**

В случай че даден клас съдържа вътрешни класове, сложността на методите на вътрешните класове са разгледани отделно. Анонимни класове не са включени. В Таблица 3.7 е посочена сложността на методите в системата за всеки клас.

Таблица 3.7

| Сложност на методите на класа. | |
|---|---|
| **ИМЕ НА КЛАС** | **СЛОЖНОСТ НА МЕТОДИТЕ НА КЛАСА** |
| SetFiltersBySymbolAndPeriod (вътрешен за клас NetworkPane) | 31 |
| SettingsPane | 12 |
| ChooseNeuroinColorListener (вътрешен за клас SettingsPane) | 3 |
| NewAnnPane | 5 |
| DeleteAnnPane | 4 |
| SymbolPeriodKeyValue | 4 |
| InformationMessages | 2 |
| Texts | 0 |

За максимална допустима граница на метриката се счита стойност 20. Дадените стойности показват, че по-сложен и труден за поддръжка от останалите класове е клас `SetFiltersBySymbolAndPeriod`. Вероятността за програмни грешки в посочения клас е по-голяма и е препоръчително той да бъде преработен в следващата версия на приложението.

**Дълбочина на дървото на наследяване**

При установяване на стойностите на метриката вътрешните класове са разгледани отделно. Таблица 3.8 онагледява дълбочината на дървото на наследяване за всеки клас.

Таблица 3.8

| Дълбочина на дървото на наследяване. | |
|---|---|
| **ИМЕ НА КЛАС** | **ДЪЛБОЧИНА НА ДЪРВОТО НА НАСЛЕДЯВАНЕ** |
| DeleteAnnPane | 5 |
| SetFiltersBySymbolAndPeriod (вътрешен за клас NetworkPane) | 1 |
| SettingsPane | 5 |
| ChooseNeuronColorListener (вътрешен за клас SettingsPane) | 1 |
| NewAnnPane | 5 |
| InformationMessages | 1 |
| SymbolPeriodKeyValue | 1 |
| Texts | 1 |

Прието е, че дълбочина на дървото на наследяване по-голяма от 5 повишава сложността на разработката. От дадените стойности в Таблица 3.8 може да се направи заключение, че сложността на приложението не е много висока, което води до редуциране на броя на потенциалните програмни грешки.

**Липса на кохезия между методите.**

Класове и вътрешни класове са разгледани отделно. Таблица 3.9 представя стойностите на метриката за всеки клас, с изключение на анонимните.

Таблица 3.9

| Липса на кохезия между методите. | |
|---|---|
| **ИМЕ НА КЛАС** | **ЛИПСА НА КОХЕЗИЯ МЕЖДУ МЕТОДИТЕ** |
| SettingsPane | 0.972 |
| ChooseNeuronColorListener (вътрешен за клас SettingsPane) | 0 |
| DeleteAnnPane | 0.5 |
| SymbolPeriodKeyValue | 0.5 |
| SetFiltersBySymbolAndPeriod | 0 |

| (вътрешен за клас NetworkPane) | |
|---|---|
| NewAnnPane | 0 |
| InformationMessages | 0 |
| Texts | 0 |

Ниските стойности на метриката сочат голяма степен на свързаност между методите, което е знак за качеството на класа. Стойности, близки до 1, показват висока липса на кохезия и предполагат разделяне на класа на няколко класа или подкласа. От данните в Таблица 3.9. може да се направи извод, че кохезията между методите в класовете е задоволителна. Единственият клас с ниска кохезия е `SettingsPane`, притежаваща стойност за липса на кохезия 0.972.

**Брой редове код на ниво „метод”**

В Таблица 3.10 е представен броя на редовете код на ниво „метод”.

Таблица 3.10

| **Брой редове код на ниво „метод”.** | | |
|---|---|---|
| **ИМЕ НА КЛАС** | **ИМЕ НА МЕТОД** | **БРОЙ РЕДОВЕ КОД** |
| **SettingsPane** | readNeuronsTypesAndWorkAreaBackgroundColor | 11 |
| | readNeuronsNumbersSize | 10 |
| | readNeuronsSize | 9 |
| | arrangeElements | 5 |
| | writeNeuronsNumbersSize | 4 |
| | writeNeuronsTypesAndWorkAreaBackgroundColor | 3 |
| | writeNeuronsSize | 3 |
| **ChooseNeuronColorListener (вътрешен за клас SettingsPane)** | actionPerformed | 6 |
| **DrawingPane** | selectConnectionsColorByChosenMesh | 66 |
| | showNeuronNumbers | 26 |
| | calculateNeuronRadius | 24 |
| | readNeuronColor | 12 |
| | selectBackgroundColor | 11 |
| **NewAnnPane** | showInformationMessage | 5 |

| | | |
|---|---|---|
| **VitoshaTradeApplet** | showMenuBarAfterLogin | 23 |
| | showNewAnnPane | 10 |
| | showDeleteAnnPane | 10 |
| | hideMenuBarAfterLogout | 10 |
| | showSettingsPane | 9 |
| **DatabaseHelper** | saveNewAnn | 31 |
| | loadPeriods | 22 |
| | loadCurrencyPairs | 21 |
| | deleteAnn | 11 |
| **NetworkPane** | showInformationMessage | 5 |
| **SetFiltersBySymbolAndPeriod (вътрешен за клас NetworkPane)** | setFilteredAnnIdsBySymbolAndPeriod | 13 |
| | setFilteredAnnIdsByPeriod | 12 |
| | setFilteredAnnIdsBySymbol | 11 |
| | setAllAnnIds | 5 |
| **LoginPane** | showConnectionError | 3 |
| **DeleteAnnPane** | loadAllAnnIds | 6 |
| | showInformationMessage | 5 |
| **InformationMessages** | showMessage | 2 |
| **SymbolPeriodKeyValue** | getValue | 1 |
| | getKey | 1 |
| | toString | 1 |

За максимална стойност на метриката в добре структуриран програмен код се счита 50 реда на метод. Единствено метод `selectConnectionsColorByChosenMesh`, принадлежащ към клас `DrawingPane`, не отговаря на добрата практика. Би следвало методът да се оптимизира в следващата версия на приложението.

### Брой параметри на ниво „метод”

Методите във вътрешните класове са разгледани отделно. В Таблица 3.11 е представен броят на параметрите на всеки метод.

Таблица 3.11

| Брой параметри на ниво „метод”. | | |
|---|---|---|
| **ИМЕ НА КЛАС** | **ИМЕ НА МЕТОД** | **БРОЙ ПАРАМЕТРИ** |
| **DatabaseHelper** | saveNewAnn | 6 |
| | deleteAnn | 1 |
| | loadCurrencyPairs | 0 |
| | loadPeriods | 0 |
| **DrawingPane** | showNeuronNumbers | 5 |
| | selectConnectionsColorByChosenMesh | 3 |
| | readNeuronColor | 1 |
| | selectBackgroundColor | 1 |
| | calculateNeuronRadius | 0 |
| **SettingsPane** | arrangeElements | 4 |
| | writeNeuronsTypesAndWorkAreaBackgroundColor | 3 |
| | readNeuronsTypesAndWorkAreaBackgroundColor | 2 |
| | writeNeuronsSize | 1 |
| | writeNeuronsNumbersSize | 1 |
| | readNeuronsNumbersSize | 0 |
| | readNeuronsSize | 0 |
| **ChooseNeuronColorListener (вътрешен за клас SettingsPane)** | actionPerformed | 1 |
| **InformationMessages** | showMessage | 3 |
| **NewAnnPane** | showInformationMessage | 0 |
| **VitoshaTradeApplet** | showMenuBarAfterLogin | 0 |
| | showNewAnnPane | 0 |
| | showDeleteAnnPane | 0 |
| | hideMenuBarAfterLogout | 0 |
| | showSettingsPane | 0 |
| **NetworkPane** | showInformationMessage | 0 |
| **SetFiltersBySymbolAndPeriod (вътрешен за клас NetworkPane)** | setFilteredAnnIdsBySymbolAndPeriod | 0 |
| | setFilteredAnnIdsByPeriod | 0 |
| | setFilteredAnnIdsBySymbol | 0 |
| | setAllAnnIds | 0 |
| **LoginPane** | showConnectionError | 0 |
| **DeleteAnnPane** | loadAllAnnIds | 0 |

| | | |
|---|---|---|
| | showInformationMessage | 0 |
| **SymbolPeriodKeyValue** | getValue | 0 |
| | getKey | 0 |
| | toString | 0 |

За максимална стойност на метриката е определен брой от 5 параметъра. Всички използвани методи са в границата, с изключение на `saveNewAnn`, принадлежащ към клас `DatabaseHelper`. Той получава 6 параметъра при извикване и подлежи на евентуална промяна в структурата.

**Дълбочина на вложените блокове.**

Метриката е разгледана за методите на всички класове, включително на вътрешните. В Таблица 3.12 са представени стойностите за дълбочина на вложените блокове на методите.

Таблица 3.12

| **Дълбочина на вложените блокове.** | | |
|---|---|---|
| **ИМЕ НА КЛАС** | **ИМЕ НА МЕТОД** | **ДЪЛБОЧИНА** |
| **DatabaseHelper** | loadCurrencyPairs | 4 |
| | loadPeriods | 4 |
| | saveNewAnn | 3 |
| | deleteAnn | 2 |
| **DrawingPane** | calculateNeuronRadius | 3 |
| | selectConnectionsColorByChosenMesh | 3 |
| | showNeuronNumbers | 3 |
| | readNeuronColor | 2 |
| | selectBackgroundColor | 2 |
| **DeleteAnnPane** | loadAllAnnIds | 2 |
| | showInformationMessage | 1 |
| **VitoshaTradeApplet** | showMenuBarAfterLogin | 2 |
| | showNewAnnPane | 1 |
| | showDeleteAnnPane | 1 |
| | hideMenuBarAfterLogout | 1 |

| | | |
|---|---|---|
| | showSettingsPane | 1 |
| **NetworkPane** | showInformationMessage | 1 |
| **SetFiltersBySymbolAndPeriod (вътрешен за клас NetworkPane)** | setFilteredAnnIdsBySymbolAndPeriod | 3 |
| | setFilteredAnnIdsByPeriod | 3 |
| | setFilteredAnnIdsBySymbol | 3 |
| | setAllAnnIds | 2 |
| **SettingsPane** | readNeuronsTypesAndWorkAreaBackgroundColor | 2 |
| | readNeuronsSize | 2 |
| | readNeuronsNumbersSize | 2 |
| | writeNeuronsTypesAndWorkAreaBackgroundColor | 1 |
| | writeNeuronsSize | 1 |
| | writeNeuronsNumbersSize | 1 |
| | arrangeElements | 1 |
| **ChooseNeuronColorListener (вътрешен за клас SettingsPane)** | actionPerformed | 1 |
| **InformationMessages** | showMessage | 1 |
| **NewAnnPane** | showInformationMessage | 1 |
| **LoginPane** | showConnectionError | 1 |
| **SymbolPeriodKeyValue** | getValue | 1 |
| | getKey | 1 |
| | toString | 1 |

За добра стройност на дълбочината на вложените блокове се счита всяка стойност, по-малка или равна на 5. Посечените в Таблица 3.12 стойности удовлетворяват това условие, което допринася за високото качество на приложението.

### Цикломатична сложност на МакКейб

При установяване на стойностите на метриката методите на вътрешните класове са разгледани отделно. Таблица 3.13 представя цикломатичната стойност на МакКейб за всеки метод.

Таблица 3.13

| **Цикломатична сложност на МакКейб.** | | |
|---|---|---|
| **ИМЕ НА КЛАС** | **ИМЕ НА МЕТОД** | **СЛОЖНОСТ** |
| **DrawingPane** | selectConnectionsColorByChosenMesh | 13 |
| | calculateNeuronRadius | 5 |
| | showNeuronNumbers | 3 |
| | readNeuronColor | 2 |
| | selectBackgroundColor | 2 |
| **DatabaseHelper** | saveNewAnn | 5 |
| | loadPeriods | 4 |
| | loadCurrencyPairs | 4 |
| | deleteAnn | 2 |
| **VitoshaTradeApplet** | showMenuBarAfterLogin | 5 |
| | showNewAnnPane | 1 |
| | showDeleteAnnPane | 1 |
| | hideMenuBarAfterLogout | 1 |
| | showSettingsPane | 1 |
| **DeleteAnnPane** | loadAllAnnIds | 2 |
| | showInformationMessage | 1 |
| **NetworkPane** | showInformationMessage | 1 |
| **SetFiltersBySymbolAndPeriod (вътрешен за клас NetworkPane)** | setFilteredAnnIdsBySymbolAndPeriod | 3 |
| | setFilteredAnnIdsByPeriod | 3 |
| | setFilteredAnnIdsBySymbol | 3 |
| | setAllAnnIds | 2 |
| **SettingsPane** | readNeuronsTypesAndWorkAreaBackgroundColor | 2 |
| | readNeuronsSize | 2 |
| | readNeuronsNumbersSize | 2 |
| | writeNeuronsTypesAndWorkAreaBackgroundColor | 1 |
| | writeNeuronsSize | 1 |
| | writeNeuronsNumbersSize | 1 |
| | arrangeElements | 1 |
| **ChooseNeuronColorListener (вътрешен за клас SettingsPane)** | actionPerformed | 2 |
| **InformationMessages** | showMessage | 1 |
| **NewAnnPane** | showInformationMessage | 1 |

| | | |
|---|---|---|
| **LoginPane** | showConnectionError | 1 |
| **SymbolPeriodKeyValue** | getValue | 1 |
| | getKey | 1 |
| | toString | 1 |

За допустима максимална цикломатична стойност се счита 10. Всички разгледани методи притежават стойност по-малка от 10, с изключение на функцията `selectConnectionsColorByChosenMesh`, принадлежаща към клас `DrawingPane`. Цикломатичната стойност на метод `selectConnectionsColorByChosenMesh` е 13, което сочи предразположен към грешки програмен код.

В заключение би могло да се каже, че разработеният аплет притежава добро качество, що се отнася до разгледаните метрики. Неудовлетворителните стойности следва да бъдат анализирани и пораждащият ги програмен код да бъде оптимизиран.

### *3.7 Публикуване на проекта в SourceForge*

Софтуерът за прогнозиране на вътрешни пазари VitoshaTrade е проект с отворен код. Поради тази причина системата за графично изобразяване и визуално манипулиране на изкуствени невронни мрежи, проектирана да подпомага използването на VitoshaTrade, също е с отворен код. И двата проекта са публикувани в най-големия доставчик на ресурси за публикуване на софтуерни продукти с отворен код SourceForge.

Програмният код и всички прилежащи към него допълнителни ресурси като документи и използвани библиотеки могат да бъдат намерени на адрес http://vitoshatrade.svn.sourceforge.net/viewvc/vitoshatrade/trunk/.

# Заключение

Дипломната работа разглежда подробно планирането и разработката на информационна система за графично изобразяване и визуално манипулиране на изкуствени невронни мрежи. По време на разработката могат да се отличат няколко етапа - анализ на съществуващи методи и системи предлагащи възможност за изчертаване и работа с изкуствени невронни мрежи, избор на технология за реализация, проектиране и реализация, анализ на качеството на програмния код.

