My Project

Generated by Doxygen 1.10.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

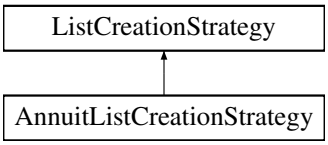
Class Documentation

4.1 AnnuitListCreationStrategy Class Reference

The AnnuitListCreationStrategy class represents a strategy for creating a list of month information for an annuity-based loan.

`#include <calculations.h>`

Inheritance diagram for AnnuitListCreationStrategy:



Public Member Functions

- **AnnuitListCreationStrategy** (Calculations *calc)
- void createList () override

  This method creates a list of monthly payment information using the linear repayment method.

Public Member Functions inherited from ListCreationStrategy

- **ListCreationStrategy** (Calculations *calc)

Protected Attributes

- Calculations * **calculations**

Protected Attributes inherited from ListCreationStrategy

- Calculations * **calculations**

4.1.1 Detailed Description

The AnnuitListCreationStrategy class represents a strategy for creating a list of month information for an annuity-based loan.

Author

Aurelijus Lukšas

4.1.2 Member Function Documentation

4.1.2.1 createList()

```
void AnnuitListCreationStrategy::createList ( )  [override], [virtual]
```

This method creates a list of monthly payment information using the linear repayment method.

Author

Aurelijus Lukšas

Implements ListCreationStrategy.

The documentation for this class was generated from the following files:

- calculations.h
- calculations.cpp

4.2 Calculations Class Reference

The Calculations class represents a loan calculator that performs various calculations related to a loan.

```
#include <calculations.h>
```

Public Member Functions

- **Calculations** ()

  Constructs a Calculations object with default values.
- Calculations (double loan_amount, double annual_percentage, int years, int months, int start, int end, bool is_annuit, bool is_linear)

  Constructs a Calculations object with the specified parameters.
- void recalculate ()

  Recalculates the loan based on the current parameters.
- void setLoanAmount (double amount)

  Sets the loan amount.
- void setAnnualPercentage (double percentage)

  Sets the annual interest rate percentage.
- void setYears (int years)

  Sets the number of years for the loan.

- void setMonths (int months)

  Sets the number of months for the loan.
- void setIsAnnuit (bool isAnnuit)

  Sets whether the loan is annuity-based.
- void setIsLinear (bool isLinear)

  Sets whether the loan is linear-based.
- double getLoanAmount () const

  Gets the loan amount.
- double getAnnualPercentage () const

  Gets the annual interest rate percentage.
- int getYears () const

  Gets the number of years for the loan.
- int getMonths () const

  Gets the number of months for the loan.
- bool getIsAnnuit () const

  Gets whether the loan is annuity-based.
- bool getIsLinear () const

  Gets whether the loan is linear-based.
- std::vector< MonthInfo > getList () const

  Gets the list of month information for the loan.
- void createList ()

  Creates the list of month information for the loan.
- void createAnnuitList ()

  Creates the list of month information for the annuity-based loan.
- void createLinearList ()

  Creates the list of month information for the linear-based loan.
- void setStrategy (ListCreationStrategy *newStrategy)

  Sets the strategy for creating the list of month information.

4.2.1 Detailed Description

The Calculations class represents a loan calculator that performs various calculations related to a loan.

Implementation file for the Calculations class.

This file contains the implementation of the Calculations class, which is responsible for performing loan calculations. The class provides methods to set and get loan parameters, recalculate the loan, and create a list of monthly payment information. The calculations can be done using either annuity or linear repayment methods. Also able to set the strategy of calculations.

Authors

Julius Jauga, Aurelijus Lukšas

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Calculations()

```
Calculations::Calculations (
            double loan_amount,
            double annual_percentage,
            int years,
            int months,
            int start,
            int end,
            bool is_annuit,
            bool is_linear )
```

Constructs a Calculations object with the specified parameters.

