МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра ИС

ОТЧЕТ

по практической работе №6

по дисциплине «Объектно-ориентированное программирование»

Студент гр. 0374 _____ Багаев Д.А
Басин И.Д.

Преподаватель _____ Егоров С. С.

Санкт-Петербург

2023

1. **Задание на практическую работу**

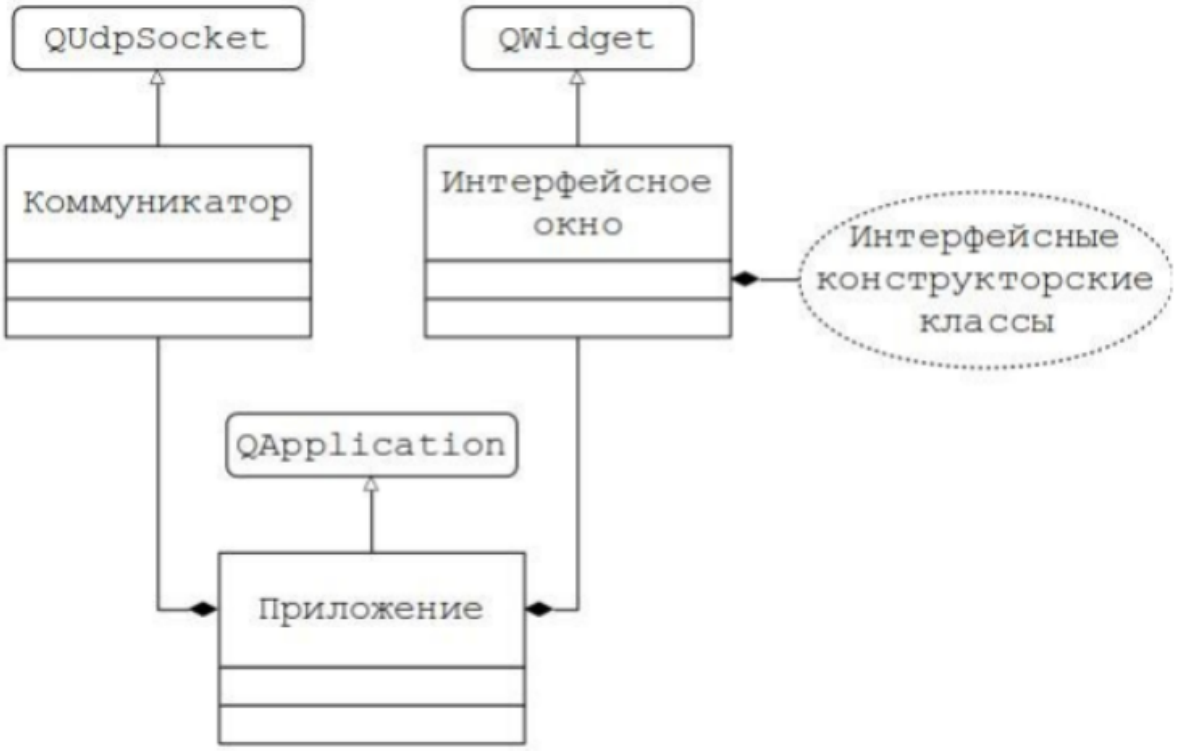

Рис.1. Диаграмма классов работы №6 (клиентская часть)

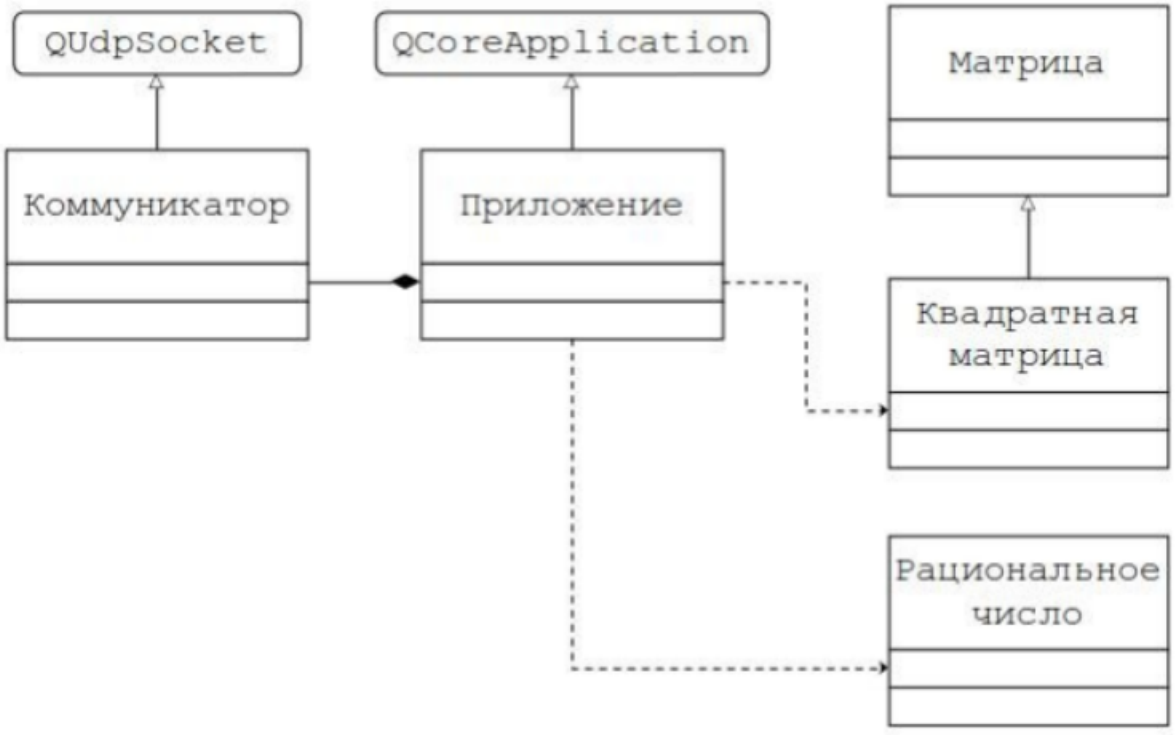

Рис.2. Диаграмма классов работы №6 (серверная часть)

Создать распределенное приложение, включающее клиентскую и серверную части, взаимодействующие посредством сетевого обмена сообщениями.