При разработката на приложението за визуализиране и манипулиране на изкуствени невронни мрежи е използвана среда за разработка Eclipse SDK версия 3.4.2. и, система за управление на бази от данни MySQL 5.5.13, уеб-базиран инструмент за администриране на MySQL – PhpMyAdmin 3.4.3.1.

За създаване на UML диаграми е ползван софтуер Visual Paradigm for UML Community Edition.

За създаване на физически и концептуален модел на базата данни е използван софтуерът Sybase PowerDesigner V12.5 Evaluation.

За анализ на качеството на програмния код на приложението е използван допълнителен софтуер, инсталиран към Eclipse SDK, а именно Metrics 1.3.6.

По време на проектирането и разработката на уеб-базираната система, освен посочените в изложението на дипломната работа, са използвани и източници на информация [1], [3], [4], [5], [6], [10], [12], [15], [20] (виж Използвана литература).

В близко бъдеще се предвижда усъвършенстване и доразвитие на системата. Ще бъдат добавени опции за прибавяне на невронни единици към изкуствените невронни мрежи, както и възможност за премахване на такива. Планира се преработка на методите, дали недобри резултати при анализа на качеството на програмния код.

# Използвана литература

[1] Екел Брус (2001). Да мислим на Java, том I. СофтПрес ООД.

[2] Екел Брус (2001). Да мислим на Java, том II. СофтПрес ООД.

[3] Пенева Юлиана (2005). БАЗИ ОТ ДАННИ. РЕГАЛИЯ 6.

[4] Тупаров Георги (2008). Лекционен материал към курс „Обектно-ориентирано моделиране”.

[5] Pender Thomas (2002). UML Weekend Crash Course. Wiley Publishing, Inc.

[6] Вътков Богдан. "Програма за автоматичен анализ на софтуерни метрики". 2005г.

[7] Наков Светлин. "Интернет програмиране с Java". 25.11.2002г. <http://inetjava.sourceforge.net/lectures/InetJava-textbook-draft-by-Svetlin-Nakov.pdf>. (11.11.2011г.).

[8] Foley Mary-Jo. "Will there be a Silverlight 6". 08.11.2011г. <http://www.zdnet.com/blog/microsoft/will-there-be-a-silverlight-6-and-does-it-matter/11180>. (10.11.2011г.).

[9] Streeter Matthew J., Ward Matthew O., Alvarez Sergio A. "$N^2$VIS: An Interactive Visualization Tool for Neural Networks". <ftp://ftp.cs.wpi.edu/pub/techreports/pdf/00-11.pdf>. (03.11.2011г.).

[10] "Софтуерни метрики". 20.05.2010г. <http://bg.wikipedia.org/wiki/Софтуерни_метрики>. (01.01.2012г.).

[11] "Adobe Flash". <http://bg.wikipedia.org/wiki/Adobe_Flash>. (06.11.2011г.).

[12] "ColorChooser Sample 1". <http://www.java2s.com/Code/Java/Swing-JFC/ColorChooserSample1.htm>. (21.10.2011г.).

[13] "Java applet". <http://en.wikipedia.org/wiki/Java_applet>. (11.11.2011г.).

[14] "MathWorks". <http://en.wikipedia.org/wiki/MathWorks>. (05.11.2011г.).

[15] "Metrics 1.3.6 - Getting started". <http://metrics.sourceforge.net>. (18.12.2011г.).

[16] "Microsoft Silverlight". <http://en.wikipedia.org/wiki/Microsoft_Silverlight>. (10.11.2011г.).

[17] "Neural Network Toolbox". <http://www.mathworks.com/products/neural-network>. (03.01.2012г.).

[18] "NEVT NeuroEvolution Visualization Tool". <http://nevt.sourceforge.net>. (05.11.2011г.).

[19] "SharpNEAT Evolution of Neural Networks". <http://sharpneat.sourceforge.net>. (05.11.2011г.).

[20] "Sybase® PowerDesigner® Conceptual Data Model User's Guide". Ноември 2004г. <http://download.sybase.com/pdfdocs/pdd1100e/cdug.pdf>. (19.12.2011г.).

[21] <http://www.adobe.com>. (06.11.2011г.)

[22] <http://www.microsoft.com/silverlight/>. (10.11.2011г.).

# Приложение 1 Предписания към Use case модела на информационната система

**Предписания за Use case “Визуализиране на изкуствена невронна мрежа”.**

1. Предпоставки.

Потребителят желае да визуализира на екрана изкуствена невронна мрежа.

2. Предусловия.

Потребителят трябва да влезе в системата с коректно потребителско име и парола.

3. Стартиране на Use case.

Системата трябва да е в изправност. Не трябва да съществуват събития, които да възпрепятстват влизането на потребителя в нея (спиране на тока, прекъсната връзка с Интернет, хардуерен проблем с компютъра на потребителя).

4. Диалог.

- Потребителят въвежда коректни хост, порт, потребителско име и парола за връзка на системата към базата от данни.
- Системата инициализира начален екран за визуализиране на изкуствени невронни мрежи, съдържащ работно поле за изчертаване, падащи менюта за избор на валутна двойка, времеви период и идентификационен номер на наличните изкуствени невронни мрежи, както и бутони за зареждане на информацията в падащите менюта, запаметяване на промени в избрана невронна мрежа и изчертаване на топологията на мрежата в работното поле.
- Потребителят натиска бутон за зареждане в падащото меню на наличните в системата валутни двойки, времеви периоди и идентификационни номера на невронни мрежи.
- Системата обновява стойностите в падащите менюта.
- Потребителят избира идентификационен номер на мрежа и натиска бутон за зареждането ѝ на екрана.
- Системата визуализира топологията на избраната невронна мрежа.

5. Прекратяване на Use case.

Потребителят излиза от системата или избира друго меню.

6. След-условия.

Потребителят е набрал нужната му информация за невронната мрежа чрез визуализирането й.

**Предписания за Use case “Търсене на изкуствена невронна мрежа”.**

1. Предпоставки.

Потребителят желае да намери конкретна изкуствена невронна мрежа.

2. Предусловия.

Потребителят трябва да влезе в системата с коректно потребителско име и парола.

3. Стартиране на Use case.

Системата трябва да е в изправност. Не трябва да съществуват събития, които да възпрепятстват влизането на потребителя в нея (спиране на тока, прекъсната връзка с Интернет, хардуерен проблем с компютъра на потребителя).

4. Диалог.

- Потребителят въвежда коректни хост, порт, потребителско име и парола за връзка на системата към базата от данни.
- Системата инициализира начален екран за визуализиране на изкуствени невронни мрежи, съдържащ работно поле за изчертаване, падащи менюта за избор на валутна двойка, времеви период и идентификационен номер на наличните изкуствени невронни мрежи, както и бутони за зареждане на информацията в падащите менюта, запаметяване на промени в избрана невронна мрежа и изчертаване на топологията на мрежата в работното поле.
- Потребителят натиска бутон за зареждане в падащото меню на наличните в системата валутни двойки, времеви периоди и идентификационни номера на невронни мрежи.
- Системата обновява стойностите в падащите менюта.
- Потребителят избира валутна двойка и/или времеви период, по които търси конкретна невронна мрежа.

- Системата зарежда в падащото меню с идентификационни номера номерата на мрежите, притежаващи избраните от потребителя валутна двойка и/или времеви период.

5. Прекратяване на Use case.

Потребителят натиска бутон за визуализиране на желаната невронна мрежа, избира друго меню или излиза от системата .

6. След-условия.

Потребителят е намерил търсената невронна мрежа.

**Предписания за Use case “Визуализиране на параметри на изкуствена невронна мрежа”.**

1. Предпоставки.

Потребителят желае да разгледа параметрите на връзките и невроните в конкретна изкуствена невронна мрежа.

2. Предусловия.

Потребителят трябва да влезе в системата с коректни хост, порт, потребителско име и парола за връзка на системата към базата от данни и да зареди желаната невронна мрежа.

3. Стартиране на Use case.

Системата трябва да е в изправност. Не трябва да съществуват събития, които да възпрепятстват влизането на потребителя в нея (спиране на тока, прекъсната връзка с Интернет, хардуерен проблем с компютъра на потребителя).

4. Диалог.

- От вече заредена и изчертана на екрана изкуствена невронна мрежа потребителят избира неврон или връзка, чиито параметри желае да разгледа.
- Системата визуализира стойностите на параметрите за избрания неврон или връзка.

5. Прекратяване на Use case.

Потребителят излиза от системата или натиска друго меню.

6. След-условия.

Потребителят е направил справка за стойностите на параметрите на изкуствена невронна мрежа.

**Предписания за Use case “Промяна на параметри на изкуствена невронна мрежа”.**

1. Предпоставки.

Потребителят желае да промени параметрите на връзките и невроните в конкретна изкуствена невронна мрежа.

2. Предусловия.

Потребителят трябва да влезе в системата с коректни хост, порт, потребителско име и парола за връзка на системата към базата от данни, да зареди желаната невронна мрежа и да избере неврон или връзка, чиито параметри би желал да промени.

3. Стартиране на Use case.

Системата трябва да е в изправност. Не трябва да съществуват събития, които да възпрепятстват влизането на потребителя в нея (спиране на тока, прекъсната връзка с Интернет, хардуерен проблем с компютъра на потребителя).

4. Диалог.

- За вече заредена и изчертана на екрана изкуствена невронна мрежа и визуализирани стойности на параметрите на избран неврон или връзка, потребителят прави промени в стойностите на параметрите.
- Системата изчертава наново невронната мрежа в зависимост от направените промени по стойностите на параметрите й.
- Потребителят натиска бутон за запаметяване на промените по стойностите на параметрите.
- Системата запаметява новите стойности.

5. Прекратяване на Use case.

Потребителят излиза от системата или натиска друго меню.

6. След-условия.

Потребителят е направил промени в стойностите на параметрите на изкуствена невронна мрежа.

**Предписания за Use case “Изчертаване на връзките в различни цветове според параметрите им”.**

1. Предпоставки.

Потребителят желае да разгледа връзките между невроните в дадена изкуствена невронна мрежа, изчертани в по-блед или по-тъмен цвят, в зависимост от стойностите на параметрите на връзките.

2. Предусловия.

Потребителят трябва да влезе в системата с коректни хост, порт, потребителско име и парола за връзка на системата към базата от данни и да зареди желаната невронна мрежа.

3. Стартиране на Use case.

Системата трябва да е в изправност. Не трябва да съществуват събития, които да възпрепятстват влизането на потребителя в нея (спиране на тока, прекъсната връзка с Интернет, хардуерен проблем с компютъра на потребителя).

4. Диалог.

- Потребителят избира от меню начина на задаване на цветове на връзките – само според активностите им, само според теглата им, според активностите и теглата им или независимо от активностите и теглата им.
- Системата изчертава наново невронната мрежа, като оцветява връзките в зависимост от избрания тип на изчертаването им.

5. Прекратяване на Use case.

Потребителят излиза от системата или натиска друго меню.

6. След-условия.

Потребителят е разгледал връзките на изкуствената невронна мрежа, визуализирани според избрания тип на изчертаване.

**Предписания за Use case “Визуализиране на номерата на невроните”.**

1. Предпоставки.

Потребителят желае да номерата на невроните на дадена изкуствена невронна мрежа да са видими на екрана.

2. Предусловия.

Потребителят трябва да влезе в системата с коректни хост, порт, потребителско име и парола за връзка на системата към базата от данни и да зареди желаната невронна мрежа.

3. Стартиране на Use case.

Системата трябва да е в изправност. Не трябва да съществуват събития, които да възпрепятстват влизането на потребителя в нея (спиране на тока, прекъсната връзка с Интернет, хардуерен проблем с компютъра на потребителя).

4. Диалог.

- Потребителят активира от меню изписването на номерата на невроните на екрана.
- Системата изчертава наново невронната мрежа, като за всеки неврон визуализира съответния му номер.

5. Прекратяване на Use case.

Потребителят излиза от системата или натиска друго меню.

6. След-условия.

Потребителят е получил информация за номерацията на невроните в изкуствената невронна мрежа.

**Предписания за Use case “Задаване на цветове на типове неврони”.**

1. Предпоставки.

Потребителят желае да зададе различни цветове, в които да се изчертават на екрана различните типове неврони.

2. Предусловия.

Потребителят трябва да влезе в системата с коректно потребителско име и парола.

3. Стартиране на Use case.

Системата трябва да е в изправност. Не трябва да съществуват събития, които да възпрепятстват влизането на потребителя в нея (спиране на тока, прекъсната връзка с Интернет, хардуерен проблем с компютъра на потребителя).

4. Диалог.

- Потребителят въвежда коректни хост, порт, потребителско име и парола за връзка на системата към базата от данни.
- Системата инициализира начален екран за визуализиране на изкуствени невронни мрежи.
- Потребителят избира меню за промяна на настройки на системата.
- Системата визуализира екран за промяна на настройки, съдържащ текущите настройки на системата, както и бутони за запаметяване и затваряне на екрана.
- Потребителят избира желаните цветове, в които да се изчертават на екрана различните типове неврони и натиска бутон за запаметяване.
- Системата запаметява промяната в настройката за цветове на типове неврони.

5. Прекратяване на Use case.

Потребителят избира друго меню или излиза от системата .

6. След-условия.

Промените в настройките за задаване на цветове на различни типове неврони са запаметени в системата. При изчертаване на мрежа, невроните й се изчертават в новоизбраните цветове.

**Предписания за Use case “Задаване на размер на радиуса на невроните”.**

1. Предпоставки.

Потребителят желае да увеличи или намали радиуса на невроните при изчертаването им на екрана.

2. Предусловия.

Потребителят трябва да влезе в системата с коректно потребителско име и парола.

3. Стартиране на Use case.

Системата трябва да е в изправност. Не трябва да съществуват събития, които да възпрепятстват влизането на потребителя в нея (спиране на тока, прекъсната връзка с Интернет, хардуерен проблем с компютъра на потребителя).

4. Диалог.

- Потребителят въвежда коректни хост, порт, потребителско име и парола за връзка на системата към базата от данни.
- Системата инициализира начален екран за визуализиране на изкуствени невронни мрежи.
- Потребителят избира меню за промяна на настройки на системата.
- Системата визуализира екран за промяна на настройки, съдържащ текущите настройки на системата, както и бутони за запаметяване и затваряне на екрана.
- Потребителят избира желаната големина на радиуса на неврона и натиска бутон за запаметяване.
- Системата запаметява промяната в настройката за големина на радиус на неврони.

5. Прекратяване на Use case.

Потребителят избира друго меню или излиза от системата .

6. След-условия.

Промените в настройките за големината на радиус на неврони са запаметени в системата. При изчертаване на мрежа, невроните се изчертават с обновената големина на радиуса.

**Предписания за Use case “Задаване на размер на номерата на невроните”.**

1. Предпоставки.

Потребителят желае да увеличи или намали големината на номерата на невроните при изчертаването им на екрана.

2. Предусловия.

Потребителят трябва да влезе в системата с коректно потребителско име и парола.

3. Стартиране на Use case.

Системата трябва да е в изправност. Не трябва да съществуват събития, които да възпрепятстват влизането на потребителя в нея (спиране на тока, прекъсната връзка с Интернет, хардуерен проблем с компютъра на потребителя).

4. Диалог.