Parameters

loan_amount	The loan amount.
annual_percentage	The annual interest rate percentage.
years	The number of years for the loan.
months	The number of months for the loan.
start	The starting month for the loan calculations.
end	The ending month for the loan calculations.
is_annuit	A flag indicating whether the loan is annuity-based.
is_linear	A flag indicating whether the loan is linear-based.
loan_amount	The loan amount.
annual_percentage	The annual interest rate as a percentage.
years	The number of years for the loan.
months	The number of months for the loan.
start	The start delay in months.
end	The end delay in months.
is_annuit	A boolean indicating whether the loan is annuity-based.
is_linear	A boolean indicating whether the loan is linear-based.

Author

Julius Jauga

4.2.3 Member Function Documentation

4.2.3.1 createAnnuitList()

```
void Calculations::createAnnuitList ( )
```

Creates the list of month information for the annuity-based loan.

This method creates a list of monthly payment information using the annuity repayment method.

Authors

Julius Jauga

4.2.3.2 createLinearList()

```
void Calculations::createLinearList ( )
```

Creates the list of month information for the linear-based loan.

This method creates a list of monthly payment information using the linear repayment method.

Authors

Julius Jauga

4.2.3.3 createList()

```
void Calculations::createList ( )
```

Creates the list of month information for the loan.

This method selects strategy and calls method the creates a list of monthly payment information using the annuity or linear repayment method.

Authors

Julius Jauga, Aurelijus Lukšas

4.2.3.4 getAnnualPercentage()

```
double Calculations::getAnnualPercentage ( ) const
```

Gets the annual interest rate percentage.

Gets the annual interest rate as a percentage.

Returns

The annual interest rate percentage.

The annual interest rate.

Author

Julius Jauga

4.2.3.5 getIsAnnuit()

```
bool Calculations::getIsAnnuit ( ) const
```

Gets whether the loan is annuity-based.

Returns

A flag indicating whether the loan is annuity-based.

A boolean indicating whether the loan is annuity-based.

Author

Julius Jauga

4.2.3.6 getIsLinear()

```
bool Calculations::getIsLinear ( ) const
```

Gets whether the loan is linear-based.

Returns

A flag indicating whether the loan is linear-based.

A boolean indicating whether the loan is linear-based.

Author

Julius Jauga

4.2.3.7 getList()

```
std::vector< MonthInfo > Calculations::getList ( ) const
```

Gets the list of month information for the loan.

Gets the list of monthly payment information.

Returns

The list of month information.

The list of monthly payment information.

Author

Julius Jauga

4.2.3.8 getLoanAmount()

```
double Calculations::getLoanAmount ( ) const
```

Gets the loan amount.

Returns

The loan amount.

The loan amount.

Author

Julius Jauga

4.2.3.9 getMonths()

```
int Calculations::getMonths ( ) const
```

Gets the number of months for the loan.

Returns

The number of months for the loan.

The number of months.

Author

Julius Jauga

4.2.3.10 getYears()

```
int Calculations::getYears ( ) const
```

Gets the number of years for the loan.

Returns

The number of years for the loan.

The number of years.

Author

Julius Jauga

4.2.3.11 recalculate()

```
void Calculations::recalculate ( )
```

Recalculates the loan based on the current parameters.

Recalculates the list of monthly payment information.

Author

Julius Jauga

4.2.3.12 setAnnualPercentage()

```
void Calculations::setAnnualPercentage (
            double annual_percentage )
```

Sets the annual interest rate percentage.

Sets the annual interest rate as a percentage.

Parameters

percentage	The annual interest rate percentage.
annual_percentage	The new annual interest rate.

Author

Julius Jauga

4.2.3.13 setIsAnnuit()

```
void Calculations::setIsAnnuit (
            bool is_annuit )
```

Sets whether the loan is annuity-based.

Parameters

isAnnuit	A flag indicating whether the loan is annuity-based.
is_annuit	A boolean indicating whether the loan is annuity-based.

Author

Julius Jauga

4.2.3.14 setIsLinear()

```
void Calculations::setIsLinear (
            bool is_linear )
```

Sets whether the loan is linear-based.

Parameters

isLinear	A flag indicating whether the loan is linear-based.
is_linear	A boolean indicating whether the loan is linear-based.

Author

Julius Jauga

4.2.3.15 setLoanAmount()

```
void Calculations::setLoanAmount (
            double loan_amount )
```

Sets the loan amount.