Клиентские и серверные части представляют собой приложения, реализованные в работе №5.

Клиентская часть модифицируется таким образом, что реализованные функции матриц могут исполняться по желанию пользователя на областях определения: вещественная, комплексная и рациональная.

Отличие серверной части заключается в том, что классы «Матрица» и «Квадратная матрица» параметризуются. Параметром класса делается абстрактный тип number, при этом файл number.h исключается из серверного приложения

Реализовать и отладить программу, удовлетворяющую сформулированным требованиям и заявленным целям. Разработать контрольные примеры и оттестировать на них программу. Оформить отчет, сделать выводы по работе.

2. Спецификации классов.

Класс **TInterface** содержит такие методы как:

- Конструктор класса по-умолчанию **TInterface() -** нужен для создания объекта класса и инициализирование атрибутов значениями по-умолчанию.
- Слот **FindTransposed()** - соединен с сигналом pressed() у кнопки **_ButtonFindTransposed,** чтобы сформировать запроса на серверную часть.
- Слот **FindRank()** - соединен с сигналом pressed() у кнопки **_ButtonFindRank,** чтобы сформировать запроса на серверную часть.
- Слот **FindTransposed()** - соединен с сигналом pressed() у кнопки **_ButtonFindTransposed,** чтобы сформировать запроса на серверную часть.

- Слот **PrintMatrix()** - соединен с сигналом pressed() у кнопки **_ButtonPrintMatrix,** чтобы сформировать запроса на серверную часть.
- Слот **FindDeterminant()** - соединен с сигналом pressed() у кнопки **_ButtonFindDeterminant,** чтобы сформировать запроса на серверную часть.

Класс **TInterface** содержит атрибуты:

- Атрибут **_TextEditInput** типа **QTextEdit*** был создан для возможности ввода матрицы от пользователя. Был объявлен private, так как работа с этим атрибутом должна происходить только внутри класса.
- Атрибут **_TextEditOutput** типа **QTextEdit*** был создан для возможности вывода матриц или числовых значений пользователю. Был объявлен private, так как работа с этим атрибутом должна происходить только внутри класса.
- Атрибут **_LabelInput** типа **QLabel*** был создан для подписи основных элементов на экране. Был объявлен private, так как работа с этим атрибутом должна происходить только внутри класса.
- Атрибут **_LabelOutput** типа **QLabel*** был создан для подписи основных элементов на экране. Был объявлен private, так как работа с этим атрибутом должна происходить только внутри класса.
- Атрибут **_ButtonFindTransposed** типа **QPushButton*** был создан для реализации кнопок меню, чтобы обеспечить взаимодействие пользователя с серверной частью. Был объявлен private, так как работа с этим атрибутом должна происходить только внутри класса. При нажатии активирует слот **FindTransposed().**
- Атрибут **_ButtonFindRank** типа **QPushButton*** был создан для реализации кнопок меню, чтобы обеспечить взаимодействие пользователя с серверной частью. Был объявлен private, так как работа с этим атрибутом должна происходить только внутри класса. При нажатии активирует слот **FindRank().**
- Атрибут **_ButtonPrintMatrix** типа **QPushButton*** был создан для реализации кнопок меню, чтобы обеспечить

взаимодействие пользователя с серверной частью. Был объявлен private, так как работа с этим атрибутом должна происходить только внутри класса. При нажатии активирует слот **PrintMatrix().**

- Атрибут **_ButtonFindDeterminant** типа **QPushButton*** был создан для реализации кнопок меню, чтобы обеспечить взаимодействие пользователя с серверной частью. Был объявлен private, так как работа с этим атрибутом должна происходить только внутри класса. При нажатии активирует слот **FindDeterminant().**
- Атрибут **_RadioButtonReal** типа **QRadioButton*** был создан для выбора типа вещественного числа. Был объявлен private, так как работа с этим атрибутом должна происходить только внутри класса.
- Атрибут **_RadioButtonComplex** типа **QRadioButton*** был создан для выбора типа комплексного числа. Был объявлен private, так как работа с этим атрибутом должна происходить только внутри класса.
- Атрибут **_RadioButtonRational** типа **QRadioButton*** был создан для выбора типа рационального числа. Был объявлен private, так как работа с этим атрибутом должна происходить только внутри класса.
- QRadioButton* _RadioButtonReal;
- QRadioButton* _RadioButtonComplex;
- QRadioButton* _RadioButtonRational;

Класс **TMatrix** содержит методы:

- Конструктор класса по-умолчанию **TMatrix() -** нужен для создания объекта класса и инициализирование атрибутов значениями по-умолчанию. По умолчанию матрица является нулевой, размерностью 1x1.
- Конструктор класса **TMatrix(TMatrix matrix) -** нужен для создания матрицы и инициализирование атрибутов значениями на основе другой матрицы.
- Конструктор класса **TMatrix(int sizeH, int sizeW) -** нужен для создания нулевой матрицы с заданными размерами.

