Program Qmak
Dokument Architektury Programu Qmak

Krzysztof Mroczek Leszek Tur Maciej Zuchniak
Damian Wójcik

2 lipca 2007

Spis treści

1 Wprowadzenie

1.1 Cel

Dokument ten ukazuje system Qmak od strony architektury. Celem jest pokazanie systemu z różnych perspektyw. Zadaniem tego dokumentu jest również uchwycenie wszelkich rozwiązań projektowych, które później zostaną zastosowane do implementacji systemu.

1.2 Zakres

Niniejszy dokument prezentuje główne założenia projektowe systemu wizualizacji przetwarzania danych - Qmak. Obejmuje on całość działania systemu, jednak bez szczegółowo opisanych rozwiązań implementacyjnych.

1.3 Definicje

Qmak - nazwa projektowanego systemu.

Rseslib - biblioteka napisana w języku Java, udostępniająca funkcje do przetwarzania danych.

Swing - biblioteka w Javie zawierająca komponenty do tworzenia graficznego interfejsu użytkownika.

1.4 Materiały źródłowe

Niniejszy dokument oraz aplikacja, która powstanie na jego podstawie są częściowo oparte o pomysły, dokumentację oraz kody źródłowe programów Trickster oraz VisualRseslib, które zostały wykonane w latach poprzednich w ramach zajęć ZPP na MIMUW. W szczególności fragmentu kodów źródłowych wyżej wymianionych programów zostały z odpowiednimi zmianami użyte powtórnie w programie Qmak.

1.5 Omówienie reszty dokumentu

W kolejnych punktach tego dokumentu zostały przedstawione różne spojrzenia na architekturę sytemu. Do wspomnianych perspektyw należą: perspektywa przypadków użycia, dekompozycja logiczna systemu, dekompozycja na procesy, sposób implementacji, reprezentacja danych, sposób instalacji i końcowa wydajność systemu.

2 Prezentacja architektury systemu

W dalszej części tego dokumentu opis architektury systemu został podzielony na nastepujące podpunkty:

- Przypadki użycia - opis przypadków użycia zawartych wcześniej w dokumencie BUC. Tym razem są one przedstawione bardziej od technicznej strony ich realizacji.
- Dekompozycja logiczna - podział systemu na klasy i moduły.
- Podział na procesy - wyszczególnienie procesów które będą działały w ramach systemu.
- Implementacja - struktura klas pisanych w Javie.
- Instalacja - sposób instalacji programu.

3 Założenia i zależności

3.1 Bezpieczeństwo i trwałość danych

Bezpieczeństwo i trwałość danych związanych z programem jest zapewniana poprzez lokalność przechowywanych danych.

3.2 Przenośność

Program można zainstalować na komputerach o architekturze pozwalającej na działanie maszyny wirtualnej Javy oraz odpowiednich bibliotek graficznych Javy. W ramach projektu zostanie zapewniona możliwość uruchamiania programu na systemach operacyjnych Windows oraz Linux.

3.3 Modularność

Poszczególne moduły systemu będą związane z konkretnymi funkcjonalnościami w systemie i będą miały dobrze opisany interfejs do komunikacji, tak aby można było ponownie je wykorzystać przy podobnych projektach informatycznych lub przy rozbudowie programu.

3.4 Architektura

Program jest przewidziany do instalacji na końcówkach. Nie będzie optymalizowany pod względem działania na wielu procesorach jednocześnie. W tym zakresie należy polegać na maszynie wirtualnej Javy.

3.5 Interfejs użytkownika

Projektowany graficzny interfejs użytkownika powinien być czytelny, wygodny w nawigacji i powinien umożliwiać:

- dostęp do wszystkich funkcjonalności poprzez kliknięcia myszką
- dostęp do dowolnej funkcjonalności poprzez nie więcej niż cztery kliknięcia myszką

- płynne przesuwanie ikonek w otwartym projekcie na komputerach klasy porównywalnej z komputerami w wydziałowym laboratorium komputerowym.
- informowanie użytkownika jaka czynność jest właśnie wykonywana, dlaczego musi czekać
- przy przegladaniu tebel z danymi program powinien zaznaczać kolorowym tłem wiersze danych niosące pewną szczególną informację dla użytkownika
- możliwość przerwania wykonywania czynności, która może ze względu na złożoność obliczeniową wykonywać się kilka minut lub dłużej

3.6 Wymagane oprogramowanie

Do kompilacji i uruchomienia produktu będzie potrzebny kompilator Javy z bibliotekami, projekt będzie napisany w zgodności z pakietem J2SDK 1.6.
Program zostanie napisany w środowisku programistycznym Eclipse, program będzie korzystać z bibliotek graficznych Swing, JFreeChart dla Javy.

4 Przegląd przypadków użycia

4.1 Krótki opis

4.1.1 Utworzenie nowego projektu

Przypadek użycia pozwalający rozpocząć właściwą prace z programem, tworzony jest pusty projekt, użytkownik wtedy ma dostęp do większej części opcji aplikacji.

4.1.2 Zapisywanie projektu

Przypadek ten odnosi się do zapisania stanu projektu na dysk, umożliwia to późniejszy powrót do stanu w którym projekt był tuż przed zapisaniem i kontynuowanie pracy nad nim.

4.1.3 Wczytanie projektu

Wczytanie do pamięci operacyjnej stanu projektu zapisanego w pamięci podręcznej Wczytany projekt znajduje się w stanie w jakim był przed zapisem. Użytkownik może kontynuować pracę nad projektem.

4.1.4 Budowa grafu historii przepływu danych

Przypadek ten będzie wykonywany na bieżąco zawsze gdy zostanie dodany do projektu jakiś nowy element. Dodanie nowej tabeli spowoduje ukazanie się ikony tabeli w pewnym miejscu na grafie. Wytrenowanie klasyfikatora lub multiklasyfikatora z danej tabeli spowoduje dodanie

do projektu nowego, skonfigurowanego klasyfikatora i ukazanie jego ikony na grafie historii przepływu danych. Na grafie historii przepływu danych ikony reprezentujące powiązane ze sobą elementy będą łączone strzałkami o zwrocie zgodnym z kolejnością powstawania. Na grafie będą też reprezentowane wyniki testu.

4.1.5 Zastosowanie funkcji do pojedynczych wierszy danych

Umożliwia wykonanie określonej funkcji na dowolnym wierszu w przetwarzanej tabeli: np klasyfikacji tylko jednego wiersza, edycji jednego wiersza tabelii.

4.1.6 Dodanie klasyfikatora

Przypadek ten odpowiada dodaniu klasyfikatora do istniejącego projektu. Podczas dodawania klasyfikatora użytkownik musi przejść przez proces konfiguracji klasyfikatora: zdefiniować wstępnie jego parametry - będzie mógł je później modyfikować i przez to wpływać na jego zachowanie. Elementem konfiguracji jest wskazanie danych treningowych dla klasyfikatora, ale ta czynność będzie mogła być odłożona do czasu próby zastosowania klasyfikatora na konkretnych danych.

4.1.7 Wizualizacja klasyfikatora

Przypadek użycia odpowiadający oglądaniu parametrów gotowego klasyfikatora (tj. skonfigurowanego i wytrenowanego).

4.1.8 Zapis klasyfikatora

Przypadek użycia odpowiadający zapisaniu niezbędnych informacji o klasyfikatorze na jakimś nośniku danych. Umożliwia to odtworzenie stanu tuż przed zapisaniem klasyfikatora poprzez wczytanie z odpowiedniego pliku.