Parameters

amount	The loan amount.
loan_amount	The new loan amount.

Author

Julius Jauga

4.2.3.16 setMonths()

```
void Calculations::setMonths (
            int months )
```

Sets the number of months for the loan.

Parameters

months	The number of months for the loan.
months	The new number of months.

Author

Julius Jauga

4.2.3.17 setStrategy()

```
void Calculations::setStrategy (
            ListCreationStrategy * newStrategy )
```

Sets the strategy for creating the list of month information.

Sets the strategy for creating the list of monthly payment information.

Parameters

newStrategy	The new strategy for creating the list.
newStrategy	The new strategy to be used.

Author

Julius Jauga

4.2.3.18 setYears()

```
void Calculations::setYears (
            int years )
```

Sets the number of years for the loan.

Parameters

years	The number of years for the loan.
years	The new number of years.

Author

Julius Jauga

The documentation for this class was generated from the following files:

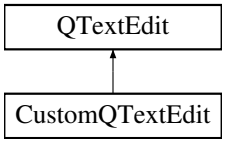
- calculations.h
- calculations.cpp

4.3 CustomQTextEdit Class Reference

Constructor for the CustomQTextEdit class.

`#include <customQTextEdit.h>`

Inheritance diagram for CustomQTextEdit:



Public Member Functions

- CustomQTextEdit (QWidget *parent=nullptr)
  Constructs a CustomQTextEdit object.

Protected Member Functions

- void keyPressEvent (QKeyEvent *event) override
  Overrides the keyPressEvent() function to handle key events.

4.3.1 Detailed Description

Constructor for the CustomQTextEdit class.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 CustomQTextEdit()

```
CustomQTextEdit::CustomQTextEdit (
            QWidget * parent = nullptr )  [explicit]
```

Constructs a CustomQTextEdit object.

Constructor for the CustomQTextEdit class.

Parameters

parent	The parent widget.

4.3.3 Member Function Documentation

4.3.3.1 keyPressEvent()

```
void CustomQTextEdit::keyPressEvent (
            QKeyEvent * event )  [override], [protected]
```

Overrides the keyPressEvent() function to handle key events.

Overrides the keyPressEvent function to handle the Tab key press event.

Parameters

event	The key event.

This function is called when a key press event occurs. If the Tab key is pressed, it ignores the event and focuses on the next child widget, allowing the user to move between input fields using the Tab key. If any other key is pressed, it calls the base class's keyPressEvent function to handle the event.

Parameters

event	The key press event.

The documentation for this class was generated from the following files:

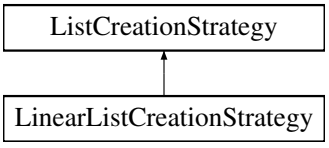
- customQTextEdit.h
- customQTextEdit.cpp

4.4 LinearListCreationStrategy Class Reference

The LinearListCreationStrategy class represents a strategy for creating a list of month information for a linear-based loan.

```
#include <calculations.h>
```

Inheritance diagram for LinearListCreationStrategy:



Public Member Functions

- **LinearListCreationStrategy** (Calculations *calc)
- void createList () override

  This method creates a list of monthly payment information using the linear repayment method.

Public Member Functions inherited from ListCreationStrategy

- **ListCreationStrategy** (Calculations *calc)

Protected Attributes

- Calculations * **calculations**

Protected Attributes inherited from ListCreationStrategy

- Calculations * **calculations**

4.4.1 Detailed Description

The LinearListCreationStrategy class represents a strategy for creating a list of month information for a linear-based loan.

Author

Aurelijus Lukšas

4.4.2 Member Function Documentation

4.4.2.1 createList()

```
void LinearListCreationStrategy::createList ( )  [override], [virtual]
```

This method creates a list of monthly payment information using the linear repayment method.

Author

Aurelijus Lukšas

Implements ListCreationStrategy.

The documentation for this class was generated from the following files:

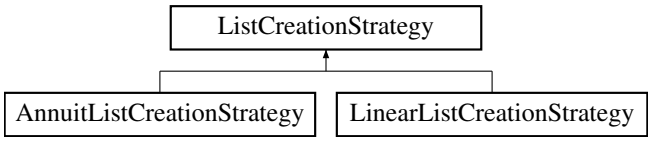
- calculations.h
- calculations.cpp

4.5 ListCreationStrategy Class Reference

Classes used for calling Annuit and Linear calculations for monthly payments.