- Деструктор **~TMatrix()** служит для освобождения динамической памяти, при удалении объекта, чтобы избежать утечки памяти.
- Метод **EnterMatrix(number** matrix, int sizeH, int sizeW)** служит для ввода/замены содержимого матрицы. Была объявлена public, так как данный метод используется в классе **TApplication.**
- Метод **DeleteMemoryMatrix()** служит для освобождения динамической памяти, которая выделяется под квадратную матрицу. Объявлен private, так как должен использоваться только внутри класса деструктором и методом **EnterMatrix**.
- Перегрузка оператора вывода **friend ostream& operator <<(ostream&, const TMatrix)** используется для реализации возможности вывода матриц в поток.
- Метод **FindTransposed()** нужен для поиска транспонированной матрицы. В динамической памяти создается матрица, которая является транспонированной матрицей для основной матрицы, и передается по указателям. В результате возвращает указатель на матрицу в динамической памяти.

Шаблонный класс **TMatrix<T>** содержит атрибуты:

- Атрибут **_Matrix** типа **T**** служит для хранения указателя на двумерный динамический массив. Был объявлен **protected**, так как работа с этим атрибутом должна происходить только внутри класса и атрибут должен быть доступен для дочерних классов.
- Атрибут **_SizeH** типа **int** служит для хранения высоты матрицы. Был объявлен **protected**, так как работа с этим атрибутом должна происходить только внутри класса и атрибут должен быть доступен для дочерних классов.
- Атрибут **_SizeW** типа **int** служит для хранения ширины матрицы. Был объявлен **protected**, так как работа с этим атрибутом должна происходить только внутри класса и атрибут должен быть доступен для дочерних классов.

Шаблонный класс **TSquareMatrix<T>** наследуется от **TMatrix** и содержит методы:

- Конструктор класса по-умолчанию **TSquareMatrix () -** нужен для создания объекта класса и инициализирование атрибутов значениями по-умолчанию. По умолчанию матрица является нулевой, размерностью 1x1.
- Конструктор класса **TSquareMatrix (TSquareMatrix matrix) -** нужен для создания квадратной матрицы и инициализирование атрибутов значениями на основе другой квадратной матрицы.
- Метод **EnterMatrix(T** matrix, int size)** служит для ввода/замены содержимого квадратной матрицы. Была объявлена public, так как данный метод используется в классе **TApplication.**
- Метод **FindDeterminant()** класса **TMatrix** используется для поиска определителя матрицы. Объявлен public для возможности использования этого метода вне класса **TMatrix**, в данном случае в классе **TApplication.** Результатом работы данного метода является число типа **double.**
- Метод **FindRank()** используется для поиска ранга матрицы. Результатом работы данного метода является число типа **int.** Метод объявлен public, так как используется в классе **TApplication.**

Класс **TComplex** содержит методы:

- Конструктор класса по-умолчанию **TComplex()** - нужен для создания объекта класса и инициализирование атрибутов значениями по-умолчанию. По умолчанию комплексное число равняется (0+0i).
- Конструктор класса **TComplex(double realNumber, double imaginaryUnit)** используется для создания объекта класса и заполнения действительной и мнимой частей.
- Конструктор класса **TComplex(double num**) используется для создания объекта класса и заполнения только действительной части. Мнимая часть будет равняться нулю.
- Перегрузка оператора присваивания **TComplex& operator=(const TComplex &other)** для присваивания значений другого комплексного числа.

- Перегрузка оператора присваивания **TComplex& operator=(const double &other)** для присваивания значений вещественного числа.
- Перегрузка оператора умножения **friend TComplex operator*(const TComplex &first, const TComplex &second)** для реализации умножения двух комплексных чисел.
- Перегрузка оператора умножения **friend TComplex operator*(const double &first, const TComplex &second)** для реализации умножения вещественного на комплексное число.
- Перегрузка оператора сложения **friend TComplex operator+(const TComplex &first, const TComplex &second)** для реализации сложения двух комплексных чисел.
- Перегрузка оператора вычитания **friend TComplex operator-(const TComplex &first, const TComplex &second)** для реализации вычитания у двух комплексных чисел.
- Перегрузка оператора деления **friend TComplex operator/(const TComplex &first, const TComplex &second)** для реализации вычитания у двух комплексных чисел.
- Перегрузка оператора вывода **friend ostream& operator<< (ostream& out, const TComplex& complexNum)** используется для реализации возможности вывода комплексного числа в поток.
- Перегрузка оператора ввода **friend istream& operator>> (istream &in, TComplex &complexNum)** используется для реализации возможности ввода комплексного числа из потока.
- Перегрузка функции **abs friend double abs(TComplex &complexNum)** для получения моделя комплексного числа.

Класс **TComplex** содержит атрибуты:

- Атрибут **_RealNumber** типа **double** служит для хранения действительной части комплексного числа. Был объявлен private, так как работа с этим атрибутом должна происходить только внутри класса.
- Атрибут **_ImaginaryUnit** типа **double** служит для хранения мнимой части комплексного числа. Был объявлен private, так как работа с этим атрибутом должна происходить только внутри класса.