4.1.9 Rejestracja nowej implementacji klasyfikatora

Przypadek użycia pozwalający rozszerzyć funkcjonalność aplikacji o dodatkowy rodzaj klasyfikatora. Pozwala to uelastycznyć aplikację i sprawić że będzie bardziej modyfikowalna.

4.1.10 Dodanie worka klasyfikatorów

Przypadek użycia pozwalający na wprowadzenie do projektu zbioru (worka) klasyfikatorów.

4.1.11 Przepuszczenie danych przez worek klasyfikatorów

Przypadek użycia odpowiadający zastosowaniu wszystkich klasyfikatów z worka do konkretnych danych.

4.1.12 Oglądanie wyników testu

Przypadek użycia odpowiadający przeglądaniu wyników otrzymanych po wykonaniu zadanego testu.

4.1.13 Zapisanie wyników testu

Przypadek użycia odpowiadający zapamietaniu na jakimś nośniku danych otrzymanych wyników testu.

4.1.14 Klasyfikacja pojedynczych wierszy danych

Przypadek użycia pozwalający sprawdzić jak zachowa się klasyfikator dla pojedynczego obiektu. Umożliwia to testowanie zachowania klasyfikatora.

4.1.15 Klasyfikacja całej tabeli

Przypadek użycia pozwalający na zastosowanie klasyfikatora dla tabeli. Klasyfikator ten jednak musi pasować do tabeli inaczej klasyfikacja jest niemożliwa. Jest to jeden z ważniejszych przypadków użycia.

4.1.16 Oglądanie wyników klasyfikacji

Przypadek użycia opisujący proces przeglądania danych jakimi są wyniki uprzednio wykonanej klasyfikacji obiektów tabeli.

4.1.17 Zapis wyników klasyfikacji

Przypadek pozwalający na zapamietanie wyników klasyfikacji.

4.1.18 Dodanie tabelki do projektu

Przypadek odpowiadający poszerzeniu projektu o pewną tabelę. Ikona tej tabeli będzie od razu widoczna na grafie projektu.

4.1.19 Oglądanie danych

Przypadek użycia odpowiadający oglądaniu obiektów znajdujących się w tabelce.

4.1.20 Edycja danych

Umożliwia modyfikowanie i tworzenie od nowa tabel. Użytkownik będzie mógł definiować ilość kolumn, nazwy oraz typy atrybutów. Następnie będzie mógł wypełniać kolejne wiersze i modyfikować je w każdej chwili.

4.1.21 Wizualizacja tabelki na wykresie

Przypadek użycia odpowiadający pokazaniu danych z tabelki w sposób umożliwiający zauważenie pewnych różnic i zależności których nie widać gdy patrzy się na gołe wiersze. Takim sposobem wizualizacji może być na przykład wykres.

4.1.22 Zapis tabelki

Przypadek użycia odpowiadający zapamietaniu tabeli z danymi na nośniku danych.

4.2 Realizacje przypadków użycia

4.2.1 Przykład 1 - Klasyfikacja całej tabeli

- W projekcie jest otworzona tabelka i klasyfikator zgodny z tą tabelką.
- Klikamy na tabelkę, zaznaczając ją.
- Klikamy prawym przyciskiem myszy na wytrenowany klasyfikator i wybieramy opcję: klasyfikuj zaznaczoną tabelę.
- Powstaje nowa tabelka z nową kolumną będącą wynikiem klasyfikacji.
- W projekcie ukazuje się nowa ikona reprezentująca wspomnianą tabelkę, oraz strzałki wskazujące źródło jej pochodzenia.

5 Dekompozycja logiczna systemu

5.1 Omówienie

Dekompozycja logiczna systemu operuje na modułach(podsystemch), pakietach oraz klasach analitycznych.
Główne moduły programu to:

- UI (UserInterface)
- dataprocess

5.2 Moduł UI

Moduł zawiera następujące pakiety:

- ProgViews - obsługa funkcji udostępnianych użytkownikowi w oknach aplikacji
- Chart - obsługa generowania wykresów dla danych

- Progress - obsługa okna postępu wykonywaniej opercji
- ResourceBundle - pliki properites programu związane z opcją pomocy oraz z opcją zmiany języka

5.2.1 ProgViews

Funkcje pakietu ProgViews:

- Interfejs umożliwiający oglądanie, modyfikowanie, zapisywanie, wczytywanie elementów projektu, w szczególności:
  - Klasyfikatorów,
  - Worków klasyfikatorów,
  - Tabel danych
  - Wyników testów.
- Interfejs umożliwiający wizualizację elementów programu.
- Interfejs umożliwiający oglądanie i edycję tabel z danymi.
- Interfejs umozliwiający operacje na pulpicie projektu, np:
  - Przesuwanie i wybieranie ikonek
  - Posługiwanie się wskaźnikiem i menu kontekstowym(prawy klawisz myszki)
  - Rozwijanie menu i wybieranie opcji
- Interfejs pomocy programu.

5.2.2 Chart

Pakiet Chart to zbiór klas służacych do tworzenia graficznych diagramów danych z kolumn tabelek w oparciu o bibliotekę JFreeChart. Pakiet został napisany z wykorzystaniem gotowych komponentów z programu Visual Rseslib.

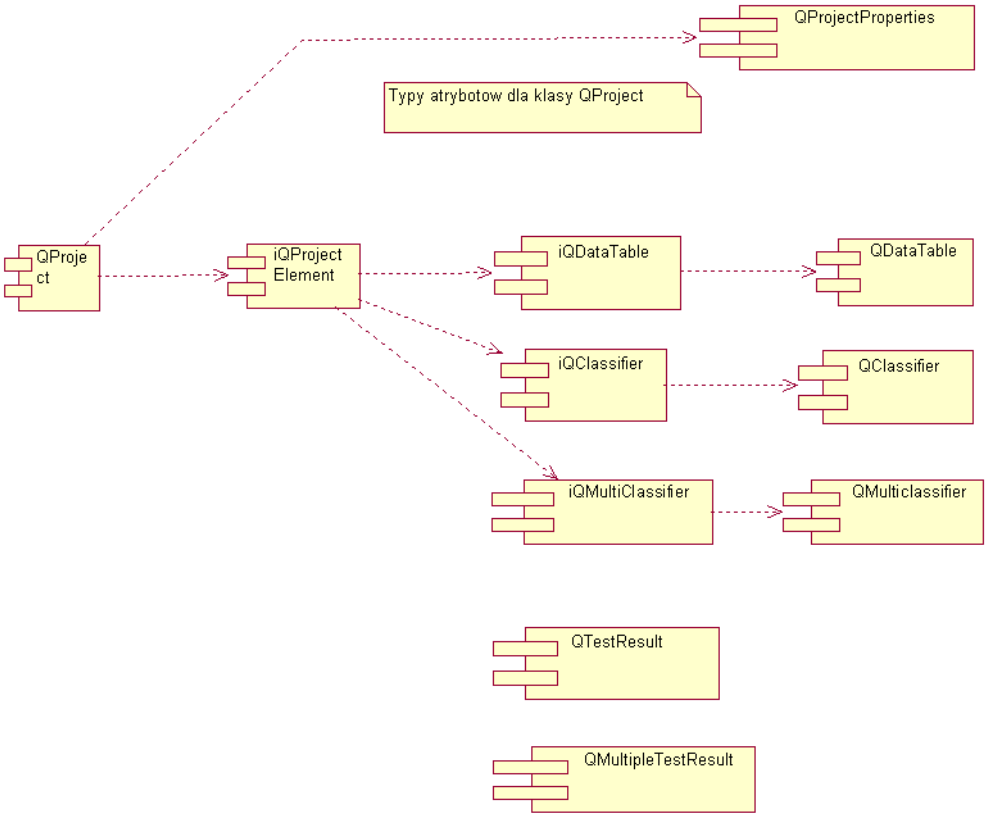
5.2.3 Progress

Pakiet Progress to klasy definiujące działanie okna Progresu.