`#include <calculations.h>`

Inheritance diagram for ListCreationStrategy:



Public Member Functions

- **ListCreationStrategy** (Calculations ∗calc)
- virtual void createList ()=0

Protected Attributes

- Calculations ∗ **calculations**

4.5.1 Detailed Description

Classes used for calling Annuit and Linear calculations for monthly payments.

Author

Aurelijus Lukšas

4.5.2 Member Function Documentation

4.5.2.1 createList()

```
virtual void ListCreationStrategy::createList ( )  [pure virtual]
```

Implemented in AnnuitListCreationStrategy, and LinearListCreationStrategy.

The documentation for this class was generated from the following file:

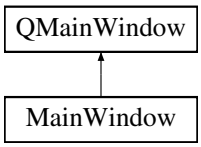
- calculations.h

4.6 MainWindow Class Reference

The MainWindow class represents the main window of the Mortgage Buddy application.

`#include <mainwindow.h>`

Inheritance diagram for MainWindow:



Public Member Functions

- MainWindow (QWidget *parent=nullptr)
  Constructs a MainWindow object.
- ~MainWindow ()
  Destroys the MainWindow object.
- void createGraph ()
  Creates the graph view for displaying mortgage data.
- void fillView (std::vector< MonthInfo > list)
  Fills the graph view with data.
- void setFilterLimits (int years, int months)
  Sets the filter limits for the mortgage data.
- void drawGraph (std::vector< MonthInfo > list)
  Draws the graph based on the mortgage data.
- void printGraphAsPDF ()
  Prints the graph as a PDF file.
- void exportToCSV (std::vector< MonthInfo > list, QString filename)
  Exports the mortgage data to a CSV file.
- void filterData ()
  Filters the mortgage data based on the set filter limits.
- void importFromCSV (QString filename)
  Imports mortgage data from a CSV file.

4.6.1 Detailed Description

The MainWindow class represents the main window of the Mortgage Buddy application.

This class inherits from QMainWindow and provides functionality for creating and managing the user interface, as well as performing various calculations and operations related to mortgage calculations.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 MainWindow()

```
MainWindow::MainWindow (
            QWidget * parent = nullptr )
```

Constructs a MainWindow object.

Constructor for the MainWindow class.

Parameters

parent	The parent widget.
parent	The parent widget of the MainWindow.

Authors

Julius Jauga, Rokas Baliutavičius

4.6.2.2 ~MainWindow()

```
MainWindow::~MainWindow ( )
```

Destroys the MainWindow object.

Destructor for the MainWindow class.

4.6.3 Member Function Documentation

4.6.3.1 createGraph()

```
void MainWindow::createGraph ( )
```

Creates the graph view for displaying mortgage data.

Creates line graph for later use.

This method initializes components needed for displaying the monthly payment line graph. This includes adding X and Y axes to the graph, setting their titles and numeration format.

Author

Rokas Baliutavičius

4.6.3.2 drawGraph()

```
void MainWindow::drawGraph (
            std::vector< MonthInfo > list )
```

Draws the graph based on the mortgage data.

Re-draws the line graph with calculated data.

Parameters

list	The list of MonthInfo objects representing the mortgage data.

This method is used for re-drawing the monthly payment line graph upon pressing "Calculate" button. The process includes adding new data to the series as well as adjusting the scale of the axes based on the new data.

Parameters

list	The vector of MonthInfo objects containing the loan payment information.

Author

Rokas Baliutavičius

4.6.3.3 exportToCSV()

```
void MainWindow::exportToCSV (
            std::vector< MonthInfo > list,
            QString filename )
```

Exports the mortgage data to a CSV file.

Parameters

list	The list of MonthInfo objects representing the mortgage data.

Author

Aurelijus Lukšas

Parameters

list	The list of MonthInfo objects representing the mortgage data.
filename	The name of the file to export data to.

4.6.3.4 fillView()

```
void MainWindow::fillView (
            std::vector< MonthInfo > list )
```

Fills the graph view with data.