- Потребителят въвежда коректни хост, порт, потребителско име и парола за връзка на системата към базата от данни.
- Системата инициализира начален екран за визуализиране на изкуствени невронни мрежи.
- Потребителят избира меню за промяна на настройки на системата.
- Системата визуализира екран за промяна на настройки, съдържащ текущите настройки на системата, както и бутони за запаметяване и затваряне на екрана.
- Потребителят избира желаната големина на номерата на невроните и натиска бутон за запаметяване.
- Системата запаметява промяната в настройката за големина на номерата на невроните.

5. Прекратяване на Use case.

Потребителят избира друго меню или излиза от системата .

6. След-условия.

Промените в настройките за големината на номера на неврони са запаметени в системата. При изчертаване на мрежа и избрана опция за видими номера, номерата на невроните са с обновена големина.

**Предписания за Use case “Задаване на фонов цвят на работното поле”.**

1. Предпоставки.

Потребителят желае фона на работното поле да притежава конкретен цвят.

2. Предусловия.

Потребителят трябва да влезе в системата с коректно потребителско име и парола.

3. Стартиране на Use case.

Системата трябва да е в изправност. Не трябва да съществуват събития, които да възпрепятстват влизането на потребителя в нея (спиране на тока, прекъсната връзка с Интернет, хардуерен проблем с компютъра на потребителя).

4. Диалог.

- Потребителят въвежда коректни хост, порт, потребителско име и парола за връзка на системата към базата от данни.

- Системата инициализира начален екран за визуализиране на изкуствени невронни мрежи.
- Потребителят избира меню за промяна на настройки на системата.
- Системата визуализира екран за промяна на настройки, съдържащ текущите настройки на системата, както и бутони за запаметяване и затваряне на екрана.
- Потребителят избира желания цвят за фон на работното поле и натиска бутон за запаметяване.
- Системата запаметява промяната в настройката за цвят на фона на работното поле.

5. Прекратяване на Use case.

Потребителят избира друго меню или излиза от системата .

6. След-условия.

Промените в настройките за цвят на фона на работното поле са запаметени в системата. Работното поле е с фон в новоизбрания цвят.

**Предписания за Use case “Идентификация на потребителя”.**

1. Предпоставки.

Потребителят желае да влезе в системата за графично изобразяване и визуално манипулиране на изкуствени невронни мрежи.

2. Предусловия.

Потребителят трябва да избере меню за вход в системата.

3. Стартиране на Use case.

Системата трябва да е в изправност. Не трябва да съществуват събития, които да възпрепятстват влизането на потребителя в нея (спиране на тока, прекъсната връзка с Интернет, хардуерен проблем с компютъра на потребителя).

4. Диалог.

- Системата визуализира екран за вход.
- Потребителят въвежда хост и порт за връзка с базата от данни, както и потребителско име и парола.

- Системата валидира въведените хост, порт, потребителско име и парола. При успешно удостоверение за самоличност системата инициализира начален екран за визуализиране на изкуствени невронни мрежи.

5. Прекратяване на Use case.

Потребителят избира друго меню или излиза от системата .

6. След-условия.

Промените в настройките за цвят на фона на работното поле са запаметени в системата. Работното поле е с фон в новоизбрания цвят.

# Приложение 2 Основни успешни сценарии към Use case модела на информационната система

**Сценарий на Use case “Визуализиране на изкуствена невронна мрежа”.**

1. Use case започва, когато потребителят влезе в прозореца за вход в системата.
2. Системата подканва потребителя да въведе хост, порт, потребителско име и парола.
3. Потребителят въвежда поисканата информация. Ако хостът, портът, потребителското име и/или паролата са некоректни – стартира се Поток на грешка E1.
4. Системата инициализира начален екран за визуализиране на изкуствени невронни мрежи, съдържащ работно поле за изчертаване, падащи менюта за избор на валутна двойка, времеви период и идентификационен номер на наличните изкуствени невронни мрежи, както и бутони за зареждане на информацията в падащите менюта, запаметяване на промени в избрана невронна мрежа и изчертаване на топологията на мрежата в работното поле. Ако потребителят избере друго меню, стартира се алтернативен поток A1.
5. Потребителят натиска бутон за зареждане в падащите менюта на наличните в системата валутни двойки, времеви периоди и идентификационни номера на невронни мрежи.
6. Системата обновява стойностите в падащите менюта.
7. Потребителят избира идентификационен номер на мрежа и натиска бутон за зареждането й на екрана. Ако не е избран идентификационен номер, стартира се поток на грешка E2.
8. Системата визуализира топологията на избраната невронна мрежа.
9. Ако потребителят желае да визуализира друга мрежа, се преминава към т. 7.
10. Use case приключва.

Алтернативен поток A1.

1. Инициализира се избраният от потребителя екран.
2. Use case приключва.

Поток на грешка E1.

1. Системата уведомява потребителя, че въведените идентификационни данни са невалидни.

2. Use case приключва.

Поток на грешка E2.

1. Системата уведомява потребителя, че е необходимо да избере идентификационен номер на невронна мрежа.

2. Use case приключва.

**Сценарий на Use case “Търсене на изкуствена невронна мрежа”.**

1. Use case започва, когато потребителят влезе в прозореца за вход в системата.

2. Системата подканва потребителя да въведе хост, порт, потребителско име и парола.

3. Потребителят въвежда поисканата информация. Ако хостът, портът, потребителското име и/или паролата са некоректни – стартира се Поток на грешка E1.

4. Системата инициализира начален екран за визуализиране на изкуствени невронни мрежи, съдържащ работно поле за изчертаване, падащи менюта за избор на валутна двойка, времеви период и идентификационен номер на наличните изкуствени невронни мрежи, както и бутони за зареждане на информацията в падащите менюта, запаметяване на промени в избрана невронна мрежа и изчертаване на топологията на мрежата в работното поле. Ако потребителят избере друго меню, стартира се алтернативен поток A1.

5. Потребителят натиска бутон за зареждане в падащите менюта на наличните в системата валутни двойки, времеви периоди и идентификационни номера на невронни мрежи.

6. Системата обновява стойностите в падащите менюта.

7. Потребителят избира валутна двойка и/или времеви период, по които търси конкретна невронна мрежа.

8. Системата зарежда в падащото меню с идентификационни номера номерата на мрежите, притежаващи избраните от потребителя валутна двойка и/или времеви период.

9. Ако потребителят желае да потърси по валутна двойка и/или времеви период друга мрежа, се преминава към т. 7.

10. Use case приключва.

Алтернативен поток А1.

1. Инициализира се избраният от потребителя екран.

2. Use case приключва.

Поток на грешка Е1.

1. Системата уведомява потребителя, че въведените идентификационни данни са невалидни.

2. Use case приключва.

**Сценарий на Use case “Визуализиране на параметри на изкуствена невронна мрежа”.**

1. Use case започва, когато потребителят влезе в прозореца за вход в системата.

2. Системата подканва потребителя да въведе хост, порт, потребителско име и парола.

3. Потребителят въвежда поисканата информация. Ако хостът, портът, потребителското име и/или паролата са некоректни – стартира се Поток на грешка Е1.

4. Системата инициализира начален екран за визуализиране на изкуствени невронни мрежи.

5. Потребителят зарежда на екрана желаната невронна мрежа. Ако потребителят не избере невронна мрежа за изчертаване на екрана или избере друго меню, стартира се Алтернативен поток А1.

6. От вече заредена и изчертана на екрана изкуствена невронна мрежа потребителят избира неврон или връзка, чиито параметри желае да разгледа.

7. Системата визуализира стойностите на параметрите за избрания неврон или връзка.

8. Ако потребителят желае да разгледа параметри на друга невронна мрежа, преминава се към т. 5.

9. Use case приключва.

Алтернативен поток А1.

1. Инициализира се избраният от потребителя екран.

2. Use case приключва.

Поток на грешка Е1.

1. Системата уведомява потребителя, че въведените идентификационни данни са невалидни.

2. Use case приключва.

**Сценарий на Use case “Промяна на параметри на изкуствена невронна мрежа”.**

1. Use case започва, когато потребителят влезе в прозореца за вход в системата.

2. Системата подканва потребителя да въведе хост, порт, потребителско име и парола.

3. Потребителят въвежда поисканата информация. Ако хостът, портът, потребителското име и/или паролата са некоректни – стартира се Поток на грешка Е1.

4. Системата инициализира начален екран за визуализиране на изкуствени невронни мрежи.

5. Потребителят зарежда на екрана желаната невронна мрежа и избира неврон или връзка, чиито параметри би желал да промени. Ако потребителят не избере невронна мрежа за изчертаване на екрана, не избере неврон или връзка или избере друго меню, стартира се Алтернативен поток А1.

6. Потребителят прави промени в стойностите на параметрите.

7. Потребителят натиска бутон за запаметяване на промените по стойностите на параметрите. Ако потребителят не натисне бутон за запаметяване, се стартира Алтернативен поток А2.

8. Системата запаметява новите стойности.

9. Ако потребителят желае да направи промени по параметри на друга невронна мрежа, преминава се към т. 5.

10. Use case приключва.

Алтернативен поток А1.

1. Инициализира се избраният от потребителя екран.

2. Use case приключва.

Алтернативен поток А2.

1. Промените в стойностите на параметрите на избраната изкуствена невронна мрежа не биват запаметени в системата.

2. Use case приключва.

Поток на грешка Е1.

1. Системата уведомява потребителя, че въведените идентификационни данни са невалидни.

2. Use case приключва.

**Сценарий на Use case “Изчертаване на връзките в различни цветове според параметрите им ”.**

1. Use case започва, когато потребителят влезе в прозореца за вход в системата.

2. Системата подканва потребителя да въведе хост, порт, потребителско име и парола.

3. Потребителят въвежда поисканата информация. Ако хостът, портът, потребителското име и/или паролата са некоректни – стартира се Поток на грешка Е1.

4. Системата инициализира начален екран за визуализиране на изкуствени невронни мрежи.

5. Потребителят зарежда на екрана желаната невронна мрежа. Ако потребителят не избере невронна мрежа за изчертаване на екрана и избере друго меню, стартира се Алтернативен поток А1.

6. Потребителят избира от меню начина на задаване на цветове на връзките – само според активностите им, само според теглата им, според активностите и теглата им или независимо от активностите и теглата им.

7. Системата изчертава наново невронната мрежа, като оцветява връзките в зависимост от избрания тип на изчертаването им.

8. Use case приключва.

Алтернативен поток А1.

1. Инициализира се избраният от потребителя екран.

2. Use case приключва.

Поток на грешка Е1.

1. Системата уведомява потребителя, че въведените идентификационни данни са невалидни.

2. Use case приключва.

**Сценарий на Use case “Визуализиране на номерата на невроните”.**

1. Use case започва, когато потребителят влезе в прозореца за вход в системата.

2. Системата подканва потребителя да въведе хост, порт, потребителско име и парола.

3. Потребителят въвежда поисканата информация. Ако хостът, портът, потребителското име и/или паролата са некоректни – стартира се Поток на грешка Е1.

4. Системата инициализира начален екран за визуализиране на невронни мрежи.

5. Потребителят зарежда на екрана желаната невронна мрежа. Ако потребителят не избере невронна мрежа за изчертаване на екрана и избере друго меню, стартира се Алтернативен поток А1.

6. Потребителят активира от меню изписването на номерата на невроните на екрана.

7. Системата изчертава наново невронната мрежа, като за всеки неврон визуализира съответния му номер

8. Use case приключва.

Алтернативен поток А1.

1. Инициализира се избраният от потребителя екран.

2. Use case приключва.

Поток на грешка Е1.

1. Системата уведомява потребителя, че въведените идентификационни данни са невалидни.

2. Use case приключва.

**Сценарий на Use case “Задаване на цветове на типове неврони”.**

1. Use case започва, когато потребителят влезе в прозореца за вход в системата.
2. Системата подканва потребителя да въведе хост, порт, потребителско име и парола.
3. Потребителят въвежда поисканата информация. Ако хостът, портът, потребителското име и/или паролата са некоректни – стартира се Поток на грешка E1.
4. Системата инициализира начален екран за визуализиране на изкуствени невронни мрежи.
5. Потребителят избира меню за промяна на настройки на системата. Ако потребителят избере друго меню, стартира се алтернативен поток A1.
6. Системата визуализира екран за промяна на настройки, съдържащ текущите настройки на системата, както и бутони за запаметяване и затваряне на екрана.
7. Потребителят избира желаните цветове, в които да се изчертават на екрана различните типове неврони и натиска бутон за запаметяване. Ако потребителят натисне бутон за затваряне на екрана, се стартира алтернативен поток A2.
8. Системата запаметява промяната в настройката за цветове на типове неврони.
9. Use case приключва.

Алтернативен поток A1.

1. Инициализира се избраният от потребителя екран.
2. Use case приключва.

Алтернативен поток A2.

1. Инициализира се екран за визуализиране на изкуствени невронни мрежи.
2. Use case приключва.

Поток на грешка E1.

1. Системата уведомява потребителя, че въведените идентификационни данни са невалидни.
2. Use case приключва.

**Сценарий на Use case “Задаване на размер на радиуса на невроните”.**

1. Use case започва, когато потребителят влезе в прозореца за вход в системата.
2. Системата подканва потребителя да въведе хост, порт, потребителско име и парола.
3. Потребителят въвежда поисканата информация. Ако хостът, портът, потребителското име и/или паролата са некоректни – стартира се Поток на грешка E1.
4. Системата инициализира начален екран за визуализиране на изкуствени невронни мрежи.
5. Потребителят избира меню за промяна на настройки на системата. Ако потребителят избере друго меню, стартира се алтернативен поток A1.
6. Системата визуализира екран за промяна на настройки, съдържащ текущите настройки на системата, както и бутони за запаметяване и затваряне на екрана.
7. Потребителят избира желаната големина на радиуса на невроните натиска бутон за запаметяване. Ако потребителят натисне бутон за затваряне на екрана, се стартира алтернативен поток A2.
8. Системата запаметява промяната в настройката за големина на радиус на неврони.
9. Use case приключва.

Алтернативен поток A1.

1. Инициализира се избраният от потребителя екран.
2. Use case приключва.

Алтернативен поток A2.

1. Инициализира се екран за визуализиране на изкуствени невронни мрежи.
2. Use case приключва.

Поток на грешка E1.

1. Системата уведомява потребителя, че въведените идентификационни данни са невалидни.
2. Use case приключва.

**Сценарий на Use case “Задаване на размер на номерата на невроните”.**

1. Use case започва, когато потребителят влезе в прозореца за вход в системата.
2. Системата подканва потребителя да въведе хост, порт, потребителско име и парола.
3. Потребителят въвежда поисканата информация. Ако хостът, портът, потребителското име и/или паролата са некоректни – стартира се Поток на грешка E1.
4. Системата инициализира начален екран за визуализиране на изкуствени невронни мрежи.
5. Потребителят избира меню за промяна на настройки на системата. Ако потребителят избере друго меню, стартира се алтернативен поток A1.
6. Системата визуализира екран за промяна на настройки, съдържащ текущите настройки на системата, както и бутони за запаметяване и затваряне на екрана.
7. Потребителят избира желаната големина на номерата на невроните и натиска бутон за запаметяване. Ако потребителят натисне бутон за затваряне на екрана, се стартира алтернативен поток A2.
8. Системата запаметява промяната в настройката за големина на номерата на невроните.
9. Use case приключва.