5.3 Moduł DataProcess

Moduł DataProcess składa się z następujących pakietów:

- Project - przechowywanie projeku i podstawowe operacje na projekcie jako całości
- Table - dostarczanie prostych statystyk dla tabel i przechowywanie tabel
- Classifer - operacje na danych związne z klasyfikacją i klasyfikatorami
- MultiClassifier - operacje na danych związane z workami klasyfikatorów
- TestResult - wyniki testów klasyfikatorów i multiklasyfikatorów



5.3.1 Project

Funkcje pakietu Project:

- Zawiera informacje o wszystkich elementach projektu, ich ustawieniach oraz związkach pomiędzy nimi
- Dokonuje operacji dołączania i usuwania elementów z projektu

- Zawiera informacje o pulpicie projektu i dotyczących niego preferencjach użytkownika
- Dokonuje operacji zapisu projektu na dysku

5.3.2 Table

Funkcje pakietu Table:

- Przechowywanie formatu i danych zawartych w tabeli
- Dostarczanie prostych statystyk dla tabeli
- Edycja zawartości tabeli
- Przechowywanie informacji o umieszczeniu tabeli na pulpicie projektu
- Zapisywanie i odczytywanie tabeli z dysku

5.3.3 Classifier

Funkcje pakietu Classifier:

- Przechowywanie danych o konfiguracji klasyfikatorów
- Dokonywanie operacji klasyfikacji oraz uczenie się klasyfikatora
- Wizualizacja klasyfikatora
- Przechowywanie informacji o umieszczeniu klasyfikatora na pulpicie projektu
- Zapisywanie i odczytywanie klasyfikatora z dysku

5.3.4 MultiClassifier

Funkcje pakietu MultiClassifier:

- Przechowywanie konfiguracji każdego worka klasyfikatorów w projekcie, dodawanie i usuwanie klasyfikatorów z worka
- Prowadzenie operacji klasyfikacji tabeli przez klasyfikatory z worka
- Przechowywanie informacji o umieszczeniu worka na pulpicie projektu
- Zapisywanie i odczytywanie worka z dysku

6 Dekompozycja na procesy

6.1 Dekompozycja na procesy na poziomie aplikacji działającej w systemie

Program jest przewidziany jako aplikacja z jednym, głównym procesem. Jedynym wyjątkiem stanowi obsługa progresów, które wymagają krótkotrwałego uruchomienia nowego procesu.

6.2 Dekompozycja na procesy na poziomie generowania interfejsu graficznego w JavaSwing

Przewidywany jest prosty interfejs graficzny, który przeważnie nie będzie wymagał wielowątkowego wykonywania metod biblioteki Swing. Jedynym wyjątkiem jest okno progresu, w którym wymagane jest krótkotrwałe uruchomienie nowego procesu. Proces ten kończy się zaraz po zamknięciu okna progresu.

7 Instalacja systemu

Przewiduje się możliwość instalowania programu w systemach operacyjnych Linux oraz Windows. Do działania programu będzie niezbędna maszyna wirtualna Javy oraz odpowiednia dostępna bez opłaty biblioteka graficzna (JavaSwing oraz JFreeChart), której funkcjonalności są potrzebne do wizualizacji w programie.
Instalacja będzie polegać na przekopiowaniu z serwera plików programu w postaci spakowanej i rozpakowanie ich do jednego katalogu. Wśród rozpakowanych plików będą skrypty .bat (Windows) oraz .sh(Linux) do uruchomienia programu na danym systemie operacyjnym.
Do uruchomienia programu jest potrzebny komputer z jednym z powyżej wymienionych systemów operacyjnych pracujący w trybie graficznym i wyposażony w pełni funkcjonalną maszynę wirtualną Javy.

8 Implementacja systemu

8.1 Konwencje notacyjne

Na potrzeby projektu zostały przyjęte nastepujące konwencje notacyjne:

- ‘QNazwa’ - klasa fizyczna
- ‘iQNazwa’ - interfejs związany z klasą fizyczną ‘QNazwa’
- ‘(słowo z małej litery)Nazwa’ - atrybuty oraz metody klas fizycznych
- ‘Nazwa’ - klasy fizyczne oraz typy danych spoza programu Qmak, np. String , BigInt

- ‘Nazwa’ lub ‘QNazwa’ - typ argumentu wywołania funkcji
- ‘nazwaNazwacd’ - argument wywołania funkcji

Przykład użycia: Klasa QProject posiada interfejs iQProject, klasa ma metodę setProperties(projectProperties), której argument projectProperties jest instancją klasy QProjectProperties.

8.2 Moduł UI

Moduł odpowiada za komunikację użytkownika z programem. Dla tego modułu cały projekt powstał. Główną ideą całego modułu jest pulpit z ikonkami, za pomocą którego można zrobić wszelkie operacje. Pulpit ma być jak najbardziej komunikatywny użytkownikowi. Dlatego też niemal każda klasa UI będzie się komunikowała z klasą Głównego pulpitu (QProjectView).

8.2.1 Pakiet qmak.ui

Oto spis najważniejszych klas służących kontaktowi z użytkownikiem. Jedynym powiązaniem pomiędzy poszczególnymi klasami są tablice z elementami będącymi obiektami innych klas. Pomiędzy poszczególnymi klasami nie występuje tutaj jednak dziedziczenie (poza biblioteką JavaSwing, z której dziedziczy niemal każda opisywana poniżej klasa). Poniższe klasy w dużej części są wzorowane lub nawet przepisane i z programu Trickster. Stąd też duże podobieństwo (lecz nie identyczność) w nazewnictwie klas.

- QMainFrame
  Klasa MainFrame dziedziczy z klasy JFrame biblioteki JavaSwing. Jest odpowiedzialna za koordynację poszczególnych okien programu oraz za obsługę menu głównego. W programie będzie dokładnie jeden obiekt tej klasy.

  Pola tej klasy:

  1. selectedLanguage - Język, który zostanie pokazany przy następnym uruchomieniu Qmaka
  2. currentLanguage - Aktualnie pokazywany język
  3. messages - obiekt służący do obsługi technologii ’ResourceBoundle’ javy. Stąd są pobierane wszelkie napisy. Dzięki temu możliwa jest zmiana języka programu Qmak.
  4. ClassifierTypes - klasyfikatory dostępne użytkownikowi
  5. qproject - wyświetlany projekt. Może być tylko jeden projekt wyświetlany w jednym momencie. Program był jednak pisany tak, by można było łatwo zmienić ten obiekt na listę obiektów tego typu.
  6. projectView - panel na ktorym bedzie pokazywany aktualny projekt (qproject). Na tym projekcie będą umieszczane takie ikony jak Tabelki, klasyfikatory, czy wyniki klasyfikacji. Wszelkie podstawowe akcje będzie można wykonać poprzez kliknięcie na tym obiekcie prawym przyciskiem myszy.