Fills the view with data from a vector of MonthInfo objects.

Parameters

list	The list of MonthInfo objects representing the mortgage data.

This method clears the month_list widget and populates it with data from the provided vector. Each MonthInfo object in the vector represents a month's worth of information for a loan payment. The data includes the month number,

monthly payment, interest payment, and remaining balance.

Parameters

list	The vector of MonthInfo objects containing the loan payment information.

Author

Julius Jauga

4.6.3.5 filterData()

```
void MainWindow::filterData ( )
```

Filters the mortgage data based on the set filter limits.

Filters the data for use in graph and table.

Author

Aurelijus Lukšas

4.6.3.6 importFromCSV()

```
void MainWindow::importFromCSV (
            QString filename )
```

Imports mortgage data from a CSV file.

Author

Aurelijus Lukšas

Parameters

filename	The name of the file to import data from.

4.6.3.7 printGraphAsPDF()

```
void MainWindow::printGraphAsPDF ( )
```

Prints the graph as a PDF file.

Saves the line graph to a PDF file.

Author

Rokas Baliutavičius

4.6.3.8 setFilterLimits()

```
void MainWindow::setFilterLimits (
            int years,
            int months )
```

Sets the filter limits for the mortgage data.

Sets the limits of the filter sliders.

Parameters

years	The number of years to include in the filter.
months	The number of months to include in the filter.

Author

Aurelijus Lukšas

The documentation for this class was generated from the following files:

- mainwindow.h
- mainwindow.cpp

4.7 MonthInfo Class Reference

Represents information about a specific month in a loan calculation.

`#include <monthinfo.h>`

Public Member Functions

- MonthInfo (int month, double monthly_payment, double interest_payment, double remaining_balance)
  Constructs a MonthInfo object with the given parameters.
- MonthInfo (const MonthInfo &other)
  Copy constructor.
- MonthInfo & operator= (const MonthInfo &other)
  Assignment operator.
- void setMonth (int month)
  Sets the month number.
- void setMonthlyPayment (double monthly_payment)
  Sets the monthly payment amount.
- void setInterestPayment (double interest_payment)
  Sets the interest payment amount.
- void setRemainingBalance (double remaining_balance)
  Sets the remaining balance amount.
- int getMonth () const
  Returns the month number.
- double getMonthlyPayment () const
  Returns the monthly payment amount.
- double getInterestPayment () const
  Returns the interest payment amount.
- double getRemainingBalance () const
  Returns the remaining balance amount.

4.7.1 Detailed Description

Represents information about a specific month in a loan calculation.

The MonthInfo class stores information such as the month number, monthly payment, interest payment, and remaining balance for a specific month in a loan calculation.

Author

Julius Jauga

Julius Jauga

4.7.2 Constructor & Destructor Documentation

4.7.2.1 MonthInfo() [1/2]

```
MonthInfo::MonthInfo (
            int month,
            double monthly_payment,
            double interest_payment,
            double remaining_balance )
```

Constructs a MonthInfo object with the given parameters.

Constructs a MonthInfo object with default values.

Parameters

month	The month number.
monthly_payment	The monthly payment amount.
interest_payment	The interest payment amount.
remaining_balance	The remaining balance amount.

4.7.2.2 MonthInfo() [2/2]

```
MonthInfo::MonthInfo (
            const MonthInfo & other )
```

Copy constructor.

Constructs a MonthInfo object by copying another MonthInfo object.

Parameters

other	The MonthInfo object to be copied.

4.7.3 Member Function Documentation

4.7.3.1 getInterestPayment()

```
double MonthInfo::getInterestPayment ( ) const
```

Returns the interest payment amount.

Gets the interest payment amount.

Returns

The interest payment amount.

4.7.3.2 getMonth()

```
int MonthInfo::getMonth ( ) const
```

Returns the month number.

Gets the month number.

Returns

The month number.

4.7.3.3 getMonthlyPayment()

```
double MonthInfo::getMonthlyPayment ( ) const
```

Returns the monthly payment amount.

Gets the monthly payment amount.

Returns

The monthly payment amount.