Алтернативен поток A1.

1. Инициализира се избраният от потребителя екран.
2. Use case приключва.

Алтернативен поток A2.

1. Инициализира се екран за визуализиране на изкуствени невронни мрежи.
2. Use case приключва.

Поток на грешка E1.

1. Системата уведомява потребителя, че въведените идентификационни данни са невалидни.
2. Use case приключва.

**Сценарий на Use case “Задаване на фонов цвят на работното поле”.**

1. Use case започва, когато потребителят влезе в прозореца за вход в системата.
2. Системата подканва потребителя да въведе хост, порт, потребителско име и парола.
3. Потребителят въвежда поисканата информация. Ако хостът, портът, потребителското име и/или паролата са некоректни – стартира се Поток на грешка E1.
4. Системата инициализира начален екран за визуализиране на изкуствени невронни мрежи.
5. Потребителят избира меню за промяна на настройки на системата. Ако потребителят избере друго меню, стартира се алтернативен поток A1.
6. Системата визуализира екран за промяна на настройки, съдържащ текущите настройки на системата, както и бутони за запаметяване и затваряне на екрана.
7. Потребителят избира желания цвят за фон на работното поле и натиска бутон за запаметяване. Ако потребителят натисне бутон за затваряне на екрана, се стартира алтернативен поток A2.
8. Системата запаметява промяната в настройката за задаване на фонов цвят на работното поле.
9. Use case приключва.

Алтернативен поток A1.

1. Инициализира се избраният от потребителя екран.
2. Use case приключва.

Алтернативен поток A2.

1. Инициализира се екран за визуализиране на изкуствени невронни мрежи.
2. Use case приключва.

Поток на грешка E1.

1. Системата уведомява потребителя, че въведените идентификационни данни са невалидни.
2. Use case приключва.

**Сценарий на Use case “Идентификация на потребителя”.**

1. Use case започва, когато потребителят влезе в прозореца за вход в системата.

2. Системата подканва потребителя да въведе хост, порт, потребителско име и парола.

3. Потребителят въвежда поисканата информация.

4. Системата валидира въведените идентификационни данни. Ако хостът, портът, потребителското име и/или паролата са некоректни – стартира се Поток на грешка E1.

4. Системата инициализира начален екран за визуализиране на изкуствени невронни мрежи.

5. Use case приключва.

Поток на грешка E1.

1. Системата уведомява потребителя, че въведените идентификационни данни са невалидни.

2. Use case приключва.

# Приложение 3 Наръчник на системния администратор

В настоящия наръчник на системния администратор е разгледана процедурата по инсталиране и конфигуриране на сървър, предназначен да обслужва системата за графично изобразяване и визуално манипулиране на изкуствени невронни мрежи. Приема се, че администраторът разполага с физическа машина без инсталирана операционна система и софтуер.

С цел максимално редуциране на разходите за внедряване на продукта за визуално представяне и манипулиране на изкуствени невронни мрежи, се препоръчва използването на безплатни софтуерни продукти и технологии.

**Лиценз**

Софтуерът е безплатен, с отворен код и се разпространява под лиценза GPLv3. Пълният текст, съдържащ правилата и условията за ползване на софтуер, разпространяван под лиценз GPLv3, е достъпен на адрес: http://www.gnu.org/licenses/gpl-3.0.txt.

**Инсталиране на операционна система**

Подходящи операционни системи за инсталиране на приложението са дистрибуциите на Linux и Unix.

**Инсталиране на уеб сървър**

Препоръчително е да бъде избран най-разпространеният уеб сървър с отворен код, Apache HTTP Server.

Можете да изберете и свалите версия, подходяща за избраната от Вас операционан система, от уеб сайта на Apache HTTP Server Project. Наличните версии могат да бъдат намерени на адрес: http://httpd.apache.org/download.cgi.

В процеса на инсталация на уеб сървъра би могъл да се следва потребителският наръчник за инсталиране, наличен на адрес: http://httpd.apache.org/docs/2.0/install.html.

**Инсталиране на сървър за бази от данни**

Тъй като базата от данни, необходима за правилното функциониране на приложението, е MySQL, нужно е да бъде инсталиран сървър за бази от данни MySQL Community Server.

Можете да изберете и свалите версия, подходяща за избраната от Вас операционан система, от уеб сайта на MySQL. Наличните версии могат да бъдат намерени на адрес: http://dev.mysql.com/downloads/mysql/.

По време на инсталация на сървъра за бази от данни можете да използвате потребителски наръчник за инсталиране, наличен на адрес: http://dev.mysql.com/doc/refman/5.1/en/installing.html.

**Създаване на база от данни и задаване на права**

Необходимо е да се създаде база от данни с име todorb_vitoshatrade и колация utf8-unicode-ci. Тя ще съдържа информацията, с която работи системата за визуализиране и манипулиране на изкуствени невронни мрежи. Задължително е да се зададат конкретни права за потребителите на системата, които да важат единствено за създадената база от данни.

Създайте нов потребител с парола, като зададете привилегии единствено над данните, а именно възможност за извършване на операции “SELECT”, “INSERT”, “UPDATE”, “DELETE”. Потребителят не трябва да има други привилегии поради съображения за сигурност.

Използвайки създадените потребителско име и парола, потребителите на приложението за графично изобразяване и манипулиране на изкуствени невронни мрежи ще могат да влезнат в него.

След създаването на базата от данни и задаването на права, изпълнете SQL заявките за създаване и попълване на таблиците към нея, намиращи се във файла todorb_vitoshatrade.sql. Файлът може да бъде намерен в папката с име „sql”.

**Публикуване на системата в мрежата**

За да бъде системата за графично изобразяване и визуално манипулиране публично достъпна, е необходимо да копирате наличните файлове с разширения .php, .css, .jar, в директорията public_html на сървъра.

# Приложение 4 Програмен текст на информационната система

### DatabaseHelper.java

```
/**
 * Provides a list with available currency pairs.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 17 Oct 2011
 *
 * @return Array of strings with currency pairs.
 */
public String[][] loadCurrencyPairs() {
	String[][] result = null;
	try {
		Statement select = connection.createStatement();
		ResultSet resultSet = select.executeQuery(SQL_QUERY_SYMBOL);

		resultSet.last();
		int size = resultSet.getRow();
		resultSet.beforeFirst();
		if (size > 0) {
			result = new String[size][2];
			int i = 0;
			while (resultSet.next() == true) {
				result[i][0] = resultSet.getString("symbol");
				i++;
			}
		}
	} catch (Exception ex) {
		result = null;
		System.err.println(Texts.ERROR_LOAD_CURRENCY_PAIRS_LIST);
		ex.printStackTrace();
		// TODO Inform user that there is no valid currency pairs list.
	}
	return (result);
}

/**
 * Provides a list with available periods.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 17 Oct 2011
 *
 * @return 2D array of strings with periods.
 */
public String[][] loadPeriods() {
	String[][] result = null;
	try {
		Statement select = connection.createStatement();
```

```
                    ResultSet resultSet = select.executeQuery(SQL_QUERY_PERIOD);

                    resultSet.last();
                    int size = resultSet.getRow();
                    resultSet.beforeFirst();
                    if (size > 0) {
                          result = new String[size][2];
                          int i = 0;
                          while (resultSet.next() == true) {
                                result[i][0] = resultSet.getString("id");
                                result[i][1] = resultSet.getString("period");
                                i++;
                          }
                    }
             } catch (Exception ex) {
                    result = null;
                    System.err.println(Texts.ERROR_LOAD_PERIODS_LIST);
                    ex.printStackTrace();
                    // TODO Inform user that there is no valid periods list.
             }
             return (result);
       }


       /**
        * Saves a new ANN to the remote database server.
        *
        * @param annSymbol
        *            A currency pair chosen for the new ANN.
        *
        * @param annPeriod
        *            A time period chosen for the new ANN.
        *
        * @param annNumberNeurons
        *            Number of neurons of the new ANN.
        *
        * @param annActivitiesWeights
        *            Weights of the new ANN.
        *
        * @param annFitness
        *            Fitness of the new ANN.
        *
        * @param annFlags
        *            Flags (neuron types) of the new ANN's neurons.
        *
        * @author Ralitza Koleva
        *
        * @email rallly@abv.bg
        *
        * @date 20 Sep 2011
        */
       public int saveNewAnn(String annSymbol, int annPeriod,
                    int annNumberNeurons, String annActivitiesWeights, int
annFitness,
                    String annFlags) {
             String sql = "";

             /*
              * The ID of the currency pair of the new inserted ANN.
              */
             int currencyPairId = 0;

             /*
```

```
                 * The ID of the new inserted ANN kind.
                 */
                int annKindId = 0;

                /*
                 * The ID of the new inserted ANN.
                 */
                int annId = 0;

                try {

                        /*
                         * Selects the currency pair ID by symbol and period.
                         */
                        Statement insert = connection.createStatement();
                        sql = String.format(

        SQL_QUERY_CURRENCY_PAIR_BY_CURRENCY_PAIR_AND_PERIOD,
                                        annSymbol, annPeriod);
                        insert.execute(sql);
                        ResultSet selectedCurrencyPairId = insert.getResultSet();
                        if (selectedCurrencyPairId.next()) {

                                /*
                                 * Gets the automatically generated ID of the new ANN
kind.
                                 */
                                currencyPairId = selectedCurrencyPairId.getInt(1);
                        }

                        /*
                         * Inserts information about the new ANN kind.
                         */
                        sql = String.format(SQL_QUERY_INSERT_ANN_KIND, currencyPairId,
                                        annNumberNeurons, annFlags, annActivitiesWeights);
                        insert.executeUpdate(sql, Statement.RETURN_GENERATED_KEYS);
                        ResultSet generatedAnnKindId = insert.getGeneratedKeys();
                        if (generatedAnnKindId.next()) {

                                /*
                                 * Gets the automatically generated ID of the new ANN
kind.
                                 */
                                annKindId = generatedAnnKindId.getInt(1);
                        }

                        /*
                         * Inserts information about the new ANN.
                         */
                        sql = String.format(SQL_QUERY_INSERT_ANN, annKindId, annFitness,
                                        annActivitiesWeights);
                        insert.executeUpdate(sql, Statement.RETURN_GENERATED_KEYS);
                        ResultSet generatedAnnId = insert.getGeneratedKeys();
                        if (generatedAnnId.next()) {

                                /*
                                 * Gets the automatically generated ID of the new ANN.
                                 */
                                annId = generatedAnnId.getInt(1);
                        }
                } catch (Exception ex) {
                        System.err.println(Texts.ERROR_ADD_ANN);
                        ex.printStackTrace();
```

```
            }
            return annId;
      }

      /**
       * Deletes an ANN from the remote database server.
       *
       * @param id
       *            Unique identifier of the ANN.
       *
       * @author Ralitza Koleva
       *
       * @email rallly@abv.bg
       *
       * @date 27 Nov 2011
       */
      public void deleteAnn(int id) {

            /*
             * SQL string.
             */
            String sql = "";

            try {

                  /*
                   * Deletes information about the ANN kind.
                   */
                  Statement delete = connection.createStatement();
                  sql = String.format(SQL_QUERY_DELETE_ANN_KIND, id);
                  delete.executeUpdate(sql);

                  /*
                   * Deletes information about the ANN.
                   */
                  sql = String.format(SQL_QUERY_DELETE_ANN, id);
                  delete.executeUpdate(sql);
            } catch (Exception ex) {
                  System.err.println(Texts.ERROR_DELETE_ANN);
                  ex.printStackTrace();
            }
      }
```

## DeleteAnnPane.java

```
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JComboBox;

/**
 * Panel with GUI controls for Neural Network Management.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
```

```
 *
 * @date 27 Nov 2010
 */
public class DeleteAnnPane extends JPanel {

	/**
	 * Default serial version UID.
	 */
	private static final long serialVersionUID = 1L;

	/**
	 * Parent applet reference.
	 */
	private VitoshaTradeApplet parent = null;

	/**
	 * GUI control for obtaining and selecting network ID.
	 */
	private JComboBox networkId = new JComboBox();

	/**
	 * GUI control for deleting a selected network.
	 */
	private JButton delete = new JButton();

	/**
	 * GUI control for updating ANN server information.
	 */
	private JButton refresh = new JButton();

	/**
	 * Constructing delete ANN pane.
	 *
	 * @param parent
	 *            The parent class.
	 *
	 * @author Ralitza Koleva
	 *
	 * @email rallly@abv.bg
	 *
	 * @date 27 Nov 2011
	 */
	public DeleteAnnPane(final VitoshaTradeApplet parent) {
		this.parent = parent;
		this.setPreferredSize(new Dimension(VitoshaTradeApplet.EAST_PANE_WIDTH,
				VitoshaTradeApplet.EAST_PANE_HEIGHT));

		setLayout(new GridLayout(25, 1));

		add(new JLabel(Texts.LABEL_ANN_ID));
		add(networkId);
		networkId.setEditable(false);

		/*
		 * Acts as separator.
		 */
		add(new JLabel());

		add(delete);
		delete.setText(Texts.LABEL_BUTTON_DELETE);
		delete.addActionListener(new ActionListener() {

			/**
```

```
		 * Deletes all ANN information.
		 *
		 * @author Ralitza Koleva
		 *
		 * @email rallly@abv.bg
		 *
		 * @date 27 Nov 2011
		 */
		public void actionPerformed(ActionEvent event) {
			try {
				int ann_id = Integer.parseInt((String) networkId
						.getSelectedItem());
				parent.dbHelp.deleteAnn(ann_id);
				loadAllAnnIds();
				parent.workArea.repaint();
			} catch (Exception ex) {
				showInformationMessage();
			}
		}
	});

	/*
	 * Acts as separator.
	 */
	add(new JLabel());

	add(refresh);
	refresh.setText(Texts.LABEL_BUTTON_REFRESH);
	refresh.addActionListener(new ActionListener() {

		/**
		 * Refreshes ANN IDs in the ANN ID JComboBox.
		 *
		 * @author Ralitza Koleva
		 *
		 * @email rallly@abv.bg
		 *
		 * @date 27 Nov 2011
		 */
		public void actionPerformed(ActionEvent event) {
			loadAllAnnIds();
		}
	});
}

/**
 * Loads all ANN IDs in the ANN ID JComboBox.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 29 Oct 2011
 */
public void loadAllAnnIds() {
	String listIds[][] = parent.dbHelp.loadAnnList();
	networkId.removeAllItems();
	networkId.addItem("---");
	for (int i = 0; i < listIds.length; i++) {
		networkId.addItem(listIds[i][2]);
	}
}
```

```
	/**
	 * Shows a message if no ANN ID is selected to delete.
	 *
	 * @author Ralitza Koleva
	 *
	 * @email rallly@abv.bg
	 *
	 * @date 05 Dec 2011
	 */
	private void showInformationMessage(){
		InformationMessages error = new InformationMessages(
				Texts.INFORMATION_SELECT_ANN_ID,
				Texts.INFORMATION_SELECT_ANN_ID_TITLE,
				JOptionPane.INFORMATION_MESSAGE);
		error.showMessage();
	}
}
```