7. jMenuBar - standardowa lista pól:
   - jMenuFile, a w nim
     * zapiszProjekt
     * wczytajProjekt
     * zapiszWszystkieProjekty
     * nowyProjekt
     * zamknij program
   - jMenuOptions, a w nim
     * Właściwości projektu
     * Zmiana języka
     * Dodaj typ Klasyfikatora
     * Usun typ Klasyfikatora
   - jMenuHelp, a w nim
     * Informacja o autorach, o wersji programu

   Metody tej klasy:

   1. metody obslugujące przyciski menu
   2. metody obsługujące otwarcie i zamknięcie projektu
   3. metody obsługujące zapis i wczytanie projektu

- QProjectView
  Klasa reprezentująca jeden, wybrany projekt użytkownika. Wyświetla i reprezentuje przed użytkownikiem obiekty tego projektu.

  Pola tej klasy:

  1. elements - tablica elementów (ikon).
  2. qsaver - obiekt służący do zapisywani klasyfikatorów
  3. qopener - obiekt służący do wczytywanie klasyfikatorów
  4. qMultiClassifierOpener - obiekt służący do zapisywani multiklasyfikatorów
  5. qMultiClassifierSaver - obiekt służący do wczytywanie multiklasyfikatorów
  6. SelectedTable - obiekt tablicy z tablicy 'elements', który jest aktualnie wybrany
  7. SelectedClassifier - obiekt klasyfikatora z tablicy 'elements', który jest aktualnie wybrany
  8. SelectedMulticlassifier - obiekt multiklasyfikatora z tablicy 'elements', który jest aktualnie wybrany
  9. SelectedMultipleTestResult - obiekt resultatu testu multiklasyfikatora z tablicy 'elements', który jest aktualnie wybrany

10. SelectedTestResult - obiekt resultatu testu klasyfikatora z tablicy 'elements', który jest aktualnie wybrany
11. `popup_QMainView` - popupmenu, które się pokazuje po kliknięciu prawym guzikiem na pulpit
12. `popup_QMulticlassifierIcon` - popupmenu, które się pokazuje po kliknięciu prawym guzikiem na multiklasyfikator
13. `popup_QClassifierIcon` - popupmenu, które się pokazuje po kliknięciu prawym guzikiem na klasyfikator
14. `popup_QVisClassifierIcon` - popupmenu, które się pokazuje po kliknięciu prawym guzikiem na klasyfikator wizualny
15. `popup_QTableIcon` - popupmenu, które się pokazuje po kliknięciu prawym guzikiem na tabelkę
16. `popup_QMultipleTestResultIcon` - popupmenu, które się pokazuje po kliknięciu prawym guzikiem na resultat testów multiklasyfikatora
17. `popup_QTestResultIcon` - popupmenu, które się pokazuje po kliknięciu prawym guzikiem na rezultat testów klasyfikatora

metody tej klasy:

1. metody związane z operacjami dokonanych po wybraniu z opcji list popupmenu
2. metody związane z obsługą ikon na pulpicieczyli
   - zaznaczObiektiIkone - zaznacza dany obiekt, jako wybrany przez użytkownika
   - insertIcon - wstawia ikone na pulpit dla danego obiektu projektu
   - removeObjectUnderIcon - usun ikone wraz z obiektem
   - dajIkone - zwraca ikone, która reprezentuje dany obiekt projektu
3. metody związane z rysowaniem strzałek (metoda główna: rysujStrzalki(Graphics g))

- iQTables
  Klasa ta jest odpowiedzialna za wyświetlenie oraz umożliwienie edycji tabeli z danymi. Jednocześnie stanowi ona klasę otoczkę dla klasy QTableView, która zawiera samą tabelkę.

  Ważniejsze pola tej klasy:

  1. table - obiekt klasy QTableView - klasy reprezentującej tabelkę
  2. dataTable - Tabelka z danymi wyświetlana przez ten obiekt
  3. dataTableElem - ta sama tabelka co dataTable - jej kopia dokładniej - potrzebne do obsługi przycisku 'cancel'
  4. ikona - ikona na pulpicie projektu, która reprezentuje ta tabelkę.
  5. przyciski 'OK' oraz 'Cancel'

Metody tej klasy: Obsługa strony graficznej

- QTableView
  Klasa ta jest odpowiedzialna za wyświetlenie oraz umożliwieniem użytkownikowi podstawowej edycji tabelki

  Ważniejsze pola tej klasy:

  1. owner - obiekt klasy iQTable, który zawiera dany obiekt
  2. tablePopupMenu - popupmenu, które się pokazuje po naciśnięciu prawego przycisku myszy na tabelce

  Metody tej klasy:

  1. Metody obsługujące akcje po wybraniu opcji z popupmenu
  2. Metody obsługujące obsługę tabelki

- QHelp
  Klasa QHelp odpowiedzialna jest za obsługę pomocy użytkownika.

  Ważniejsze pola tej klasy:

  1. help - obiekt obsługujący technologię 'ResourceBoundle'. Za pomocą tego obiektu wczytywane są opisy w helpie
  2. helpTemat - obiekt obsługujący technologię 'ResourceBoundle'. Za pomocą tego obiektu wczytywane są hasła wyświetlane jako spis treści w pomocy
  3. obiekty fizyczne tj. przycisk 'OK'

  Metody tej klasy:

  1. pokaz - pokazuje okna, a w nim wybrane dane haslo
  2. zaznaczHaslo - zaznacza dane haslo

- QIcon
  Obiekty klasy QIcon, to dziedziczące z klasy JLabel ikony na pulpicie projektu.

  Ważniejsze pola tej klasy:

  1. Elem - element projektu, który dana ikona reprezentuje. Element ten jest ogólnej klasy QProjectElement. Żeby go użyć trzeba go zrzutować np. na iQDataTable dla tabelki, lub na iQClassifier dla klasyfikatora
  2. okno;

  Metody tej klasy:

  1. setInactive - Ustaw widok tego obiektu jako nieaktywny

2. setActive - Ustaw widok tego obiektu jako aktywny

- QClassifierPropertiesDialog
  Klasa odpowiedzialna jest za zmianę właściwości klasyfikatora przed wytrenowaniem.

  Ważniejsze pola tej klasy:

  1. prop orig - domyślne właściwości klasyfikatora
  2. klasyfikator - klasyfikator którego dotyczy dane okno dialogowe

  Metody tej klasy:

  1. assignData - przypisuje orginalne właściwości oraz wywołuje metodę umożliwiającą zmianę ich przez użytkownika

- QMulticlassifierShowDialog
  Podana klasa służy wyświetlaniu okienka wizualizującego zawartość multiklasyfikatora oraz operacje konfiguracji klasyfikatorów.

  Ważniejsze pola tej klasy:

  1. multiClassifier - multiklasyfikator którego zawartość będzie wyświetlana za pomoca tej klasy

  Metody tej klasy:

  1. metoda odpowiedzialna za wyświetlanie i konfigurację klasyfikatora
  2. metoda umożliwiająca usuwanie klasyfikatora z worka

- QSimpleResults
  Klasa odpowiadająca za wygląd i zachowanie okienka prezentującego wyniki testu wielokrotnego worka klasyfikatorów.