Класс **TRationalNumber** содержит методы:

- Конструктор класса по-умолчанию **TRationalNumber() -** нужен для создания объекта класса и инициализирование атрибутов значениями по-умолчанию. По умолчанию рациональная дробь равняется единице.
- Конструктор класса **TRationalNumber(int, int)** используется для создания объекта класса и заполнения числителя и знаменателя.
- Конструктор класса **TRationalNumber(int)** используется для создания объекта класса и заполнения только числителя. Знаменатель будет равняться единице.
- Перегрузка оператора присваивания **TRationalNumber& operator=(const TRationalNumber &other)** для присваивания значений другой рациональной дроби.
- Перегрузка оператора присваивания **TRationalNumber & operator=(const double &other)** для присваивания значений вещественного числа.
- Перегрузка оператора умножения **friend TRationalNumber operator*(const TRationalNumber &first, const TRationalNumber &second)** для реализации умножения двух рациональных дробей.
- Перегрузка оператора умножения **friend TRationalNumber operator*(const double &first, const TRationalNumber &second)** для реализации умножения вещественного на рациональную дробь.
- Перегрузка оператора сложения **friend TRationalNumber operator+(const TRationalNumber &first, const TRationalNumber &second)** для реализации сложения двух рациональных дробей.
- Перегрузка оператора вычитания **friend TRationalNumber operator-(const TRationalNumber &first, const TRationalNumber &second)** для реализации вычитания у двух рациональных дробей.
- Перегрузка оператора деления **friend TRationalNumber operator/(const TRationalNumber &first, const**

TRationalNumber &second) для реализации вычитания у двух рациональных дробей.

- Перегрузка оператора вывода **friend ostream& operator<< (ostream& out, const TRationalNumber&)** используется для реализации возможности вывода рациональных дробей в поток.
- Перегрузка оператора ввода **friend istream& operator>> (istream &in, TRationalNumber&)** используется для реализации возможности ввода рациональных дробей из потока.
- Перегрузка функции abs **friend double abs(TRationalNumber &)** для получения модуля у рациональных дробей
- Метод **MakeIrreducible()**, реализован для сокращения дроби до рациональной. Действует на основе функции нахождения НОД.
- Метод **ToDouble()**, реализован для приведения рациональной дроби в вещественное число.

Класс **TRationalNumber** содержит атрибуты:

- Атрибут **_Numerator** типа **int** служит для хранения числителя рациональной дроби. Был объявлен private, так как работа с этим атрибутом должна происходить только внутри класса.
- Атрибут **_Denominator** типа **unsigned int** служит для хранения знаменателя рациональной дроби. Был объявлен private, так как работа с этим атрибутом должна происходить только внутри класса.

Класс **TApplication** на клиентской части содержит атрибуты:

- Атрибут **comm*** типа **TCommunicator** служит для реализации взаимодействия посредством сетевого обмена сообщениями между клиентской и серверной частями.
- Атрибут **interface** типа **TInterface*** служит для хранения указателя на интерфейс. Был объявлен private, так как работа с этим атрибутом должна происходить только внутри класса.

Класс **TApplication** на клиентской части содержит методы:

- Конструктор класса по-умолчанию **TApplication() -** нужен для создания объекта класса и инициализирование атрибутов значениями по-умолчанию.
- Слот **fromCommunicator(QByteArray)** - соединен с сигналом **recieved(QByteArray)** у **TCommunicator* comm,** чтобы среагировать и обработать сообщение от серверной части.
- Слот **toCommunicator(QString)** - соединен с сигналом **request(QString)** у **TInterface* interface,** чтобы отправить сформированное сообщение на серверную часть.

Класс **TApplication** на серверной части содержит атрибуты:

- Атрибут **comm*** типа **TCommunicator** служит для реализации взаимодействия посредством сетевого обмена сообщениями между клиентской и серверной частями.

Класс **TApplication** на серверной части содержит методы:

- Конструктор класса по-умолчанию **TApplication() -** нужен для создания объекта класса и инициализирование атрибутов значениями по-умолчанию.
- Слот **recieve(QByteArray)** - соединен с сигналом **recieved(QByteArray)** у **TCommunicator* comm,** чтобы среагировать, обработать сообщение от клиентской части и отправить ответ.

3. Диаграмма классов.