## DrawingPane.java

```
	/**
	 * Calculates neurons radius.
	 *
	 * @author Ralitza Koleva
	 *
	 * @email rallly@abv.bg
	 *
	 * @date 06 Dec 2011
	 */
	private int calculateNeuronRadius() {

		/**
		 * Default neuron radius.
		 */
		int neuronRadius = Math.min(VitoshaTradeApplet.CENTRAL_PANE_WIDTH,
				VitoshaTradeApplet.CENTRAL_PANE_HEIGHT) / 100;
		/**
		 * Default neuron radius size.
		 */
		String neuronRadiusSize = "Small";
		try {
			Properties p = new Properties();
			FileInputStream in = new FileInputStream(PROPERTIES_FILE_NAME);
			p.load(in);
			neuronRadiusSize = p.getProperty("NeuronsSize");
			if (neuronRadiusSize.equals("Small")) {
				neuronRadius =
Math.min(VitoshaTradeApplet.CENTRAL_PANE_WIDTH,
					VitoshaTradeApplet.CENTRAL_PANE_HEIGHT) /
100;
			} else if (neuronRadiusSize.equals("Medium")) {
				neuronRadius =
Math.min(VitoshaTradeApplet.CENTRAL_PANE_WIDTH,
					VitoshaTradeApplet.CENTRAL_PANE_HEIGHT) /
80;
			} else if (neuronRadiusSize.equals("Large")) {
				neuronRadius =
Math.min(VitoshaTradeApplet.CENTRAL_PANE_WIDTH,
					VitoshaTradeApplet.CENTRAL_PANE_HEIGHT) /
70;
			}
```

```
			} catch (Exception ex) {
				ex.printStackTrace();
			}
			return (neuronRadius);
		}

		/**
		 * Reads the current color of the neuron from a file, depending on its type
		 * (flag).
		 *
		 * @param colorProperty
		 *            The parameter name, containing the color.
		 *
		 * @author Ralitza Koleva
		 *
		 * @email rallly@abv.bg
		 *
		 * @date 26 Oct 2011
		 *
		 * @return The current color of the neuron.
		 */
		public Color readNeuronColor(String colorProperty) {
			Color neuronColor = new Color(0, 0, 0);
			try {
				Properties properties = new Properties();
				FileInputStream in = new FileInputStream(PROPERTIES_FILE_NAME);
				properties.load(in);
				int color =
Integer.parseInt(properties.getProperty(colorProperty));
				neuronColor = new Color(color);
			} catch (Exception ex) {
				ex.printStackTrace();
			}
			return (neuronColor);
		}

		/**
		 * Draws the connection in a specific color depending on the chosen radio
		 * button in menu Tools->Mesh.
		 *
		 * @param firstNeuronIndex
		 *            Index of the first neuron in the connection.
		 *
		 * @param secondNeuronIndex
		 *            Index of the second neuron in the connection.
		 *
		 * @param g
		 *            Graphic context.
		 *
		 * @author Ralitza Koleva
		 *
		 * @email rallly@abv.bg
		 *
		 * @date 31 Oct 2011
		 */
		public void selectConnectionsColorByChosenMesh(int firstNeuronIndex,
				int secondNeuronIndex, Graphics g) {

			/*
			 * Takes minimum and maximum activity, weight and both activity and
			 * weight.
			 */
			double minActivity = parent.ann.activities[0][0];
```

```
                double maxActivity = parent.ann.activities[0][0];
                double minWeight = parent.ann.weights[0][0];
                double maxWeight = parent.ann.weights[0][0];
                double minActivityWeight = parent.ann.activities[0][0]
                                * parent.ann.weights[0][0];
                double maxActivityWeight = parent.ann.activities[0][0]
                                * parent.ann.weights[0][0];
                for (int j = 0; j < parent.ann.numberOfNeurons; j++) {
                        for (int i = 0; i < parent.ann.numberOfNeurons; i++) {
                                if (parent.ann.activities[i][j] < minActivity)
                                        minActivity = parent.ann.activities[i][j];
                                if (parent.ann.activities[i][j] > maxActivity)
                                        maxActivity = parent.ann.activities[i][j];
                                if (parent.ann.weights[i][j] < minWeight)
                                        minWeight = parent.ann.weights[i][j];
                                if (parent.ann.weights[i][j] > maxWeight)
                                        maxWeight = parent.ann.weights[i][j];
                                if (parent.ann.activities[i][j] * parent.ann.weights[i][j]
< minActivityWeight)
                                        minActivityWeight = parent.ann.activities[i][j]
                                                        * parent.ann.weights[i][j];
                                if (parent.ann.activities[i][j] * parent.ann.weights[i][j]
> maxActivityWeight)
                                        maxActivityWeight = parent.ann.activities[i][j]
                                                        * parent.ann.weights[i][j];
                        }
                }

                /*
                 * If option "Solid" is chosen in "Mesh" menu - all connections are in
                 * black color.
                 */
                if (parent.meshSolidItem.isSelected() == true) {
                        g.drawLine(parent.ann.coordinates[firstNeuronIndex][0],
                                        parent.ann.coordinates[firstNeuronIndex][1],
                                        parent.ann.coordinates[secondNeuronIndex][0],
                                        parent.ann.coordinates[secondNeuronIndex][1]);

                        /*
                         * If option "Activities" is chosen in "Mesh" menu - connections
                         * colors depend on their activities.
                         */
                } else if (parent.meshActivitiesItem.isSelected() == true) {
                        connectionColorTemp =
parent.ann.activities[firstNeuronIndex][secondNeuronIndex]
                                        - minActivity;
                        connectionColorTemp = 255 - connectionColorTemp
                                        * (255 / (maxActivity - minActivity));
                        int color = (int) Math.round(connectionColorTemp);
                        g.setColor(new Color(color, color, color));
                        g.drawLine(parent.ann.coordinates[firstNeuronIndex][0],
                                        parent.ann.coordinates[firstNeuronIndex][1],
                                        parent.ann.coordinates[secondNeuronIndex][0],
                                        parent.ann.coordinates[secondNeuronIndex][1]);

                        /*
                         * If option "Weights" is chosen in "Mesh" menu - connections
colors
                         * depend on their weights.
                         */
                } else if (parent.meshWeightsItem.isSelected() == true) {
                        connectionColorTemp =
parent.ann.weights[firstNeuronIndex][secondNeuronIndex]
```

```
                                - minWeight;
                        connectionColorTemp = 255 - connectionColorTemp
                                * (255 / (maxWeight - minWeight));
                        int color = (int) Math.round(connectionColorTemp);
                        g.setColor(new Color(color, color, color));
                        g.drawLine(parent.ann.coordinates[firstNeuronIndex][0],
                                parent.ann.coordinates[firstNeuronIndex][1],
                                parent.ann.coordinates[secondNeuronIndex][0],
                                parent.ann.coordinates[secondNeuronIndex][1]);

                        /*
                         * If option "Both" is chosen in "Mesh" menu - connections
colors
                         * depend on their activities and weights.
                         */
                } else if (parent.meshBothItem.isSelected() == true) {
                        connectionColorTemp =
parent.ann.activities[firstNeuronIndex][secondNeuronIndex]
                                *
parent.ann.weights[firstNeuronIndex][secondNeuronIndex]
                                - minActivityWeight;
                        connectionColorTemp = 255 - connectionColorTemp
                                * (255 / (maxActivityWeight - minActivityWeight));
                        int color = (int) Math.round(connectionColorTemp);
                        g.setColor(new Color(color, color, color));
                        g.drawLine(parent.ann.coordinates[firstNeuronIndex][0],
                                parent.ann.coordinates[firstNeuronIndex][1],
                                parent.ann.coordinates[secondNeuronIndex][0],
                                parent.ann.coordinates[secondNeuronIndex][1]);
                }
        }

        /**
         * Shows the IDs of the neurons on the screen.
         *
         * @param g
         *            Graphic context.
         *
         * @param neuronId
         *            Neuron ID shown on the screen.
         *
         * @param neuronColor
         *            The color of the neuron.
         *
         * @param neuronIdXCoordinate
         *            X-coordinate of the neuron ID.
         *
         * @param neuronIdYCoordinate
         *            Y-coordinate of the neuron ID.
         *
         * @author Ralitza Koleva
         *
         * @email rallly@abv.bg
         *
         * @date 13 Nov 2011
         */
        private void showNeuronNumbers(Graphics g, int neuronId, Color neuronColor,
                        int neuronIdXCoordinate, int neuronIdYCoordinate) {
                if (parent.numberingItem.isSelected()) {

                        /*
                         * Converts the neuron color to its opposite, so that the neuron
ID
```

```
                     * stays visible inside the neuron.
                     */
                    int convertedNeuronRedPrimaryColor = (neuronColor.getRed() ^
0x80) & 0xff;
                    int convertedNeuronGreenPrimaryColor = (neuronColor.getGreen() ^
0x80) & 0xff;
                    int convertedNeuronBluePrimaryColor = (neuronColor.getBlue() ^
0x80) & 0xff;

                    /*
                     * Reads the numbers size from a file. If the process fails, a
                     * default size is taken.
                     */
                    int neuronIdFontSize = 9;
                    try {
                          Properties properties = new Properties();
                          FileInputStream in = new
FileInputStream(PROPERTIES_FILE_NAME);
                          properties.load(in);
                          int currentNeuronsNumberSize = Integer.parseInt(properties
                                      .getProperty("NeuronsNumbersSize"));
                          neuronIdFontSize = currentNeuronsNumberSize;
                    } catch (Exception ex) {
                          ex.printStackTrace();
                    }

                    /*
                     * Creates a new font for the neuron IDs.
                     */
                    String neuronIdFontFamily = "Verdana";
                    Font neuronIdFont = new Font(neuronIdFontFamily, Font.BOLD,
                                neuronIdFontSize);
                    Color neuronIdColor = new Color(convertedNeuronRedPrimaryColor,
                                convertedNeuronGreenPrimaryColor,
                                convertedNeuronBluePrimaryColor);

                    /*
                     * Sets font and color of the neuron ID and draws it inside the
                     * neuron oval.
                     */
                    g.setColor(neuronIdColor);
                    g.setFont(neuronIdFont);
                    g.drawString(Integer.toString(neuronId), neuronIdXCoordinate,
                                neuronIdYCoordinate);
              }
        }

        /**
         * Reads the work area background color.
         *
         * @param g
         *            Graphic context.
         *
         * @author Ralitza Koleva
         *
         * @email rallly@abv.bg
         *
         * @date 23 Nov 2011
         */
        private void selectBackgroundColor(Graphics g) {
              try {
                    Properties properties = new Properties();
                    FileInputStream in = new FileInputStream(PROPERTIES_FILE_NAME);
```

```
                    properties.load(in);
                    int color = Integer.parseInt(properties

        .getProperty(WORK_AREA_BACKGROUND_COLOR_PROPERTY_KEY));
                    Color workAreaBackgroundColor = new Color(color);
                    g.setColor(workAreaBackgroundColor);
              } catch (Exception ex) {
                    ex.printStackTrace();
              }
        }
```

## InformationMessages.java

```
import java.awt.Component;
import javax.swing.JOptionPane;

/**
 * Shows a pop-up message on the screen.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 05 Dec 2011
 */
public class InformationMessages {

        /**
         * Frame for the connection error message.
         */
        protected static final Component frame = null;

        /**
         * Message text.
         */
        private String message;

        /**
         * Message title.
         */
        private String title;

        /**
         * Message type.
         */
        private int messageType;

        /**
         * Constructing pop-up message.
         *
         * @param message
         *            Message text.
         *
         * @param header
         *            Message header.
         *
         * @param messageType
         *            Message type (error/warning/information, etc.).
         *
         * @author Ralitza Koleva
         *
         * @email rallly@abv.bg
```

```
         *
         * @date 05 Dec 2011
         */
        public InformationMessages(String message, String header, int messageType) {
               this.message = message;
               this.title = header;
               this.messageType = messageType;
        }

        /**
         * Shows a pop-up message on the screen.
         *
         * @author Ralitza Koleva
         *
         * @email rallly@abv.bg
         *
         * @date 05 Dec 2011
         */
        void showMessage() {
               JOptionPane.showMessageDialog(frame, message, title, messageType);
               return;
        }
}
```

## LoginPane.java

```
        /**
         * Shows error message when login fails.
         *
         * @author Ralitza Koleva
         *
         * @email rallly@abv.bg
         *
         * @date 24 Nov 2011
         */
        private void showConnectionError() {
               InformationMessages error = new InformationMessages(
                             Texts.ERROR_DATABASE_CONNECT,
                             Texts.ERROR_DATABASE_CONNECT_TITLE,
JOptionPane.ERROR_MESSAGE);
               error.showMessage();
               parent.showLoginPane();
        }
```

## NetworkPane.java

```
        /**
         * Shows a message if no ANN ID is selected to load/save.
         *
         * @author Ralitza Koleva
         *
         * @email rallly@abv.bg
         *
         * @date 05 Dec 2011
         */
        private void showInformationMessage() {
               InformationMessages error = new InformationMessages(
                             Texts.INFORMATION_SELECT_ANN_ID,
                             Texts.INFORMATION_SELECT_ANN_ID_TITLE,
                             JOptionPane.INFORMATION_MESSAGE);
               error.showMessage();
        }
```

```
}
```