  Ważniejsze pola tej klasy:

  – qmtResult - obiekt klasy QMultipleTestResult. Zawiera wszelkie dane dotyczące prezentowanych wyników testu.

  Ważniejsze metody:

  – metoda służąca wyświetleniu okienka z wynikami

- QTestResultPanel
  Klasa odpowiadająca za wygląd i zachowanie okienka prezentującego macierz konfuzji dla testu pojedynczego klasyfikatora.

  Ważniejsze pola tej klasy:

  – qtr - pole typu QTestResult. Przechowuje dane dotyczące wyników testu.

Ważniejsze metody:

  - metoda powodująca wyświetlenie okienka z macierzą konfuzji na ekranie

- QAskProgramExit
  Klasa obsługujące operację wyświetlenia dialogu - zapytania o zapisanie projektu po próbie zamknięcia programu przyciskiem x w prawym górnym rogu pulpitu.

- QAskProjectSave
  Klasa pokazująca dialog -zapytanie o zapisanie projektu przy jego zamknięciu.

- QClassifierOpener
  Klasa przeprowadzająca wczytanie zapisanej w pliku instancji klasyfikatora do projektu.

- QClassifierSaver
  Klasa przeprowadzająca zapisanie instancji klasyfikatora do pliku.

- QMultiClassifierOpener
  Klasa przeprowadzająca wczytanie zapisanej w pliku instancji multiklasyfikatora do projektu.

- QMultiClassifierSaver
  Klasa przeprowadzająca zapisanie instancji multiklasyfikatora do pliku.

- QNewProjectUI
  Klasa wyświetlająca okno kreacji nowego projektu i przeprowadzająca jego wpisanie do instancji klasy QMainFrame.

- QVisClassifierView
  Klasa implementująca panel, na którym jest pokazywana wizualizacja klasyfikatora oraz wynik klasyfikacji klasyfikatorem wizualnym. Ważniejsze metody:

  - draw() - narysowanie wizualizacji klasyfikatora wizualnego
  - classifyOne() - narysowanie klasyfikacji wiersza klasyfikatorem wizualnym
  - addComment() - dodanie napisu - komentarza w dolnej części komponentu

8.2.2 Pakiet qmak.UI.Progres

Pakiet ten służy do obsługi paska progresu pokazywanego użytkownikowi. Pasek ten musi zostać wywołany w nowym wątku - w przeciwnym przypadku pasek nie przesuwałby się regularnie i nie spełniałby swojej funkcjonalności. Pakiet zawiera podstawowe dwie klasy

- QVisualProgress - klasa ta służy do wystawienia metod 'set' oraaz 'step' wywoływanych przez bibliotekę rseslib

- QProgress - Fizyczne okno progresu

Dodatkowo w pakiecie tym znajdują się klasy będące implementacją nowych wątków uruchamianych podczas pokazywania progresu.

8.3 Moduł DataProcess

Moduł DataProcess odpowiada za zarządzanie projektem oraz operowanie na danych przy pomocy funkcji dostarczanych przez bibliotekę rseslib 3. Moduł jest mocno związany z interfejsami dostarczanymi przez bibliotekę rseslib 3.

- iQProjectElement
  Interfejs implementowany przez obiekty projektu: QClassifier, QMultiClassifier, QMultipleTestResult, QTestResult.

  Metody definiowane przez interfejs:

  1. isTestResult(), isMultipleTestResult(), isTable(), isClassifier(), isMulticlassifier() - określają typ obiektu
  2. isXMLstoreable(), isFileStoreable() - informują o sposobie zapisu obiektu
  3. set/getName() - pobiera nazwę obiektu
  4. set/getFileName() - ustawia/pobiera nazwę pliku w którym zapisany jest obiekt
  5. set/getPosition() - ustawia/pobiera pozycje obiektu w projekcie
  6. set/getBaseFileExtension() - ustawia/pobiera rozszerzenie dla nazwy pliku

- QClassifier
  Klasa odpowiada za obsluge klasyfikatorów w projekcie.

  Pola tej klasy:

  1. wytrenowany - zmienna mówiąca czy klasyfikator jest wytrenowany
  2. klasyfikator - klasyfikator z biblioteki rseslib
  3. wlasciwosci - właściwości klasyfikatora podawane mu podczas trenowania i klasyfikacji
  4. wlasciwosci pr - inne właściwości klasyfikatora, używane między innymi przy zapisywaniu
  5. typ - obiekt typu QClassifierType wskazujący typ klasyfikatora
  6. p - obiekt klasy Point - określa współrzędne ikony klasyfikatora w projekcie

  Metody tej klasy:

  1. trainOnTable,trainOnTableWithProgress - trenowanie klasyfikatora na tabelce
  2. classify - klasyfikacja tabelki

- QClassifierTypes
  Klasa odpowiada za obsługę dostępnych typów klasyfikatorów w systemie. Ważniejsze pola tej klasy:

1. classifierTypes - zbiór typów dostępnych w systemie (obiektów klasy QClassifierType)

Metody tej klasy:

1. contains - sprawdza czy w systemie jest dostępny podany typ
2. getTypes - zwraca zbiór zarejsetrowanych typów klasyfikatorów
3. add - dodaje typ klasyfikatora
4. remove - usuwa podany typ klasyfikatorów
5. saveClassifierTypesConfiguration - zapisuje konfigurację typów do pliku

- QMultiClassifier
  Klasa odpowiada za obsługę multiklasyfikatorów w systemie.

  Ważniejsze pola tej klasy:

  1. klasyfikatory wytrenowane properties klasyfi p - Point, określa współrzędne ikony

  Metody tej klasy:

  1. size - zwraca ilość klasyfikatorów w multiklasyfikatorze
  2. add - dodaje nowy klasyfikator do worka
  3. remove - usuwa klasyfikator
  4. areTrained - informuje czy klasyfikatory w worku są wytrenowane
  5. classify - wykonuje klasyfikację tabelki klasyfikatorami w worku
  6. trainOn - trenuje klasyfikatory w worku
  7. doCrossValidationTest, doMultipleCrossValidationTest, doMultipleRandomSplitTest - wykonuje odpowiedni test

- QMultipleTestResult
  Klasa odpowiadająca wynikowi testu wielokrotnego dla multiklasyfikatora.

  Ważniejsze pola tej klasy:

  1. wyniki - obiekt będący mapą której kluczami są nazwy klasyfikatorow a wartosciami obiekty klasy TestResult, będące wynikami testów dla tych klasyfikatorów
  2. rodzaj - pole typu string odpowiadające rodzajowe testu. Możliwymi rodzajami są: kroswalidacja, wielokrotna kroswalidacja oraz losowy podział.

  Ważniejsze metody tej klasy:

1. mapToArray() - metoda zwracajaca tablicę dwuwymiarową obiektów, tablica ta jest modelem dla obiektu klasy JTable będącego reprezentacją graficzną wyniku testu
2. showResults() - metoda służąca do wyświetlenia na ekranie okienka z prostymi statystykami z testu

- QTestResult
  Klasa odpowiadająca wynikowi testu worka klasyfikatorów oraz pojedynczego klasyfikatora.