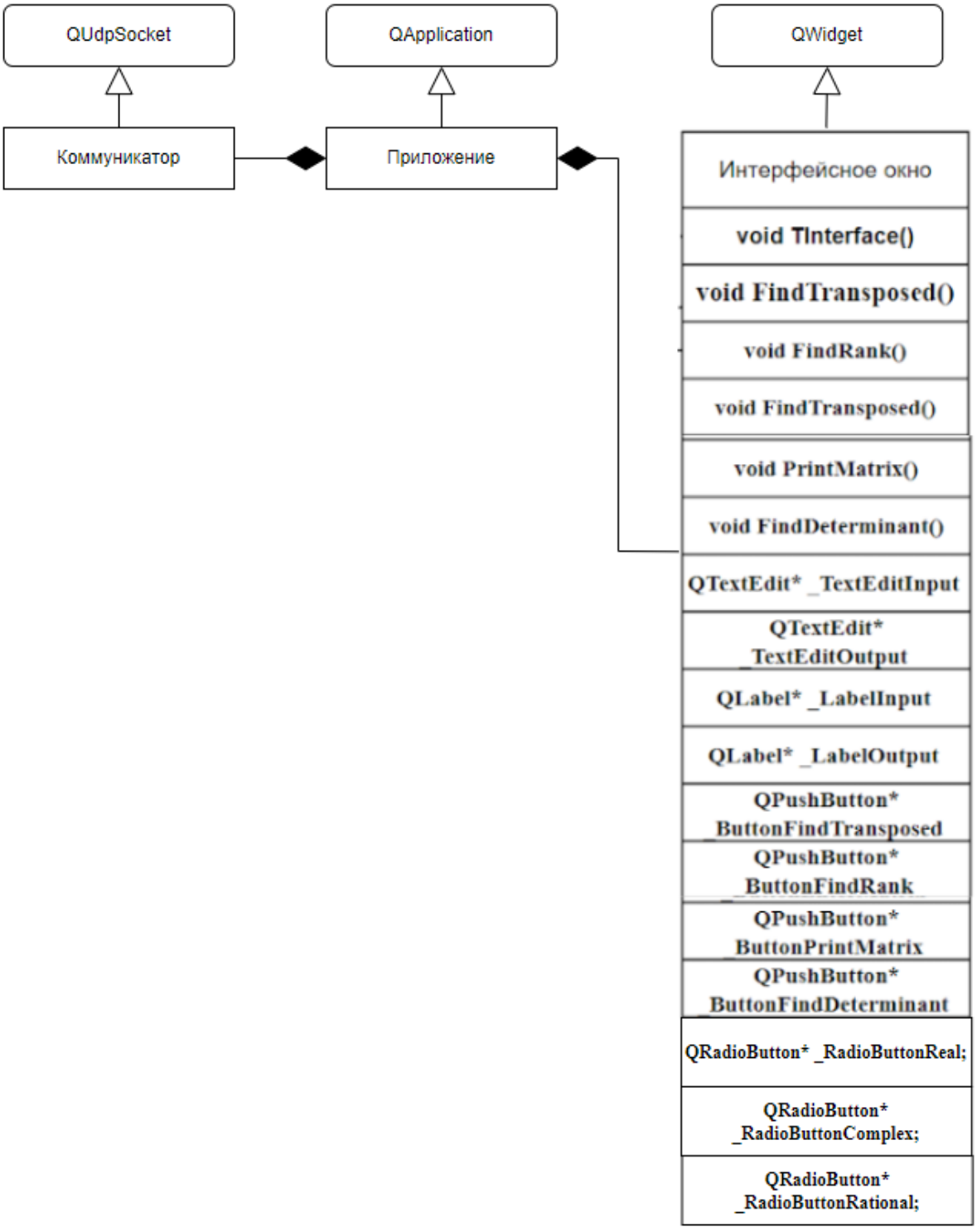

Рис. 3 - Диаграмма классов (Клиентская часть).

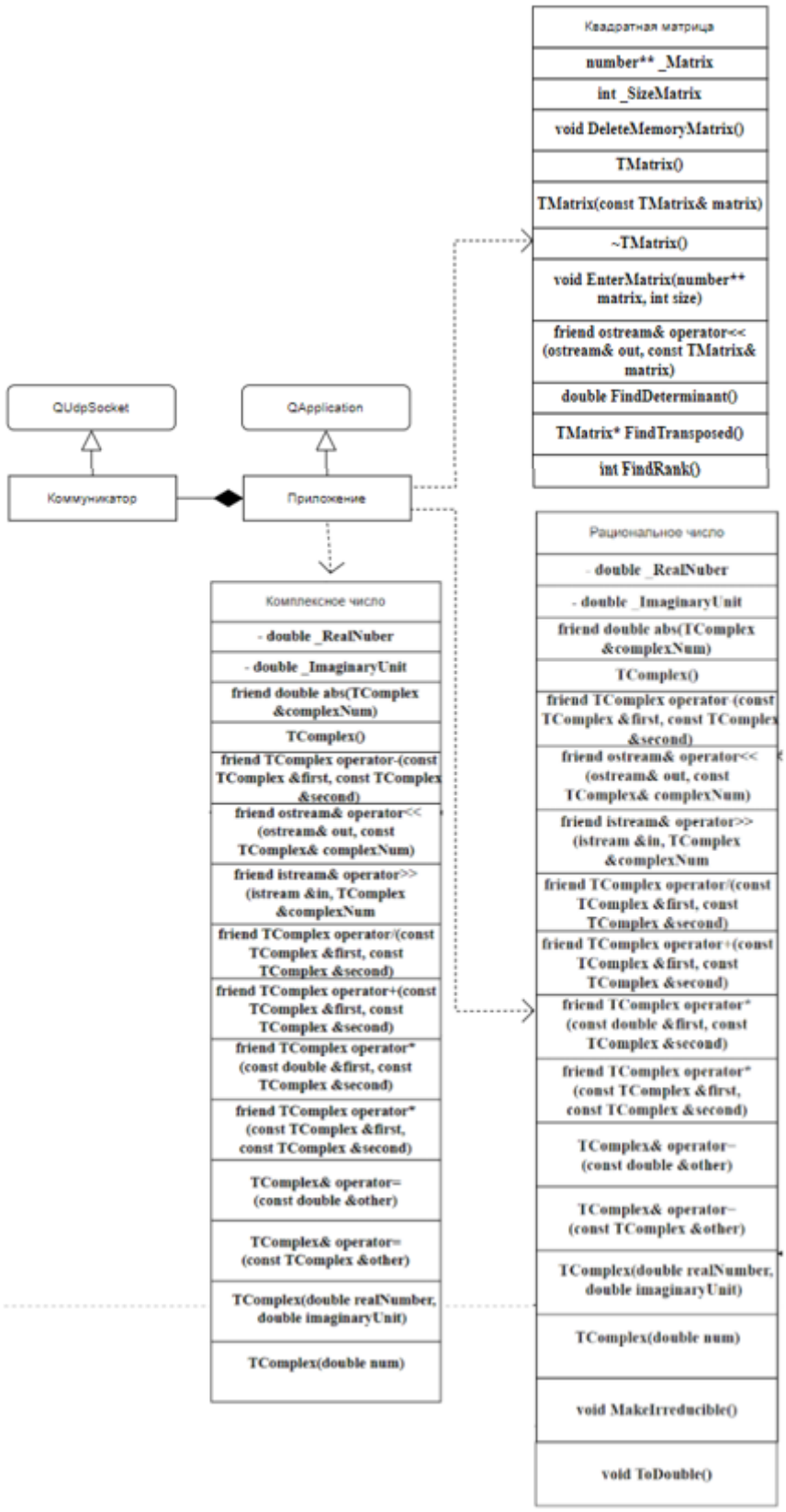

Рис. 4 - Диаграмма классов (Серверная часть).