## NewAnnPane.java

```
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;

import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

/**
 * Panel with GUI controls for a new ANN management.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 18 Sep 2011
 */
public class NewAnnPane extends JPanel {

	/**
	 * Default serial version UID.
	 */
	private static final long serialVersionUID = 1L;

	/**
	 * Parent applet reference.
	 */
	private VitoshaTradeApplet parent = null;

	/**
	 * GUI control for obtaining the ANN kind (shows symbols).
	 */
	private JComboBox networkSymbol = new JComboBox();

	/**
	 * GUI control for obtaining the ANN kind (shows period).
	 */
	private JComboBox networkPeriod = new JComboBox();

	/**
	 * GUI control for number of neurons.
	 */
	private JTextField networkNumberNeurons = new JTextField();

	/**
	 * GUI control for updating ANN server information.
	 */
	private JButton refresh = new JButton();

	/**
	 * GUI control for saving a new network.
	 */
	private JButton save = new JButton();
```

```
	/**
	 * Constructing new ANN pane.
	 *
	 * @param parent
	 *            The parent class.
	 *
	 * @author Ralitza Koleva
	 *
	 * @email rallly@abv.bg
	 *
	 * @date 18 Sep 2011
	 */
	public NewAnnPane(final VitoshaTradeApplet parent) {
		this.parent = parent;
		this.setPreferredSize(new Dimension(VitoshaTradeApplet.EAST_PANE_WIDTH,
				VitoshaTradeApplet.EAST_PANE_HEIGHT));

		setLayout(new GridLayout(25, 1));

		add(new JLabel(Texts.LABEL_SYMBOL));
		add(networkSymbol);
		networkSymbol.setEditable(false);

		add(new JLabel(Texts.LABEL_PERIOD));
		add(networkPeriod);
		networkPeriod.setEditable(false);

		add(new JLabel(Texts.LABEL_NUMBER_NEURONS));
		add(networkNumberNeurons);

		/*
		 * Deletes all tokens different from numbers. Only numbers are allowed
		 * in the field for inputting the number of neurons in the ANN.
		 */
		networkNumberNeurons.addKeyListener(new KeyAdapter() {
			/**
			 * Deletes all tokens different from numbers typed by the user.
			 *
			 * @author Ralitza Koleva
			 *
			 * @email rallly@abv.bg
			 *
			 * @date 17 Oct 2011
			 */
			public void keyTyped(KeyEvent event) {
				char c = event.getKeyChar();
				if ((c < '0') || (c > '9') || (c ==
KeyEvent.VK_BACK_SPACE)
					|| (c == KeyEvent.VK_DELETE)) {
					getToolkit().beep();
					event.consume();
				}
			}
		});

		/*
		 * Acts as separator.
		 */
		add(new JLabel());

		add(save);
		save.setText(Texts.LABEL_BUTTON_SAVE);
```

```
		save.addActionListener(new ActionListener() {

			/**
			 * Prepares a new ANN for saving and calls a function to save
it.
			 *
			 * @author Ralitza Koleva
			 *
			 * @email rallly@abv.bg
			 *
			 * @date 18 Sep 2011
			 */
			public void actionPerformed(ActionEvent event) {
				try{
				/*
				 * Gets the selected currency pair (symbol) ID of the new
ANN.
				 */
				String selectedSymbol =
networkSymbol.getSelectedItem().toString();

				/*
				 * Gets the selected period ID of the new ANN.
				 */
				int selectedPeriod = ((SymbolPeriodKeyValue) networkPeriod
						.getSelectedItem()).getKey();

				/*
				 * Gets the input number of neurons of the new ANN.
				 */
				String networkNumberNeuronsStr =
networkNumberNeurons.getText();
				int networkNumberNeurons = Integer
						.parseInt(networkNumberNeuronsStr);

				/*
				 * Prepares the default weights, fitness and flags of the
new
				 * ANN.
				 */
				String networkWeights = "";
				int networkFitness = 0;
				String networkFlags = "";
				for (int i = 0; i < networkNumberNeurons; i++) {
					networkFlags = networkFlags + "0" + " ";
					for (int j = 0; j < networkNumberNeurons; j++) {
						networkWeights = networkWeights + "1.0" + "
";
					}
				}

				/*
				 * Calls a function for saving the new ANN.
				 */
				int newAnn = parent.dbHelp.saveNewAnn(selectedSymbol,
						selectedPeriod, networkNumberNeurons,
networkWeights,
						networkFitness, networkFlags);

				/*
				 * Loads the new ANN in the work area.
				 */
				parent.ann = parent.dbHelp.loadAnn(newAnn);
```

```
                        parent.workArea.repaint();

                        /*
                         * Shows the screen for loading an ANN.
                         */
                        parent.showNetworkPane();
                        }catch (Exception ex) {
                                showInformationMessage();
                        }
                }
        });

        /*
         * Acts as separator.
         */
        add(new JLabel());

        add(refresh);
        refresh.setText(Texts.LABEL_BUTTON_REFRESH);
        refresh.addActionListener(new ActionListener() {

                /**
                 * Updates ANN server information in the symbol and period
                 * JComboBoxes.
                 *
                 * @author Ralitza Koleva
                 *
                 * @email rallly@abv.bg
                 *
                 * @date 18 Sep 2011
                 */
                public void actionPerformed(ActionEvent event) {
                        String listCurrencyPairs[][] = parent.dbHelp
                                        .loadCurrencyPairs();
                        if (listCurrencyPairs == null) {
                                return;
                        }

                        /*
                         * Updates symbols JComboBox items.
                         */
                        networkSymbol.removeAllItems();
                        for (int i = 0; i < listCurrencyPairs.length; i++) {
                                networkSymbol.addItem(listCurrencyPairs[i][0]);
                        }

                        String listPeriods[][] = parent.dbHelp.loadPeriods();
                        if (listPeriods == null) {
                                return;
                        }

                        /*
                         * Updates periods JComboBox items.
                         */
                        networkPeriod.removeAllItems();
                        for (int i = 0; i < listPeriods.length; i++) {
                                networkPeriod.addItem(new SymbolPeriodKeyValue(
                                                listPeriods[i][0],
listPeriods[i][1]));
                        }
                }
        });
    }
```

```
	/**
	 * Shows a message if no symbol/period is selected to save the new ANN.
	 *
	 * @author Ralitza Koleva
	 *
	 * @email rallly@abv.bg
	 *
	 * @date 05 Dec 2011
	 */
	private void showInformationMessage(){
		InformationMessages error = new InformationMessages(
				Texts.INFORMATION_SELECT_SYMBOL_AND_PERIOD,
				Texts.INFORMATION_SELECT_SYMBOL_AND_PERIOD_TITLE,
				JOptionPane.INFORMATION_MESSAGE);
		error.showMessage();
	}
}
```