4.7.3.4 getRemainingBalance()

```
double MonthInfo::getRemainingBalance ( ) const
```

Returns the remaining balance amount.

Gets the remaining balance amount.

Returns

The remaining balance amount.

4.7.3.5 operator=()

```
MonthInfo & MonthInfo::operator= (
            const MonthInfo & other )
```

Assignment operator.

Assigns the values of another MonthInfo object to this object.

Parameters

other	The MonthInfo object to be assigned.

Returns

A reference to the assigned MonthInfo object.

Parameters

other	The MonthInfo object to be assigned.

Returns

A reference to this MonthInfo object.

4.7.3.6 setInterestPayment()

```
void MonthInfo::setInterestPayment (
            double newInterestPayment )
```

Sets the interest payment amount.

Parameters

interest_payment	The interest payment amount.
newInterestPayment	The new interest payment amount.

4.7.3.7 setMonth()

```
void MonthInfo::setMonth (
            int newMonth )
```

Sets the month number.

Parameters

month	The month number.
newMonth	The new month number.

4.7.3.8 setMonthlyPayment()

```
void MonthInfo::setMonthlyPayment (
            double newMonthlyPayment )
```

Sets the monthly payment amount.

Parameters

monthly_payment	The monthly payment amount.
newMonthlyPayment	The new monthly payment amount.

4.7.3.9 setRemainingBalance()

```
void MonthInfo::setRemainingBalance (
            double newRemainingBalance )
```

Sets the remaining balance amount.

Parameters

remaining_balance	The remaining balance amount.
newRemainingBalance	The new remaining balance amount.

The documentation for this class was generated from the following files:

- monthinfo.h
- monthinfo.cpp

Chapter 5

File Documentation

5.1 calculations.h

```
00001
00002 #ifndef CALCULATIONS_H
00003 #define CALCULATIONS_H
00004
00005 #include "monthinfo.h"
00006 #include <vector>
00007
00008 class Calculations;
00009
00016 class ListCreationStrategy {
00017     protected:
00018         Calculations* calculations;
00019     public:
00020         ListCreationStrategy() {};
00021         ListCreationStrategy(Calculations* calc) : calculations(calc) {}
00022         virtual void createList() = 0;
00023 };
00024
00030 class AnnuitListCreationStrategy : public ListCreationStrategy {
00031     protected:
00032         Calculations* calculations;
00033     public:
00034         AnnuitListCreationStrategy(Calculations* calc) : calculations(calc) {}
00035         void createList() override;
00036 };
00041 class LinearListCreationStrategy : public ListCreationStrategy {
00042     protected:
00043         Calculations* calculations;
00044     public:
00045         LinearListCreationStrategy(Calculations* calc) : calculations(calc) {}
00046         void createList() override;
00047 };
00048
00049
00053 class Calculations
00054 {
00055 public:
00059     Calculations();
00060
00072     Calculations(double loan_amount, double annual_percentage, int years, int months, int start, int
      end, bool is_annuit, bool is_linear);
00073
00077     void recalculate();
00078
00083     void setLoanAmount(double amount);
00084
00089     void setAnnualPercentage(double percentage);
00090
00095     void setYears(int years);
00096
00101     void setMonths(int months);
00102
00107     void setIsAnnuit(bool isAnnuit);
00108
00113     void setIsLinear(bool isLinear);
00114
00119     double getLoanAmount() const;
00120
```

```
00125     double getAnnualPercentage() const;
00126
00131     int getYears() const;
00132
00137     int getMonths() const;
00138
00143     bool getIsAnnuit() const;
00144
00149     bool getIsLinear() const;
00150
00155     std::vector<MonthInfo> getList() const;
00156
00160     void createList();
00161
00165     void createAnnuitList();
00166
00170     void createLinearList();
00171
00176     void setStrategy(ListCreationStrategy* newStrategy);
00177
00178 private:
00179     ListCreationStrategy* strategy;
00180     double loan_amount;
00181     double annual_percentage;
00182     int years;
00183     int months;
00184     bool is_annuit;
00185     bool is_linear;
00186     int delay_start;
00187     int delay_end;
00188     std::vector<MonthInfo> month_list;
00189 };
00190
00191 #endif // CALCULATIONS_H
```

5.2 customQTextEdit.cpp File Reference

Implementation file for the CustomQTextEdit class.

```
#include "customQTextEdit.h"
#include <QKeyEvent>
```

5.2.1 Detailed Description

Implementation file for the CustomQTextEdit class.