4. Описание контрольного примера.

Тест 1

$$A = \begin{matrix} -1.7+8i & 3+11.3i & -8.2-2.15i \\ 6.3+8.1i & 3-4i & -1-6.25i \\ 5.71+8i & 1-2i & 3+3i \end{matrix}$$

$$A^T = \begin{matrix} -1.7+8i & 6.3+8.1i & 5.71+8i \\ 3+11.3i & 3-4i & 1-2i \\ -8.2-2.15i & -1-6.25i & 3+3i \end{matrix}$$

Определитель A = (1268.45 + 678.376i)

Ранг матрицы = 3

Тест 2

$$A = \begin{matrix} 1/3 & 3/8 & 12/3 \\ 27/12 & 11/99 & 0/14 \\ 16/16 & 3/2 & 2/4 \end{matrix}$$

Вывести матрицу:

$$A = \begin{matrix} 1/3 & 3/8 & 4 \\ 9/4 & 1/9 & 0 \\ 1 & 3/2 & 1/2 \end{matrix}$$

Транспонированная матрица:

$$A^T = \begin{matrix} 1/3 & 9/4 & 1 \\ 3/8 & 1/9 & 3/2 \\ 4 & 0 & 1/2 \end{matrix}$$

Определитель A = 12

Ранг матрицы = 3

5. Скриншоты работы программы на контрольных примерах.

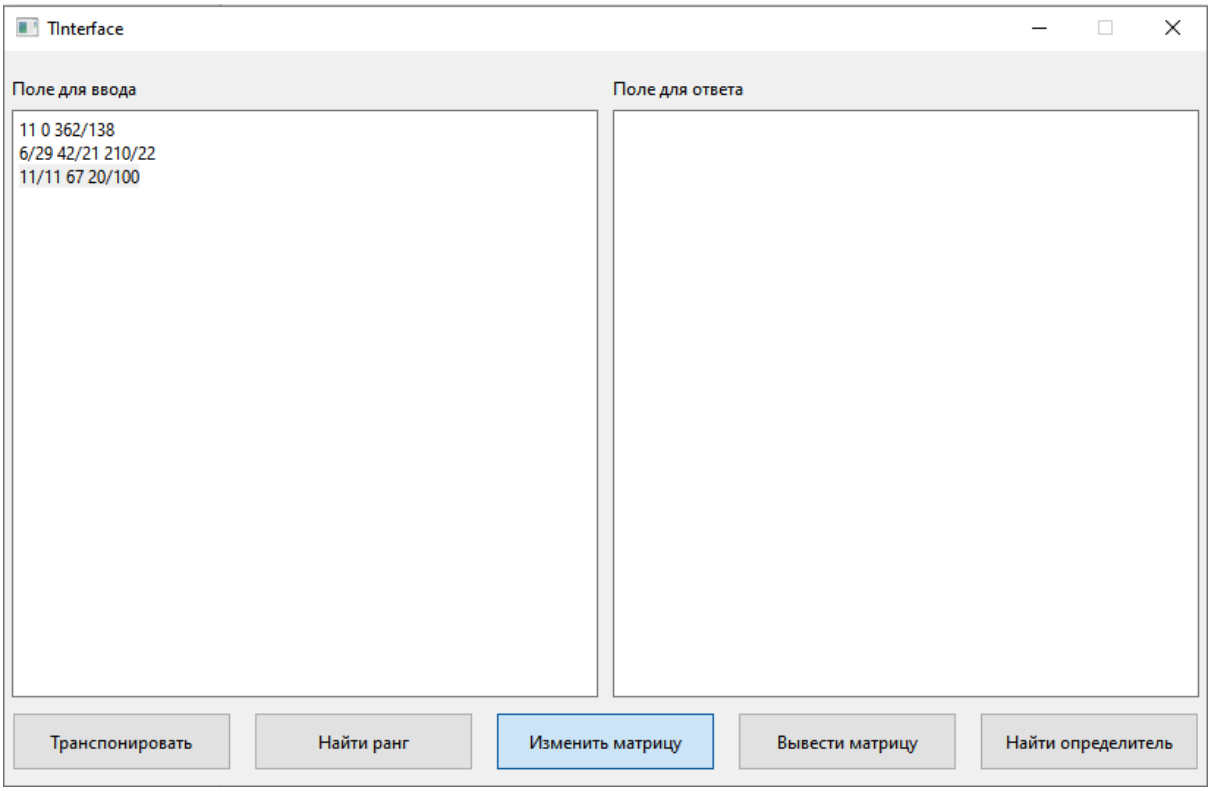

Рис. 5 - Тест 1.