## SettingsPane.java

```
import java.awt.Color;
import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.Properties;

import javax.swing.JButton;
import javax.swing.JColorChooser;
import javax.swing.JComboBox;
import javax.swing.JComponent;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SwingConstants;

/**
 * Panel with GUI controls for Neural Network Color Settings.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 21 Oct 2011
 */
public class SettingsPane extends JPanel {

	/**
	 * Class ActionListener for buttons choosing a neuron type color.
	 *
	 * @author Ralitza Koleva
	 *
	 * @email rallly@abv.bg
	 *
	 * @date 21 Oct 2011
	 */
	private class ChooseNeuronColorListener implements ActionListener {
```

```
		/**
		 * A field which shows the neuron type color.
		 */
		JTextField neuronTypeColorField;

		/**
		 * Constructor of the listener for choosing a neuron type color.
		 *
		 * @param neuronTypeColorField
		 *            Shows the chosen neuron type color.
		 *
		 * @author Ralitza Koleva
		 *
		 * @email rallly@abv.bg
		 *
		 * @date 21 Oct 2011
		 */
		public ChooseNeuronColorListener(JTextField neuronTypeColorField) {
			this.neuronTypeColorField = neuronTypeColorField;
		}

		/**
		 * Shows a color chooser to change a neuron type color and fills the
		 * relevant field in the chosen color.
		 *
		 * @author Ralitza Koleva
		 *
		 * @email rallly@abv.bg
		 *
		 * @date 21 Oct 2011
		 */
		public void actionPerformed(ActionEvent event) {
			Color initialBackground = neuronTypeColorField.getBackground();
			Color background = JColorChooser.showDialog(null,
					Texts.LABEL_BUTTON_CHOOSE_COLOR,
initialBackground);
			if (background != null) {
				neuronTypeColorField.setBackground(background);
			}
		}
	};

	/**
	 * Default serial version UID.
	 */
	private static final long serialVersionUID = 1L;

	/**
	 * Parent applet reference.
	 */
	private VitoshaTradeApplet parent = null;

	/**
	 * Label for Regular Neuron.
	 */
	private JLabel labelColorRegular = new JLabel(
			Texts.LABEL_REGULAR_NEURON_COLOR, SwingConstants.RIGHT);

	/**
	 * Button to open a color picker for choosing color for Regular Neuron.
	 */
	private JButton chooseColorRegular = new JButton(
```

```
                Texts.LABEL_BUTTON_CHOOSE_COLOR);

        /**
         * Field to show the chosen color for Regular Neuron.
         */
        private JTextField showColorRegular = new JTextField(10);

        /**
         * Label for Input Neuron.
         */
        private JLabel labelColorInput = new JLabel(Texts.LABEL_INPUT_NEURON_COLOR,
                SwingConstants.RIGHT);

        /**
         * Button to open a color picker for choosing color for Input Neuron.
         */
        private JButton chooseColorInput = new JButton(
                Texts.LABEL_BUTTON_CHOOSE_COLOR);

        /**
         * Field to show the chosen color for Input Neuron.
         */
        private JTextField showColorInput = new JTextField(10);

        /**
         * Label for Output Neuron.
         */
        private JLabel labelColorOutput = new JLabel(
                Texts.LABEL_OUTPUT_NEURON_COLOR, SwingConstants.RIGHT);

        /**
         * Button to open a color picker for choosing color for Output Neuron.
         */
        private JButton chooseColorOutput = new JButton(
                Texts.LABEL_BUTTON_CHOOSE_COLOR);

        /**
         * Field to show the chosen color for Output Neuron.
         */
        private JTextField showColorOutput = new JTextField(10);

        /**
         * Label for Bias Neuron.
         */
        private JLabel labelColorBias = new JLabel(Texts.LABEL_BIAS_NEURON_COLOR,
                SwingConstants.RIGHT);

        /**
         * Button to open a color picker for choosing color for Bias Neuron.
         */
        private JButton chooseColorBias = new JButton(
                Texts.LABEL_BUTTON_CHOOSE_COLOR);

        /**
         * Field to show the chosen color for Bias Neuron.
         */
        private JTextField showColorBias = new JTextField(10);

        /**
         * Label for Input-Output Neuron.
         */
        private JLabel labelColorInputOutput = new JLabel(
                Texts.LABEL_INPUT_OUTPUT_NEURON_COLOR, JLabel.RIGHT);
```

```
/**
 * Button to open a color picker for choosing color for Input-Output Neuron.
 */
private JButton chooseColorInputOutput = new JButton(
            Texts.LABEL_BUTTON_CHOOSE_COLOR);

/**
 * Field to show the chosen color for Input-Output Neuron.
 */
private JTextField showColorInputOutput = new JTextField(10);

/**
 * Label for Input-Bias Neuron.
 */
private JLabel labelColorInputBias = new JLabel(
            Texts.LABEL_INPUT_BIAS_NEURON_COLOR, JLabel.RIGHT);

/**
 * Button to open a color picker for choosing color for Input-Bias Neuron.
 */
private JButton chooseColorInputBias = new JButton(
            Texts.LABEL_BUTTON_CHOOSE_COLOR);

/**
 * Field to show the chosen color for Input-Bias Neuron.
 */
private JTextField showColorInputBias = new JTextField(10);

/**
 * Label for Output-Bias Neuron.
 */
private JLabel labelColorOutputBias = new JLabel(
            Texts.LABEL_OUTPUT_BIAS_NEURON_COLOR, JLabel.RIGHT);

/**
 * Button to open a color picker for choosing color for Output-Bias Neuron.
 */
private JButton chooseColorOutputBias = new JButton(
            Texts.LABEL_BUTTON_CHOOSE_COLOR);

/**
 * Field to show the chosen color for Output-Bias Neuron.
 */
private JTextField showColorOutputBias = new JTextField(10);

/**
 * Label for Input-Output-Bias Neuron.
 */
private JLabel labelColorInputOutputBias = new JLabel(
            Texts.LABEL_INPUT_OUTPUT_BIAS_NEURON_COLOR, JLabel.RIGHT);

/**
 * Button to open a color picker for choosing color for Input-Output-Bias
 * Neuron.
 */
private JButton chooseColorInputOutputBias = new JButton(
            Texts.LABEL_BUTTON_CHOOSE_COLOR);

/**
 * Field to show the chosen color for Input-Output-Bias Neuron.
 */
private JTextField showColorInputOutputBias = new JTextField(10);
```

```
/**
 * Drop down menu for choosing the size of neurons.
 */
private JComboBox chooseNeuronsSize = new JComboBox();

/**
 * Label for size of neurons.
 */
private JLabel labelNeuronsSize = new JLabel(Texts.LABEL_NEURONS_SIZE,
            JLabel.RIGHT);

/**
 * Label for size of neurons' numbers.
 */
private JLabel labelNeuronsNumbersSize = new JLabel(
            Texts.LABEL_NEURONS_NUMBERS_SIZE, JLabel.RIGHT);

/**
 * Drop down menu for choosing the size of neurons' numbers.
 */
private JComboBox chooseNeuronsNumbersSize = new JComboBox();

/**
 * Label for work area background color.
 */
private JLabel labelWorkAreaBackgroundColor = new JLabel(
            Texts.LABEL_WORK_AREA_BACKGROUND_COLOR, JLabel.RIGHT);

/**
 * Button to open a color picker for choosing color for Input-Bias Neuron.
 */
private JButton chooseWorkAreaBackgroundColor = new JButton(
            Texts.LABEL_BUTTON_CHOOSE_COLOR);

/**
 * Field to show the chosen color for Input-Bias Neuron.
 */
private JTextField showWorkAreaBackgroundColor = new JTextField(10);

/**
 * Button to save the new neuron colors.
 */
private JButton save = new JButton(Texts.LABEL_BUTTON_SAVE_CAPS);

/**
 * Button to close the setting pane and return to the network pane (load).
 */
private JButton close = new JButton(Texts.LABEL_BUTTON_CLOSE_CAPS);

/**
 * Labels width.
 */
private static final int LABELS_WIDTH = 200;

/**
 * Labels height.
 */
private static final int LABELS_HEIGHT = 20;

/**
 * Labels dimensions.
 */
```

```
	private static final Dimension LABELS_DIMENTIONS = new Dimension(
			LABELS_WIDTH, LABELS_HEIGHT);

	/**
	 * Buttons width.
	 */
	private static final int BUTTONS_WIDTH = 120;

	/**
	 * Buttons height.
	 */
	private static final int BUTTONS_HEIGHT = 25;

	/**
	 * Minimum neurons' numbers size.
	 */
	private static final int MIN_NEURONS_NUMBERS_SIZE = 8;

	/**
	 * Maximum neurons' numbers size.
	 */
	private static final int MAX_NEURONS_NUMBERS_SIZE = 14;

	/**
	 * Name of the property key which has as a value the color of regular
	 * neurons.
	 */
	private static final String REGULAR_NEURON_COLOR_PROPERTY_KEY =
"RegularNeuronColor";

	/**
	 * Name of the property key which has as a value the color of input neurons.
	 */
	private static final String INPUT_NEURON_COLOR_PROPERTY_KEY =
"InputNeuronColor";

	/**
	 * Name of the property key which has as a value the color of output
	 * neurons.
	 */
	private static final String OUTPUT_NEURON_COLOR_PROPERTY_KEY =
"OutputNeuronColor";

	/**
	 * Name of the property key which has as a value the color of output
	 * neurons.
	 */
	private static final String BIAS_NEURON_COLOR_PROPERTY_KEY =
"BiasNeuronColor";

	/**
	 * Name of the property key which has as a value the color of input-output
	 * neurons.
	 */
	private static final String INPUT_OUTPUT_NEURON_COLOR_PROPERTY_KEY =
"InputOutputNeuronColor";

	/**
	 * Name of the property key which has as a value the color of input-bias
	 * neurons.
	 */
	private static final String INPUT_BIAS_NEURON_COLOR_PROPERTY_KEY =
"InputBiasNeuronColor";
```

```
	/**
	 * Name of the property key which has as a value the color of output-bias
	 * neurons.
	 */
	private static final String OUTPUT_BIAS_NEURON_COLOR_PROPERTY_KEY =
"OutputBiasNeuronColor";

	/**
	 * Name of the property key which has as a value the color of
	 * input-output-bias neurons.
	 */
	private static final String INPUT_OUTPUT_BIAS_NEURON_COLOR_PROPERTY_KEY =
"InputOutputBiasNeuronColor";

	/**
	 * Name of the property key which has as a value the color of work area.
	 */
	private static final String WORK_AREA_BACKGROUND_COLOR_PROPERTY_KEY =
"WorkAreaBackgroundColor";

	/**
	 * Buttons dimensions.
	 */
	private static final Dimension BUTTONS_DIMENTIONS = new Dimension(
			BUTTONS_WIDTH, BUTTONS_HEIGHT);

	/**
	 * Constraints determining the elements' arrangement in the panel.
	 */
	private GridBagConstraints constraints = new GridBagConstraints();

	/**
	 * Panel to arrange all elements inside.
	 */
	private JPanel panel = new JPanel(new GridBagLayout());

	/**
	 * Properties file name constant.
	 */
	public static final String PROPERTIES_FILE_NAME = "setup.ini";

	/**
	 * Constructing settings pane.
	 *
	 * @param parent
	 *            The parent class.
	 *
	 * @author Ralitza Koleva
	 *
	 * @email rallly@abv.bg
	 *
	 * @date 21 Oct 2011
	 */
	public SettingsPane(final VitoshaTradeApplet parent) {
		this.parent = parent;

		add(panel);
		this.setPreferredSize(new Dimension(VitoshaTradeApplet.WIDTH,
				VitoshaTradeApplet.HEIGHT));

		/*
		 * Inserts available neurons sizes.
```

```
             */
            chooseNeuronsSize.addItem(Texts.LABEL_NEURONS_SIZE_SMALL);
            chooseNeuronsSize.addItem(Texts.LABEL_NEURONS_SIZE_MEDIUM);
            chooseNeuronsSize.addItem(Texts.LABEL_NEURONS_SIZE_LARGE);

            /*
             * Inserts available neurons' numbers sizes.
             */
            for (int i = MIN_NEURONS_NUMBERS_SIZE; i <= MAX_NEURONS_NUMBERS_SIZE;
i++) {
                  chooseNeuronsNumbersSize.addItem(i);
            }

            /*
             * Sets the size of the buttons.
             */
            chooseColorRegular.setPreferredSize(BUTTONS_DIMENTIONS);
            chooseColorInput.setPreferredSize(BUTTONS_DIMENTIONS);
            chooseColorOutput.setPreferredSize(BUTTONS_DIMENTIONS);
            chooseColorBias.setPreferredSize(BUTTONS_DIMENTIONS);
            chooseColorInputOutput.setPreferredSize(BUTTONS_DIMENTIONS);
            chooseColorInputBias.setPreferredSize(BUTTONS_DIMENTIONS);
            chooseColorOutputBias.setPreferredSize(BUTTONS_DIMENTIONS);
            chooseColorInputOutputBias.setPreferredSize(BUTTONS_DIMENTIONS);
            chooseWorkAreaBackgroundColor.setPreferredSize(BUTTONS_DIMENTIONS);
            save.setPreferredSize(BUTTONS_DIMENTIONS);
            close.setPreferredSize(BUTTONS_DIMENTIONS);

            /*
             * Sets the size of the combobox of neurons sizes and neurons' numbers
             * sizes.
             */
            chooseNeuronsNumbersSize.setPreferredSize(BUTTONS_DIMENTIONS);
            chooseNeuronsSize.setPreferredSize(BUTTONS_DIMENTIONS);

            /*
             * Sets the size of the labels.
             */
            labelColorRegular.setPreferredSize(LABELS_DIMENTIONS);
            labelColorInput.setPreferredSize(LABELS_DIMENTIONS);
            labelColorOutput.setPreferredSize(LABELS_DIMENTIONS);
            labelColorBias.setPreferredSize(LABELS_DIMENTIONS);
            labelColorInputOutput.setPreferredSize(LABELS_DIMENTIONS);
            labelColorInputBias.setPreferredSize(LABELS_DIMENTIONS);
            labelColorOutputBias.setPreferredSize(LABELS_DIMENTIONS);
            labelColorInputOutputBias.setPreferredSize(LABELS_DIMENTIONS);
            labelNeuronsSize.setPreferredSize(LABELS_DIMENTIONS);
            labelNeuronsNumbersSize.setPreferredSize(LABELS_DIMENTIONS);
            labelWorkAreaBackgroundColor.setPreferredSize(LABELS_DIMENTIONS);

            /*
             * Places the elements on the screen, make them editable or not.
             */
            arrangeElements(labelColorRegular, 0, 10, true);
            arrangeElements(showColorRegular, 10, 10, false);
            arrangeElements(chooseColorRegular, 20, 10, true);
            arrangeElements(labelColorInput, 0, 20, true);
            arrangeElements(showColorInput, 10, 20, false);
            arrangeElements(chooseColorInput, 20, 20, true);
            arrangeElements(labelColorOutput, 0, 30, true);
            arrangeElements(showColorOutput, 10, 30, false);
            arrangeElements(chooseColorOutput, 20, 30, true);
            arrangeElements(labelColorBias, 0, 40, true);
```

```
arrangeElements(showColorBias, 10, 40, false);
arrangeElements(chooseColorBias, 20, 40, true);
arrangeElements(labelColorInputOutput, 0, 50, true);
arrangeElements(showColorInputOutput, 10, 50, false);
arrangeElements(chooseColorInputOutput, 20, 50, true);
arrangeElements(labelColorInputBias, 0, 60, true);
arrangeElements(showColorInputBias, 10, 60, false);
arrangeElements(chooseColorInputBias, 20, 60, true);
arrangeElements(labelColorOutputBias, 0, 70, true);
arrangeElements(showColorOutputBias, 10, 70, false);
arrangeElements(chooseColorOutputBias, 20, 70, true);
arrangeElements(labelColorInputOutputBias, 0, 80, true);
arrangeElements(showColorInputOutputBias, 10, 80, false);
arrangeElements(chooseColorInputOutputBias, 20, 80, true);
arrangeElements(labelNeuronsSize, 0, 90, true);
arrangeElements(chooseNeuronsSize, 20, 90, true);
arrangeElements(labelNeuronsNumbersSize, 0, 100, true);
arrangeElements(chooseNeuronsNumbersSize, 20, 100, true);
arrangeElements(labelWorkAreaBackgroundColor, 0, 110, true);
arrangeElements(showWorkAreaBackgroundColor, 10, 110, false);
arrangeElements(chooseWorkAreaBackgroundColor, 20, 110, true);
arrangeElements(save, 20, 120, true);
arrangeElements(close, 20, 130, true);

/*
 * Gets the color for each type of neuron.
 */
readNeuronsTypesAndWorkAreaBackgroundColor(showColorRegular,
             REGULAR_NEURON_COLOR_PROPERTY_KEY);
readNeuronsTypesAndWorkAreaBackgroundColor(showColorInput,
             INPUT_NEURON_COLOR_PROPERTY_KEY);
readNeuronsTypesAndWorkAreaBackgroundColor(showColorOutput,
             OUTPUT_NEURON_COLOR_PROPERTY_KEY);
readNeuronsTypesAndWorkAreaBackgroundColor(showColorBias,
             BIAS_NEURON_COLOR_PROPERTY_KEY);
readNeuronsTypesAndWorkAreaBackgroundColor(showColorInputOutput,
             INPUT_OUTPUT_NEURON_COLOR_PROPERTY_KEY);
readNeuronsTypesAndWorkAreaBackgroundColor(showColorInputBias,
             INPUT_BIAS_NEURON_COLOR_PROPERTY_KEY);
readNeuronsTypesAndWorkAreaBackgroundColor(showColorOutputBias,
             OUTPUT_BIAS_NEURON_COLOR_PROPERTY_KEY);
readNeuronsTypesAndWorkAreaBackgroundColor(showColorInputOutputBias,
             INPUT_OUTPUT_BIAS_NEURON_COLOR_PROPERTY_KEY);

/*
 * Gets the work area background color.
 */
readNeuronsTypesAndWorkAreaBackgroundColor(showWorkAreaBackgroundColor,
             WORK_AREA_BACKGROUND_COLOR_PROPERTY_KEY);

/*
 * Gets the current size of neurons and selects it in the available
 * sizes combobox.
 */
readNeuronsSize();

/*
 * Gets the current size of neuron's numbers and selects it in the
 * available sizes combobox.
 */
readNeuronsNumbersSize();

/*
```

```
		 * Adds Action Listener to each button.
		 */
		chooseColorRegular.addActionListener(new ChooseNeuronColorListener(
				showColorRegular));
		chooseColorInput.addActionListener(new ChooseNeuronColorListener(
				showColorInput));
		chooseColorOutput.addActionListener(new ChooseNeuronColorListener(
				showColorOutput));
		chooseColorBias.addActionListener(new ChooseNeuronColorListener(
				showColorBias));
		chooseColorInputOutput.addActionListener(new ChooseNeuronColorListener(
				showColorInputOutput));
		chooseColorInputBias.addActionListener(new ChooseNeuronColorListener(
				showColorInputBias));
		chooseColorOutputBias.addActionListener(new ChooseNeuronColorListener(
				showColorOutputBias));
		chooseColorInputOutputBias
				.addActionListener(new ChooseNeuronColorListener(
						showColorInputOutputBias));
		chooseWorkAreaBackgroundColor
				.addActionListener(new ChooseNeuronColorListener(
						showWorkAreaBackgroundColor));

		save.addActionListener(new ActionListener() {

			/**
			 * Saves the newly chosen colors to a file.
			 *
			 * @author Ralitza Koleva
			 *
			 * @email rallly@abv.bg
			 *
			 * @date 21 Oct 2011
			 */
			public void actionPerformed(ActionEvent event) {
				try {
					Properties properties = new Properties();

					/*
					 * Loads all properties from the property file.
					 */
					FileInputStream allProperties = new
FileInputStream(
							PROPERTIES_FILE_NAME);
					properties.load(allProperties);

					/*
					 * Calls a function to write the newly chosen
colors to a
					 * file.
					 */
					writeNeuronsTypesAndWorkAreaBackgroundColor(
							showColorRegular,
							REGULAR_NEURON_COLOR_PROPERTY_KEY,
properties);

	writeNeuronsTypesAndWorkAreaBackgroundColor(showColorInput,
							INPUT_NEURON_COLOR_PROPERTY_KEY,
properties);
					writeNeuronsTypesAndWorkAreaBackgroundColor(
							showColorOutput,
OUTPUT_NEURON_COLOR_PROPERTY_KEY,
							properties);
```

```
        writeNeuronsTypesAndWorkAreaBackgroundColor(showColorBias,
                                                    BIAS_NEURON_COLOR_PROPERTY_KEY,
properties);
                                    writeNeuronsTypesAndWorkAreaBackgroundColor(
                                                    showColorInputOutput,

        INPUT_OUTPUT_NEURON_COLOR_PROPERTY_KEY, properties);
                                    writeNeuronsTypesAndWorkAreaBackgroundColor(
                                                    showColorInputBias,
                                                    INPUT_BIAS_NEURON_COLOR_PROPERTY_KEY,
properties);
                                    writeNeuronsTypesAndWorkAreaBackgroundColor(
                                                    showColorOutputBias,
                                                    OUTPUT_BIAS_NEURON_COLOR_PROPERTY_KEY,
properties);
                                    writeNeuronsTypesAndWorkAreaBackgroundColor(
                                                    showColorInputOutputBias,

        INPUT_OUTPUT_BIAS_NEURON_COLOR_PROPERTY_KEY,
                                                    properties);
                                    writeNeuronsTypesAndWorkAreaBackgroundColor(
                                                    showWorkAreaBackgroundColor,

        WORK_AREA_BACKGROUND_COLOR_PROPERTY_KEY, properties);

                                    /*
                                     * Calls a function to save the newly chosen
neurons size to
                                     * a file.
                                     */
                                    writeNeuronsSize(properties);

                                    /*
                                     * Calls a function to save the newly chosen
neurons'
                                     * numbers size to a file.
                                     */
                                    writeNeuronsNumbersSize(properties);

                                    FileOutputStream out = new FileOutputStream(
                                                    PROPERTIES_FILE_NAME);
                                    properties.store(out, "");
                                    allProperties.close();
                                    out.close();
                            } catch (Exception ex) {
                                    ex.printStackTrace();
                            } finally {
                                    parent.showNetworkPane();
                            }
                    }
            });

            close.addActionListener(new ActionListener() {

                    /**
                     * Closes the settings pane and goes to the screen for loading
ANN.
                     *
                     * @author Ralitza Koleva
                     *
                     * @email rallly@abv.bg
                     *
```

```
                     * @date 10 Nov 2011
                     */
                    public void actionPerformed(ActionEvent event) {
                          parent.showNetworkPane();
                    }
              });
       }

       /**
        * Reads the current neuron types colors and work area background color from
        * a file and shows the colors in their corresponding fields.
        *
        * @param showNeuronColorField
        *            The text field to be colored in the current color.
        *
        * @param colorProperty
        *            The parameter name, containing the color.
        *
        * @author Ralitza Koleva
        *
        * @email rallly@abv.bg
        *
        * @date 21 Oct 2011
        */
       public void readNeuronsTypesAndWorkAreaBackgroundColor(
                    JTextField showNeuronColorField, String colorProperty) {
              try {
                    Properties properties = new Properties();
                    FileInputStream in = new FileInputStream(PROPERTIES_FILE_NAME);
                    properties.load(in);
                    int color =
Integer.parseInt(properties.getProperty(colorProperty));
                    Color neuronColor = new Color(color);
                    showNeuronColorField.setBackground(neuronColor);
              } catch (Exception ex) {
                    ex.printStackTrace();
              }
       }

       /**
        * Writes the new neuron types colors and the new work area background color
        * to a file.
        *
        *
        * @param showNeuronColorField
        *            The text field to be colored in the current color.
        *
        * @param colorProperty
        *            The parameter name, containing the color.
        *
        * @param properties
        *            Properties to store the neuron types colors.
        *
        * @author Ralitza Koleva
        *
        * @email rallly@abv.bg
        *
        * @date 21 Oct 2011
        */
       public void writeNeuronsTypesAndWorkAreaBackgroundColor(
                    JTextField showNeuronColorField, String colorProperty,
                    Properties properties) {
```

```
			String color = Integer.toString(showNeuronColorField.getBackground()
					.getRGB());
			properties.setProperty(colorProperty, color);
		}

		/**
		 * Reads the current neurons size from a file and selects it in the
		 * available sizes combobox.
		 *
		 * @author Ralitza Koleva
		 *
		 * @email rallly@abv.bg
		 *
		 * @date 06 Dec 2011
		 */
		public void readNeuronsSize() {
			try {
				Properties properties = new Properties();
				FileInputStream in = new FileInputStream(PROPERTIES_FILE_NAME);
				properties.load(in);
				String currentNeuronsSize =
properties.getProperty("NeuronsSize");
				chooseNeuronsSize.setSelectedItem(currentNeuronsSize);
			} catch (Exception ex) {
				ex.printStackTrace();
			}
		}

		/**
		 * Reads the current neurons' numbers size from a file and selects it in the
		 * available sizes combobox.
		 *
		 * @author Ralitza Koleva
		 *
		 * @email rallly@abv.bg
		 *
		 * @date 23 Nov 2011
		 */
		public void readNeuronsNumbersSize() {
			try {
				Properties properties = new Properties();
				FileInputStream in = new FileInputStream(PROPERTIES_FILE_NAME);
				properties.load(in);
				int currentNeuronsNumberSize = Integer.parseInt(properties
						.getProperty("NeuronsNumbersSize"));

		chooseNeuronsNumbersSize.setSelectedItem(currentNeuronsNumberSize);
			} catch (Exception ex) {
				ex.printStackTrace();
			}
		}

		/**
		 * Writes the new size of neurons to a file.
		 *
		 * @param properties
		 *            Properties to store the size of neurons.
		 *
		 * @author Ralitza Koleva
		 *
		 * @email rallly@abv.bg
		 *
		 * @date 06 Dec 2011
```

```
	 */
	public void writeNeuronsSize(Properties properties) {
		String neuronsSelectedSize = chooseNeuronsSize.getSelectedItem()
				.toString();
		properties.setProperty("NeuronsSize", neuronsSelectedSize);
	}

	/**
	 * Writes the new size of neurons' numbers to a file.
	 *
	 * @param properties
	 *            Properties to store the size of neurons' numbers.
	 *
	 * @author Ralitza Koleva
	 *
	 * @email rallly@abv.bg
	 *
	 * @date 23 Nov 2011
	 */
	public void writeNeuronsNumbersSize(Properties properties) {
		String neuronsNumbersSelectedSize = chooseNeuronsNumbersSize
				.getSelectedItem().toString();
		properties
				.setProperty("NeuronsNumbersSize",
neuronsNumbersSelectedSize);
	}

	/**
	 * Helps arranging the elements on the screen.
	 *
	 * @param component
	 *            The component to be placed inside the panel.
	 *
	 * @param gridX
	 *            X-coordinate of the component.
	 *
	 * @param gridY
	 *            Y-coordinate of the component.
	 *
	 * @param editable
	 *            Determines whether the component is enabled.
	 *
	 * @author Ralitza Koleva
	 *
	 * @email rallly@abv.bg
	 *
	 * @date 25 Oct 2011
	 */
	public void arrangeElements(JComponent component, int gridX, int gridY,
			boolean enabled) {
		constraints.gridx = gridX;
		constraints.gridy = gridY;
		constraints.insets = new Insets(10, 10, 8, 10);
		panel.add(component, constraints);
		component.setEnabled(enabled);
	}
}
```