This file contains the implementation of the CustomQTextEdit class, which is a custom subclass of QTextEdit, meant to allow movement between input fields using a tab press.

Author

Rokas Baliutavičius

5.3 customQTextEdit.h File Reference

Header file for the CustomQTextEdit class.

```
#include <QTextEdit>
#include <QPushButton>
```

Classes

- class CustomQTextEdit

  Constructor for the CustomQTextEdit class.

5.3.1 Detailed Description

Header file for the CustomQTextEdit class.

This file contains the definition of the CustomQTextEdit class, which is a custom subclass of QTextEdit, meant to allow movement between input fields using a tab press.

Author

Rokas Baliutavičius

5.4 customQTextEdit.h

Go to the documentation of this file.

```
00001
00011 #ifndef CUSTOMQTEXTEDIT_H
00012 #define CUSTOMQTEXTEDIT_H
00013
00014 #include <QTextEdit>
00015 #include <QPushButton>
00016
00020 class CustomQTextEdit : public QTextEdit
00021 {
00022     Q_OBJECT
00023
00024 public:
00030     explicit CustomQTextEdit(QWidget *parent = nullptr);
00031
00032 protected:
00038     void keyPressEvent(QKeyEvent *event) override;
00039 };
00040
00041 #endif // CUSTOM_WIDGETS_H
```

5.5 mainwindow.cpp File Reference

Implementation of the MainWindow class.

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "customQTextEdit.h"
#include <QTreeWidget>
#include <QTreeWidgetItem>
#include <QtWidgets>
#include <QtCharts>
#include <QPdfWriter>
#include <QPainter>
#include <QBrush>
#include <QFont>
#include <QColor>
#include <algorithm>
#include "calculations.h"
```

Functions

- void clearTreeWidget (QTreeWidget *treeWidget)

  Clears the items in a QTreeWidget.
- void swapWidgetItems (QTreeWidget *treeWidget, int firstIndex, int secondIndex)

  Swaps the items at the specified indices in a QTreeWidget.

Variables

- QLineSeries * **series**

  Line series for the graph.
- QChart * **chart**

  Chart for the graph.
- QValueAxis * **axisX**

  X axis for the graph.
- QValueAxis * **axisY**

  Y axis for the graph.
- QChartView * **chartView**

  Chart view for the graph.
- bool **clickedFlag** = 0

  Flag to check if the calculate button was clicked.

5.5.1 Detailed Description

Implementation of the MainWindow class.

This file contains the implementation of the MainWindow class, which is the main window of the application. The class is responsible for handling user input, displaying the loan payment information, and drawing the monthly payment line graph. The class also provides functionality for exporting the loan payment information to a CSV file and importing it back from a CSV file. The class is part of the Loan Calculator application.

Version

1.0

Date

2021-05-23

See also

MainWindow

mainwindow.h

calculations.h

customQTextEdit.h

main.cpp

calculations.cpp

customQTextEdit.cpp

Authors

Julius Jauga, Rokas Baliutavičius, Aurelijus Lukšas

5.5.2 Function Documentation

5.5.2.1 clearTreeWidget()

```
void clearTreeWidget (
            QTreeWidget * treeWidget )
```

Clears the items in a QTreeWidget.

This function removes all items from the specified QTreeWidget.

Parameters

treeWidget	The QTreeWidget to be cleared.

Author

Julius Jauga

5.5.2.2 swapWidgetItems()

```
void swapWidgetItems (
            QTreeWidget * treeWidget,
            int firstIndex,
            int secondIndex )
```

Swaps the items at the specified indices in a QTreeWidget.

This function swaps the items at the given indices in the specified QTreeWidget. If both indices are valid and the items exist, the function removes the items from their original positions and inserts them at each other's positions.

Parameters

treeWidget	The QTreeWidget in which the items are to be swapped.
firstIndex	The index of the first item to be swapped.
secondIndex	The index of the second item to be swapped.

Author

Julius Jauga

5.6 mainwindow.h File Reference

This file contains the declaration of the MainWindow class.