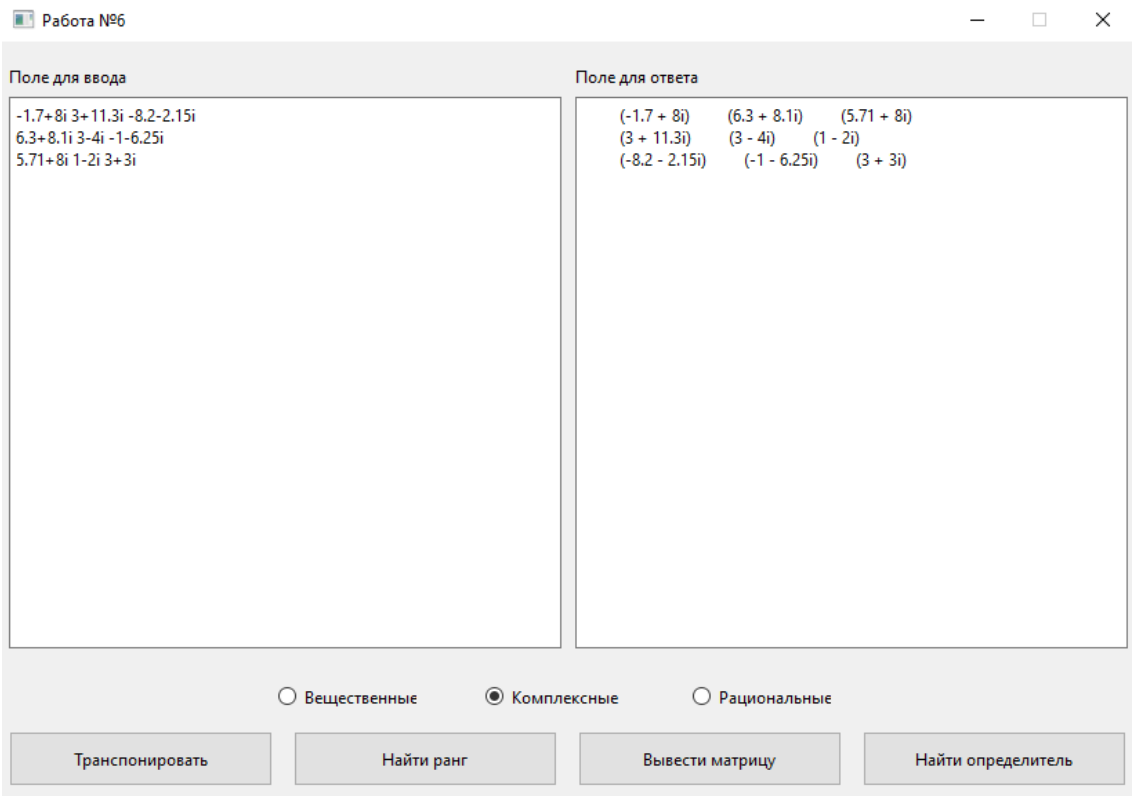

Рис. 6 - Тест 1.

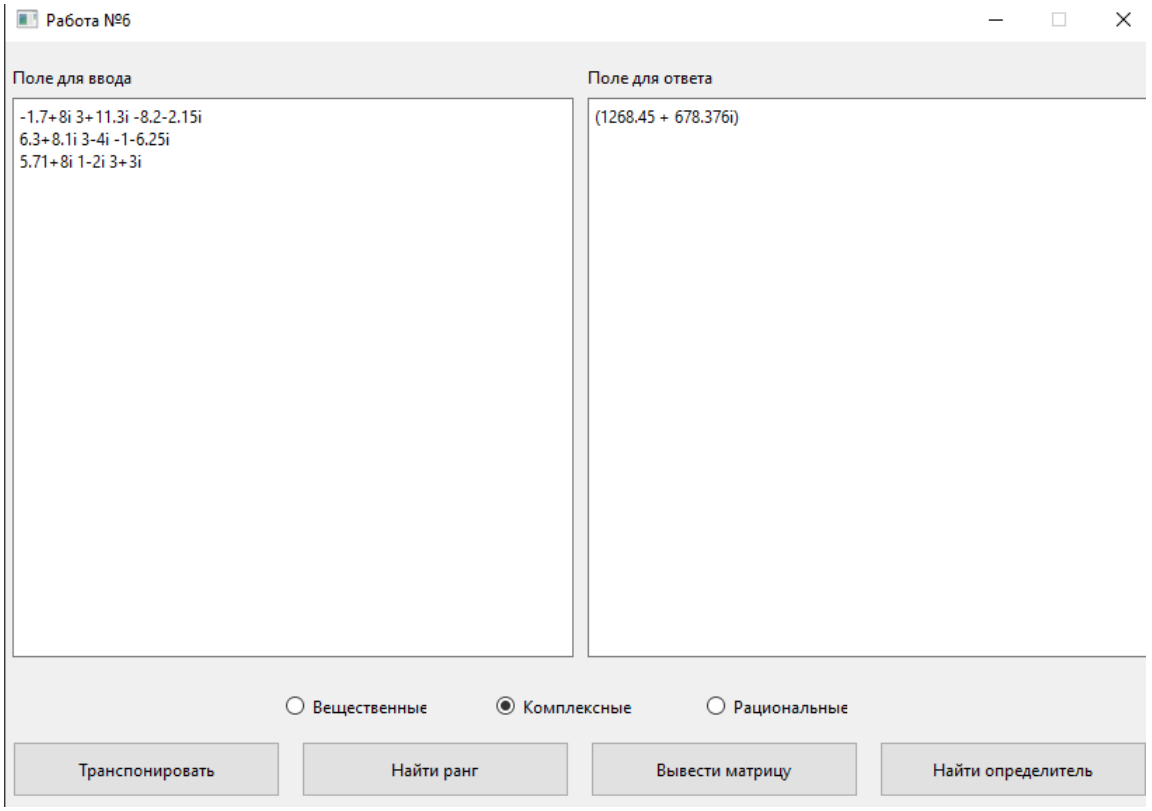

Рис. 7 - Тест 1.