## SymbolPeriodKeyValue.java

```
/**
 * Gives a JComboBox two values - a visible value in the drop down menu and a
 * hidden one.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 29 Oct 2011
 */
public class SymbolPeriodKeyValue {

	/**
	 * Hidden value of the JComboBox item.
	 */
	private String key;

	/**
	 * Visible value of the JComboBox item.
	 */
	private String value;

	/**
	 * SymbolPeriodKeyValue constructor.
	 *
	 * @param key
	 *            Hidden value of the JComboBox item.
	 *
	 * @param value
	 *            Visible value of the JComboBox item.
	 *
	 * @author Ralitza Koleva
	 *
	 * @email rallly@abv.bg
	 *
	 * @date 29 Oct 2011
	 */
	public SymbolPeriodKeyValue(String key, String value) {
		this.key = key;
		this.value = value;
	}

	/**
	 * Gets the visible value of the JComboBox item.
	 *
	 * @author Ralitza Koleva
	 *
	 * @email rallly@abv.bg
	 *
	 * @date 29 Oct 2011
	 *
	 * @return The visible value of the JComboBox item.
	 */
	public String getValue() {
		return (value);
	}

	/**
	 * Gets the hidden value of the JComboBox item.
	 *
	 * @author Ralitza Koleva
```

```
         *
         * @email rallly@abv.bg
         *
         * @date 29 Oct 2011
         *
         * @return The hidden value of the JComboBox item.
         */
        public int getKey() {
               return (Integer.parseInt(key));
        }

        /**
         * Converts the visible value of the JComboBox item to a string.
         *
         * @author Ralitza Koleva
         *
         * @email rallly@abv.bg
         *
         * @date 29 Oct 2011
         *
         * @return The visible value of the JComboBox item as a string.
         */
        public String toString() {
               return (value);
        }
}
```

## Texts.java

```
/**
 * Contains all texts used in the applet.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 05 Dec 2011
 */
public class Texts {

        /**
         * Menu text.
         */
        public static final String MENU_COMMANDS = "Commands";

        /**
         * Menu item text.
         */
        public static final String MENU_ITEM_LOGIN = "Login";

        /**
         * Menu item text.
         */
        public static final String MENU_ITEM_LOGOUT = "Logout";

        /**
         * Menu text.
         */
        public static final String MENU_NETWORK = "Network";

        /**
         * Menu item text.
```

```
     */
    public static final String MENU_ITEM_NEW = "New";

    /**
     * Menu item text.
     */
    public static final String MENU_ITEM_LOAD = "Load";

    /**
     * Menu item text.
     */
    public static final String MENU_ITEM_DELETE = "Delete";

    /**
     * Menu text.
     */
    public static final String MENU_TOOLS = "Tools";

    /**
     * Menu item text.
     */
    public static final String MENU_ITEM_SETTINGS = "Settings";

    /**
     * Menu text.
     */
    public static final String MENU_MESH = "Mesh";

    /**
     * Menu item text.
     */
    public static final String MENU_ITEM_ACTIVITIES = "Activities";

    /**
     * Menu item text.
     */
    public static final String MENU_ITEM_WEIGHTS = "Weights";

    /**
     * Menu item text.
     */
    public static final String MENU_ITEM_BOTH = "Both";

    /**
     * Menu item text.
     */
    public static final String MENU_ITEM_SOLID = "Solid";

    /**
     * Menu item text.
     */
    public static final String MENU_ITEM_NUMBERING = "Numbering";

    /**
     * Text label.
     */
    public static final String LABEL_DATABASE_HOST = "Database host:";

    /**
     * Text label.
     */
    public static final String LABEL_DATABASE_PORT = "Database port:";
```

```
/**
 * Text label.
 */
public static final String LABEL_USERNAME = "Username:";

/**
 * Text label.
 */
public static final String LABEL_PASSWORD = "Password:";

/**
 * Text label.
 */
public static final String LABEL_ANN_ID = "ID:";

/**
 * Text label.
 */
public static final String LABEL_SYMBOL = "Symbol:";

/**
 * Text label.
 */
public static final String LABEL_PERIOD = "Period:";

/**
 * Text label.
 */
public static final String LABEL_NUMBER_NEURONS = "Number of neurons:";

/**
 * Text label.
 */
public static final String LABEL_SOURCE = "Source:";

/**
 * Text label.
 */
public static final String LABEL_DESTINATION = "Destination:";

/**
 * Text label.
 */
public static final String LABEL_WEIGHT = "Weight:";

/**
 * Text label.
 */
public static final String LABEL_ACTIVITY = "Activity:";

/**
 * Text label.
 */
public static final String LABEL_NUMBER = "No:";

/**
 * Text label.
 */
public static final String LABEL_BIAS_NEURON = "Bias";

/**
 * Text label.
 */
```

```
	public static final String LABEL_INPUT_NEURON = "Input";

	/**
	 * Text label.
	 */
	public static final String LABEL_OUTPUT_NEURON = "Output";

	/**
	 * Text label.
	 */
	public static final String LABEL_REGULAR_NEURON = "Regular";

	/**
	 * Text label.
	 */
	public static final String LABEL_REGULAR_NEURON_COLOR = "REGULAR NEURON
color";

	/**
	 * Text label.
	 */
	public static final String LABEL_INPUT_NEURON_COLOR = "INPUT NEURON color";

	/**
	 * Text label.
	 */
	public static final String LABEL_OUTPUT_NEURON_COLOR = "OUTPUT NEURON color";

	/**
	 * Text label.
	 */
	public static final String LABEL_BIAS_NEURON_COLOR = "BIAS NEURON color";

	/**
	 * Text label.
	 */
	public static final String LABEL_INPUT_OUTPUT_NEURON_COLOR = "INPUT-OUTPUT
NEURON color";

	/**
	 * Text label.
	 */
	public static final String LABEL_INPUT_BIAS_NEURON_COLOR = "INPUT-BIAS NEURON
color";

	/**
	 * Text label.
	 */
	public static final String LABEL_OUTPUT_BIAS_NEURON_COLOR = "OUTPUT-BIAS
NEURON color";

	/**
	 * Text label.
	 */
	public static final String LABEL_INPUT_OUTPUT_BIAS_NEURON_COLOR = "INPUT-
OUTPUT-BIAS NEURON color";

	/**
	 * Text label.
	 */
	public static final String LABEL_NEURONS_SIZE = "Size of neurons";

	/**
```

```
         * Text label.
         */
        public static final String LABEL_NEURONS_SIZE_SMALL = "Small";

        /**
         * Text label.
         */
        public static final String LABEL_NEURONS_SIZE_MEDIUM = "Medium";

        /**
         * Text label.
         */
        public static final String LABEL_NEURONS_SIZE_LARGE = "Large";

        /**
         * Text label.
         */
        public static final String LABEL_NEURONS_NUMBERS_SIZE = "Size of neurons'
numbers";

        /**
         * Text label.
         */
        public static final String LABEL_WORK_AREA_BACKGROUND_COLOR = "Work area
background color";

        /**
         * Text label.
         */
        public static final String LABEL_X_COORDINATE = "X:";

        /**
         * Text label.
         */
        public static final String LABEL_Y_COORDINATE = "Y:";

        /**
         * Text label.
         */
        public static final String LABEL_BUTTON_LOGIN = "Login";

        /**
         * Text label.
         */
        public static final String LABEL_BUTTON_REFRESH = "Refresh";

        /**
         * Text label.
         */
        public static final String LABEL_BUTTON_LOAD = "Load";

        /**
         * Text label.
         */
        public static final String LABEL_BUTTON_SAVE = "Save";

        /**
         * Text label.
         */
        public static final String LABEL_BUTTON_SAVE_CAPS = "SAVE";

        /**
         * Text label.
```

```
         */
        public static final String LABEL_BUTTON_CLOSE_CAPS = "CLOSE";

        /**
         * Text label.
         */
        public static final String LABEL_BUTTON_DELETE = "Delete";

        /**
         * Text label.
         */
        public static final String LABEL_BUTTON_CHOOSE_COLOR = "Choose a color";

        /**
         * Information message.
         */
        public static final String INFORMATION_SELECT_ANN_ID = "Select ID.";

        /**
         * Information message.
         */
        public static final String INFORMATION_SELECT_ANN_ID_TITLE = "Missing
information";

        /**
         * Information message.
         */
        public static final String INFORMATION_SELECT_SYMBOL_AND_PERIOD = "Select
symbol, period and number of neurons.";

        /**
         * Information message.
         */
        public static final String INFORMATION_SELECT_SYMBOL_AND_PERIOD_TITLE =
"Missing information";

        /**
         * Error message.
         */
        public static final String ERROR_DATABASE_PORT = "Invalid database port
value.";

        /**
         * Error message.
         */
        public static final String ERROR_DATABASE_CONNECT = "Invalid credentials.
Connection failed.";

        /**
         * Error message.
         */
        public static final String ERROR_DATABASE_CONNECT_TITLE = "Connection error";

        /**
         * Error message.
         */
        public static final String ERROR_LOAD_ANN_LIST = "Load ANN list fail.";

        /**
         * Error message.
         */
        public static final String ERROR_LOAD_CURRENCY_PAIRS_LIST = "Load currency
pairs list fail.";
```

```
	/**
	 * Error message.
	 */
	public static final String ERROR_LOAD_PERIODS_LIST = "Load periods list
fail.";

	/**
	 * Error message.
	 */
	public static final String ERROR_NUMBER_DATABASE_RESULTS = "Wrong number of
database results.";

	/**
	 * Error message.
	 */
	public static final String ERROR_LOAD_ANN_PART_1 = "Load ANN with id ";

	/**
	 * Error message.
	 */
	public static final String ERROR_LOAD_ANN_PART_2 = " fail.";

	/**
	 * Error message.
	 */
	public static final String ERROR_UPDATE_ANN = "ANN update failed.";

	/**
	 * Error message.
	 */
	public static final String ERROR_ADD_ANN = "Adding a new ANN failed.";

	/**
	 * Error message.
	 */
	public static final String ERROR_DELETE_ANN = "Delete ANN failed.";
}
```

## VitoshaTradeApplet.java

```
	/**
	 * Visualizing new ANN pane.
	 *
	 * @author Ralitza Koleva
	 *
	 * @email rallly@abv.bg
	 *
	 * @date 18 Sep 2011
	 */
	public void showNewAnnPane() {
		remove(networkPane);
		remove(loginPane);
		remove(neuronPane);
		remove(connectionPane);
		remove(deleteAnnPane);
		remove(settingsPane);
		add(newAnnPane, BorderLayout.EAST);
		newAnnPane.revalidate();
		workArea.setVisible(false);
		repaint();
	}
```

```
/**
 * Visualizing delete ANN pane.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 27 Nov 2011
 */
public void showDeleteAnnPane() {
       remove(networkPane);
       remove(loginPane);
       remove(neuronPane);
       remove(connectionPane);
       remove(newAnnPane);
       remove(settingsPane);
       add(deleteAnnPane, BorderLayout.EAST);
       deleteAnnPane.revalidate();
       workArea.setVisible(false);
       repaint();
}

/**
 * Visualizing settings pane.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 21 Oct 2011
 */
public void showSettingsPane() {
       remove(networkPane);
       remove(loginPane);
       remove(neuronPane);
       remove(connectionPane);
       remove(newAnnPane);
       remove(deleteAnnPane);
       add(settingsPane, BorderLayout.EAST);
       settingsPane.revalidate();
       repaint();
}

/**
 * Shows the menu bar when a user logs in.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 03 Sep 2011
 */
public void showMenuBarAfterLogin() {
       menuBar.add(networkMenu);
       networkMenu.add(newNetworkItem);
       networkMenu.add(loadNetworkItem);
       networkMenu.add(deleteNetworkItem);
       menuBar.add(toolsMenu);
       toolsMenu.add(settingsItem);
       toolsMenu.add(meshMenu);
       toolsMenu.add(numberingItem);
       meshItemsGroup.add(meshActivitiesItem);
```

```
        meshItemsGroup.add(meshWeightsItem);
        meshItemsGroup.add(meshBothItem);
        meshItemsGroup.add(meshSolidItem);
        meshMenu.add(meshActivitiesItem);
        meshMenu.add(meshWeightsItem);
        meshMenu.add(meshBothItem);
        meshMenu.add(meshSolidItem);

        /*
         * At the beginning in menu "Tools->Mesh" the option "Solid" is
         * selected.
         */
        if (meshActivitiesItem.isSelected() == false
                    && meshWeightsItem.isSelected() == false
                    && meshBothItem.isSelected() == false
                    && meshSolidItem.isSelected() == false) {
              meshSolidItem.setSelected(true);
        }
        add(workArea);
  }

  /**
   * Removes the menu bar when a user logs out.
   *
   * @author Ralitza Koleva
   *
   * @email rallly@abv.bg
   *
   * @date 03 Sep 2011
   */
  public void hideMenuBarAfterLogout() {
        menuBar.remove(networkMenu);
        menuBar.remove(toolsMenu);
        remove(workArea);
        remove(loginPane);
        remove(networkPane);
        remove(neuronPane);
        remove(connectionPane);
        remove(newAnnPane);
        remove(deleteAnnPane);
        remove(settingsPane);
  }
```