```
#include <QMainWindow>
#include <QTCharts>
#include <QChartView>
```

```
#include <QLineSeries>
#include "monthinfo.h"
#include "calculations.h"
#include <vector>
```

Classes

- class MainWindow

  The MainWindow class represents the main window of the Mortgage Buddy application.

5.6.1 Detailed Description

This file contains the declaration of the MainWindow class.

Authors

Julius Jauga, Rokas Baliutavičius, Aurelijus Lukšas

5.7 mainwindow.h

Go to the documentation of this file.

```
00001
00007 #ifndef MAINWINDOW_H
00008 #define MAINWINDOW_H
00009
00010 #include <QMainWindow>
00011 #include <QTCharts>
00012 #include <QChartView>
00013 #include <QLineSeries>
00014 #include "monthinfo.h"
00015 #include "calculations.h"
00016 #include <vector>
00017
00018 QT_BEGIN_NAMESPACE
00019 namespace Ui {
00020 class MainWindow;
00021 }
00022 QT_END_NAMESPACE
00023
00030 class MainWindow : public QMainWindow
00031 {
00032     Q_OBJECT
00033
00034 public:
00040     MainWindow(QWidget *parent = nullptr);
00041
00045     ~MainWindow();
00046
00050     void createGraph();
00051
00057     void fillView(std::vector<MonthInfo> list);
00058
00065     void setFilterLimits(int years, int months);
00066
00072     void drawGraph(std::vector<MonthInfo> list);
00073
00077     void printGraphAsPDF();
00078
00084     void exportToCSV(std::vector<MonthInfo> list, QString filename);
00085
00089     void filterData();
00090
00094     void importFromCSV(QString filename);
00095
00096 private slots:
00100     void on_calculate_button_clicked();
00101
```

```
00105     void on_annuit_box_stateChanged();
00106
00110     void on_linear_box_stateChanged();
00111
00115     void on_saveChartPDF_clicked();
00116
00120     void on_exportToCSVButton_clicked();
00121
00125     void on_importFromCSVButton_clicked();
00126
00133     void on_month_list_itemDoubleClicked(QTreeWidgetItem *item, int column);
00134
00135 private:
00136     ListCreationStrategy* strategy;
00137     AnnuitListCreationStrategy* annuitStrategy;
00138     LinearListCreationStrategy* linearStrategy;
00139     Ui::MainWindow *ui;
00140     std::vector<int> addedMonths;
00141     double loan_amount;
00142     double annual_percent;
00143     int years;
00144     int months;
00145     int delay_start;
00146     int delay_end;
00147     int filter_start;
00148     int filter_end;
00149     bool is_annuit;
00150     bool is_linear;
00151
00157     int getData();
00158 };
00159
00160 #endif // MAINWINDOW_H
```

5.8 monthinfo.cpp File Reference

```
#include "monthinfo.h"
```

5.9 monthinfo.h File Reference

Classes

- class MonthInfo

  Represents information about a specific month in a loan calculation.

5.10 monthinfo.h

Go to the documentation of this file.

```
00001 #ifndef MONTHINFO_H
00002 #define MONTHINFO_H
00003
00011 class MonthInfo {
00012 public:
00020     MonthInfo(int month, double monthly_payment, double interest_payment, double remaining_balance);
00021
00026     MonthInfo(const MonthInfo &other);
00027
00033     MonthInfo& operator=(const MonthInfo &other);
00034
00039     void setMonth(int month);
00040
00045     void setMonthlyPayment(double monthly_payment);
00046
00051     void setInterestPayment(double interest_payment);
00052
00057     void setRemainingBalance(double remaining_balance);
```

```
00058
00063     int getMonth() const;
00064
00069     double getMonthlyPayment() const;
00070
00075     double getInterestPayment() const;
00076
00081     double getRemainingBalance() const;
00082
00083 private:
00084     int month;
00085     double monthly_payment;
00086     double interest_payment;
00087     double remaining_balance;
00088 };
00089
00090 #endif
```