  Najważniejsze pola tej klasy:

  1. wyniki - obiekt będący mapą której kluczami są nazwy klasyfikatorow a wartosciami obiekty klasy TestResult, będące wynikami testów dla tych klasyfikatorów
  2. isFromMultiClassifier - pole typu boolean mówiące czy dany test pochodzi od worka klasyfikatorów czy od pojedynczego klasyfikatora
  3. decAttr - obiekt klasy NominalAttribute - będący zbiorem wartości decyzyjnych
  4. tableName - obiekt klasy String - przechowuje nazwe tabeli na której został wykonany test

  Najważniejsze metody:

  1. noOfDecs() - mówi ile jest możliwych atrybutów decyzyjnych
  2. howManyObjects() - zwraca liczbę obiektow sklasyfikowanych daną decyzją dla klasyfikatora o zadanej nazwie
  3. getAccuracy() - zwraca skuteczność klasyfikatora o zadanej nazwie w postaci liczby zmiennoprzecinkowej
  4. getDecAccuracy() - zwraca skuteczność klasyfikowania obiektów o zadanej decyzji dla klasyfikatora o podanej nazwie
  5. showResults() - wyświetla okienko z prostymi statystykami

- QProject
  Klasa odpowiada zbiorowi elementów w projekcie użytkownika. Odpowiada za:

  1. organizację zapisywania i wczytywania elementów projektu
  2. unikalność nazw
  3. pamiętanie porządku pomiędzy elementami
  4. pamiętanie katalogu głównego projektu

  Najważniejsze pola tej klasy:

  1. projectProperties - obiekt pamiętające właściwości projektu w zakresie nazwenictwa oraz dat

2. projectElements - kolekcja elementw projektu (klasyfikatory oraz ,multi-, tablice, tesrt resulty
3. reserved names - obiekt pamiętający nazewnictwo obiektów w projekcie

Najważniejsze metody:

1. registerName() - dodanie kolejnej nazwy do slownika słów użytych
2. RemocePair(), PemoveParentsOf(), RemoveChildrenOf() - modyfukacja zbioru użytego nazewnictwa w programie
3. getElement() - pobranie elementu projektu na podstawie jego pywatnej nazwy
4. insertElement(), removeElement() - obsługa dodawania i usuwania elementów z projektu
5. getProperties(), setProperties - obsługa właściwości projektu

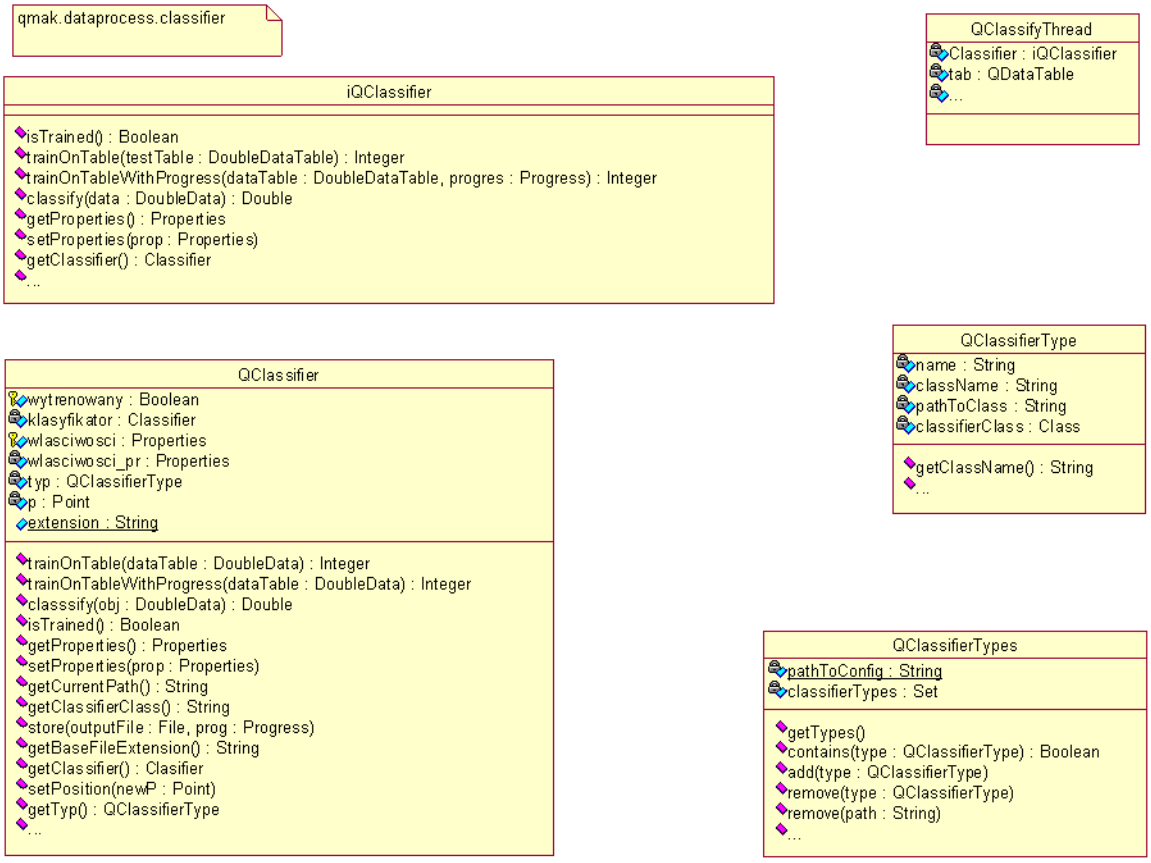
- QProjectProperties
  Klasa przechowuje właściwości projektu takie jak: nazwa, opis, daty zwiazane z projektem. Dane klasy QProjectProperties:

  1. Atrybuty klasy:
     - String projectName - nazwa projektu
     - String fileName - nazwa pliku głównego do którego jest zapisywany projekt
     - Date accessDate - data ostatniego dostępu do projektu
     - Date creationDate - data utworzenia projektu
     - Date modificationDate - data ostatniego dostępu do projektu
     - String author - nazwa atora projektu
     - String description - któtki opis projektu
     - Boolean saved - czy aktualny projekt jest zapisany do pliku
  2. Metody implementowane przez klasę - akcesory:
     - String getName
     - void setName
     - String getFileName
     - void setFileName
     - String getCurrentPath
     - Date getCreationDate
     - void setCreationDate
     - Date getAccessDate
     - void setAccessDate
     - Date getModificationDate
     - void setModificationDate

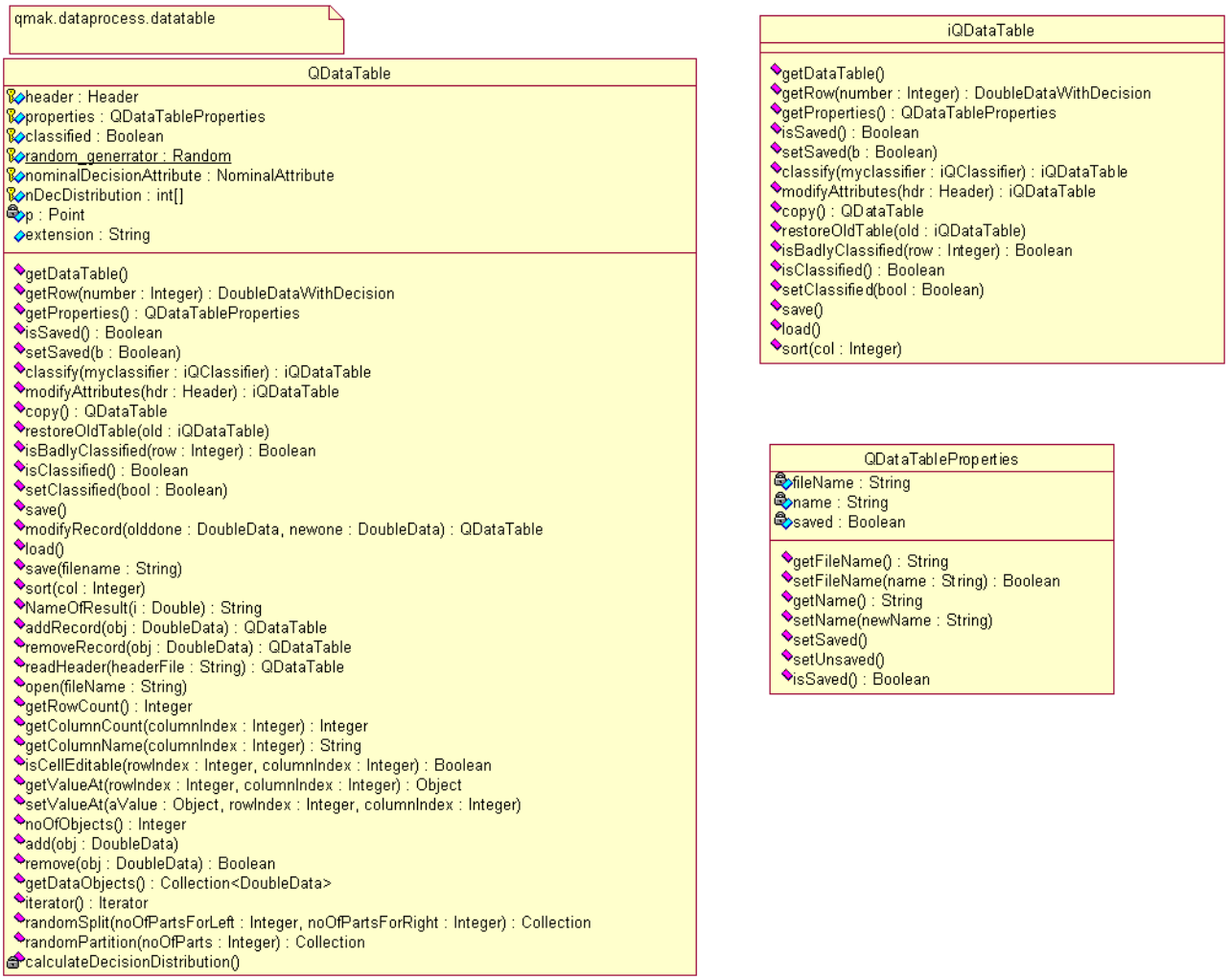
- String getAuthor
- void setAuthor
- String getDescription
- void setDescription
- void setSaved
- void setUnsaved
- boolean isSaved

3. Metody implementowane przez klasę - inne:

- XMLstore




qmak.dataprocess.multiclassifier
iQMultiClassifier
add(klasyfikator : iQClassifier)
remove(name : String)
areTrained() : Boolean
trainOn(tabelka : DoubleDataTable, prog : Progress)
classify(tabelka : DoubleDataTable, prog : Progress) : Map
...
QMultiClassifier
klasyfikatory : ClassifierSet
wytrenowane : Boolean
properties : Properties
klasyfi : Map
p : Point
extension : String
add(klasyfikator : iQClassifier)
remove(name : String)
areTrained() : Boolean
trainOn(tabelka : DoubleDataTable, prog : Progress)
classify(tabelka : DoubleDataTable, prog : Progress) : Map
setTrained()
setNotTrained()
getQClassifier(nazwa : String) : iQClassifier
getClassifierNames() : Set
size() : Integer
addWithNullString(name : String)
doCrossValidationTest() : Map
doMultipleCrossValidationTest() : Map
...
qmak.dataprocess.project
iQProject
getRelatives() : Set
RemovePair(e1 : iQProjectElement, e2 : iQProjectElement)
RemoveParentsOf(e : iQProjectElement)
RemoveChildren Of(e : iQProjectElement)
CreateUniqueName(name : String, isPath : Boolean) : String
GetProjectElements() : Set
insertElement(obj : iQProjectElement) : Boolean
removeElement(obj : iQProjectElement) : Boolean
getProperties() : QProjectProperties
setProperties(prop : QProjectProperties)
saveProject()
saveProjectAs(filename : String)
loadProject()
isSaved() : Boolean
getElement(elementName : String) : iQProjectElement
...
QProjectProperties
projectName : String
fileName : String
accessDate : Date
creationDate : Date
modificationDate : Date
author : String
description : String
saved : Boolean
initialize(newName : String, newFilename : String, newAuthor : String)
getName() : String
setName(newmane : String) : Boolean
getFileName() : String
setFileName(newName : String) : Boolean
getCurrentPath() : String
getCreationDate() : Date
setCreationDate() : Boolean
getAccessDate() : Date
setAccessDate(date : Date) : Boolean
getModificationDate() : Date
setModificationDate(date : Date) : Boolean
getAuthor() : String
setAuthor(newname : String) : Boolean
getDescription() : String
setDescription(des : String) : Boolean
setSeved()
setUnSaved()
...
QProject
projectProperties : QProjectProperties
pokrewienstwo : Set
projectElements : Set
reservedNames : Set
getRelatives() : Set
RemovePair(e1 : iQProjectElement, e2 : iQProjectElement)
RemoveParentsOf(e : iQProjectElement)
RemoveChildren Of(e : iQProjectElement)
CreateUniqueName(name : String, isPath : Boolean) : String
GetProjectElements() : Set
insertElement(obj : iQProjectElement) : Boolean
removeElement(obj : iQProjectElement) : Boolean
loadRelatives()
getProperties() : QProjectProperties
setProperties(prop : QProjectProperties)
saveProject()
saveProjectAs(filename : String)
loadProject()
isSaved() : Boolean
getElement(elementName : String) : iQProjectElement
...
iQProjectElement
isTestResult() : Boolean
isMultipleTestResult() : Boolean
isTable() : Boolean
isClassifier() : Boolean
isMulticlassifier() : Boolean
isXMLstoreable() : Boolean
isFileStoreable() : Boolean
getName() : String
setName(name : String)
getFileName() : String
setFileName(fileName : String)
getPosition() : Point
...
QPara
Rod : iQProjectElement
Dziec : iQProjectElement
...
iQXMLstoreable
XMLstore() : BufferedWriter



8.3.1 Opis implementacji zapisywania i wczytywania projektu

Zapisywanie oraz wczytywanie projektu oraz jego elementów oparte jest na następujących klasach:

1. XmlClassifierOpener - parser pliku xml opisu zapisanego osobno klasyfikatora
2. XmlMulticlassifierOpener - parser pliku xml opisu zapisanego worka klasyfikatorów
3. XmlOpener - parser pliku xml zapisanego projektu

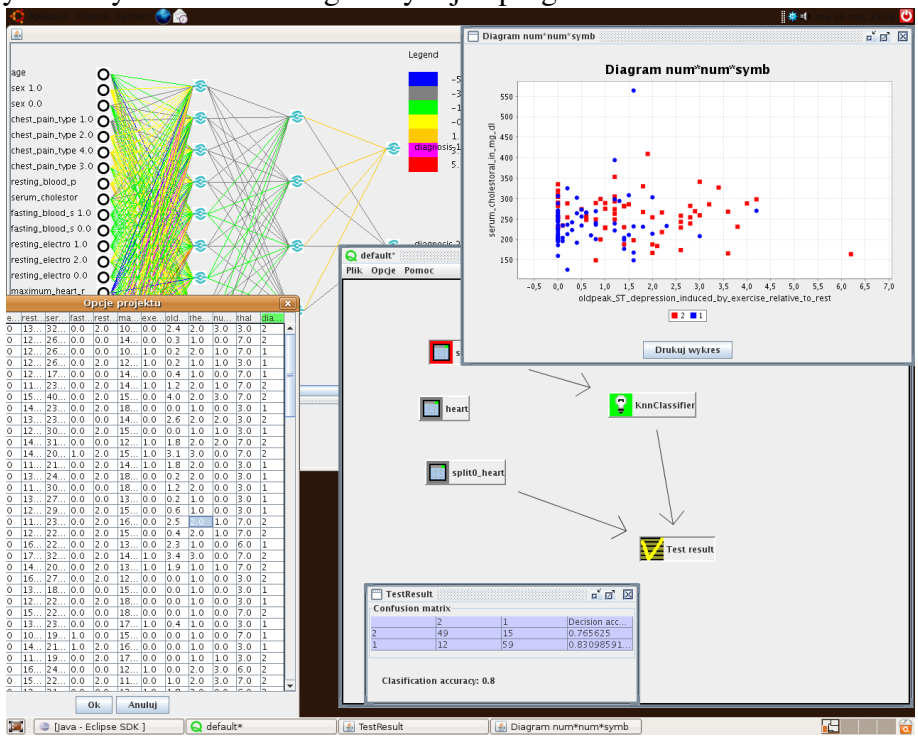
Ponadto w pakiecie ui zawarte są klasy odpowiadające za wyświetlenie odpowiedniego okna.

Zapisywane są przede wszystkim elementy z pakietu dataprocess. Dlatego posiadają one natępujące ważne metody(nie zawsze wszystkie):

- isXmlStoreable() - sprawdzenie czy obiekt może zapisać swój opis w postaci xml
- isFileStorable() - sprawdzenie czy obiekt implementuje interfejs FileStoreable biblioteki Rseslib 3
- setName(), getName() - ustawienie nazwy obiektu(może być użyta przy generowaniu nazwy pliku, do której ma być zapisany obiekt)
- setFileName(), getFileName() - nazwa pliku z opisem xml obiektu
- getcurrentPath() - zwraca ścieżkę do katalogu z plikiekm opisu xml obiektu, przydatne gdy np. oprócz multiklasyfikatora trzeba wczytać też jego klasyfikatory, które są zapisane w osobnych plikach
- XMLstore() - pobranie opisu xml obiektu jako elementu projektu
- XMLWidestore() - pobranie opisu xml obiektu jako obiektu do samodzielnego zapisu w pliku
- store() - zapisanie obiektu do pliku/ów
- load() - wcztanie obiektu na podstawie opisu xml

8.4 Przykładowe screeny

Przykładowy układ okien w gotowym już programie:



9 Przechowywane dane

Podstawową daną w programie jest projekt. Dlatego program wczytuje i zapisuje następujące typy plików:

1. zbiorczy plik projektu
2. plik z danymi i nagłówkiem tabeli
3. plik zapisu klasyfikatora
4. plik zapisu worka klasyfikatorów

9.1 Plik zbiorczy projektu

Plik zbiorczy projektu jest w formacie XML. W pliku projektu zapisywane są następujące informacje:

- nazwa projektu - tag <name>
- data utworzenia projektu - tag <creationdate>
- data ostatniej modyfikacji projektu - tag <moddate>
- właściciel(autor) projektu - tag <author>
- którki opis projektu - tag <description>
- dla każdej tabeli - tag <datatable>:
  - nazwa - tag <name>
  - nazwa pliku w katalogu z projektem - tag <filename>
  - pozycja na pulpicie - tagi <x> <y>
- dla każdego klasyfikatora:
  - nazwa - tag <name>
  - typ - tag <classpath>
  - nazwa pliku z parametrami klasyfikatora biblioteki Rseslib - tag <filename>
  - pozycja na pulpicie - tagi <x> <y>
- dla każdego worka klasyfikatorów:
  - nazwa - tag <name>
  - nazwa pliku z opisem składu worka multiklasyfikatorow - tag <filename>
  - pozycja na pulpicie - tagi <x> <y>

Cały projekt jest opisany wewnątrz tagu <project>.

9.2 Plik danych tabeli

Plik danych tabeli będzie miał format zgodny z formatem w danych tabeli w bibliotece reseslib.

9.3 Plik zapisu klasyfikatora

Każdy klasyfikator w bibliotece rseslib dostarcza własne metody swojego zapisu.

9.4 Plik porządku między obiektami w projekcie

W pliku 'relatives' należy w kolejnych wierszach umieszczać pary: przodek, potomek oddzielone spacją.

9.5 Plik zapisu worka klasyfikatorów

Plik worka klasyfikatorów ma format pliku XML. W pliku będą zamieszczone następujące informacje związane z workiem klasyfikatorów:

- nazwa worka klasyfikatorów - tag <name>
- pozycja na pulpicie - tagi <x>x oraz <y>
- dla każdego klasyfikatora z worka - tag <classificator> :
  - nazwa - tag <name>
  - typ - tag <classpath>
  - plik klasyfikatora - tag <filename>
  - pozycja na pulpicie - tagi <x>x oraz <y>

10 Wydajność systemu

Zakłada się że interfejs Qmaka nie będzie powodował narzutu na działanie biblioteki rseslib. Wydajność powinna więc być zależna jedynie od sposobu implementacji funkcji bibliotecznych, oraz od wielkość przetwarzanych danych. Do uruchomienia programu będzie nadawał się każdy system z zainstalowaną maszyną wirtualną Javy. Interfejs jest jedynie nakładką na bibliotekę, więc to od jej wydajność zależeć będzie ogólna wydajność systemu. Użytkownik chcąc poprawić efektywność pracy z programem, jeśli uzna że jest niewystarczająca, powinien zainwestować w takie podzespoły jak: procesor i pamięć, gdyż to one są głównie odpowiedzialne za wydajność implementowanego systemu. Przewiduje się że do wygodnej pracy z programem wystarczy maszyna wyposażona w procesor 500 MHz i 128 MB pamięci RAM.

10.1 Rozszerzanie

System został zaprojektowany tak, aby dało się go powiększać o dodatkowe komponenty rozszerzające jego funkcjonalność. System da się rozrzerzać o nowe implementacje klasyfikatorów już z poziomu interfejsu użytkownika.

10.2 Niezawodność

Niezawodność systemu jest nie większa niż niezawodności wirtualnej maszyny Javy, ponieważ ten proces jest odpowiedzialny za przydzielanie programowi zasobów. Ponadto program korzysta z biblioteki Rseslib, na której jakość implementacji i niezawodność nie mamy wpływu. Przewiduje się, że powyższe czynniki nie zmniejszą mimo wszystko niezawodności systemu.

10.3 Przenośność

Program można uruchomić na różnych platformach sprzętowych. Mimo, iż z programem będą tylko dostarczone skrypty do uruchamiania programu w systemach Windows oraz Linux, to dla zaawansowanego użytkownika napisanie skryptu uruchamiającego program pod inny system, na którym jest zainstalowana maszyna wirtualna Javy nie powinno być większym problemem.