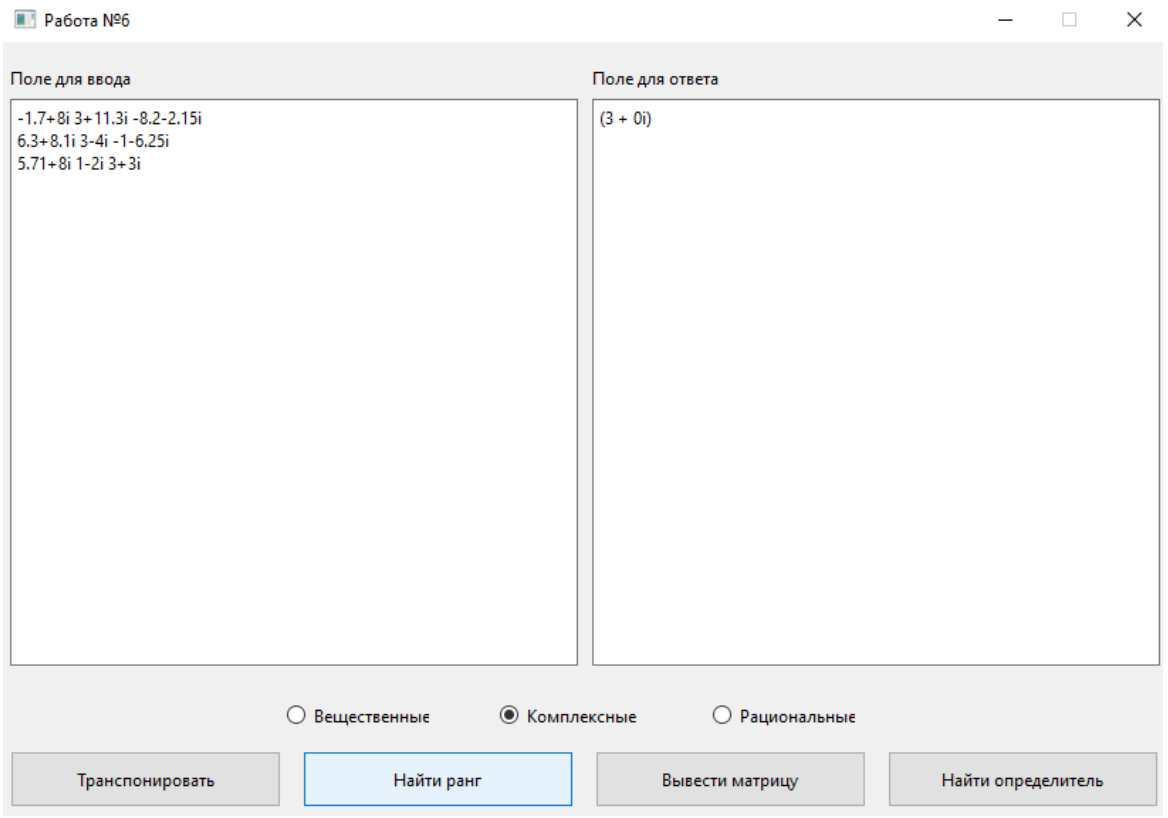

Рис. 8 - Тест 1.

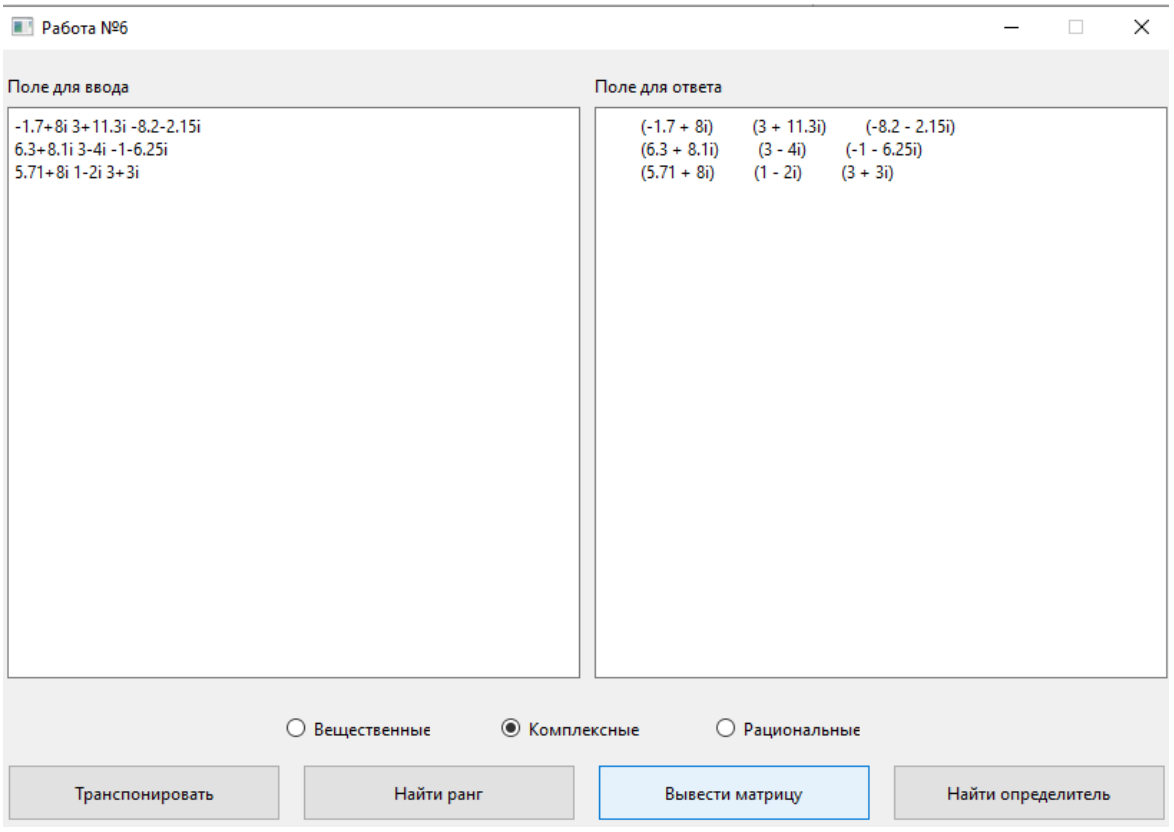

Рис. 9 - Тест 1.

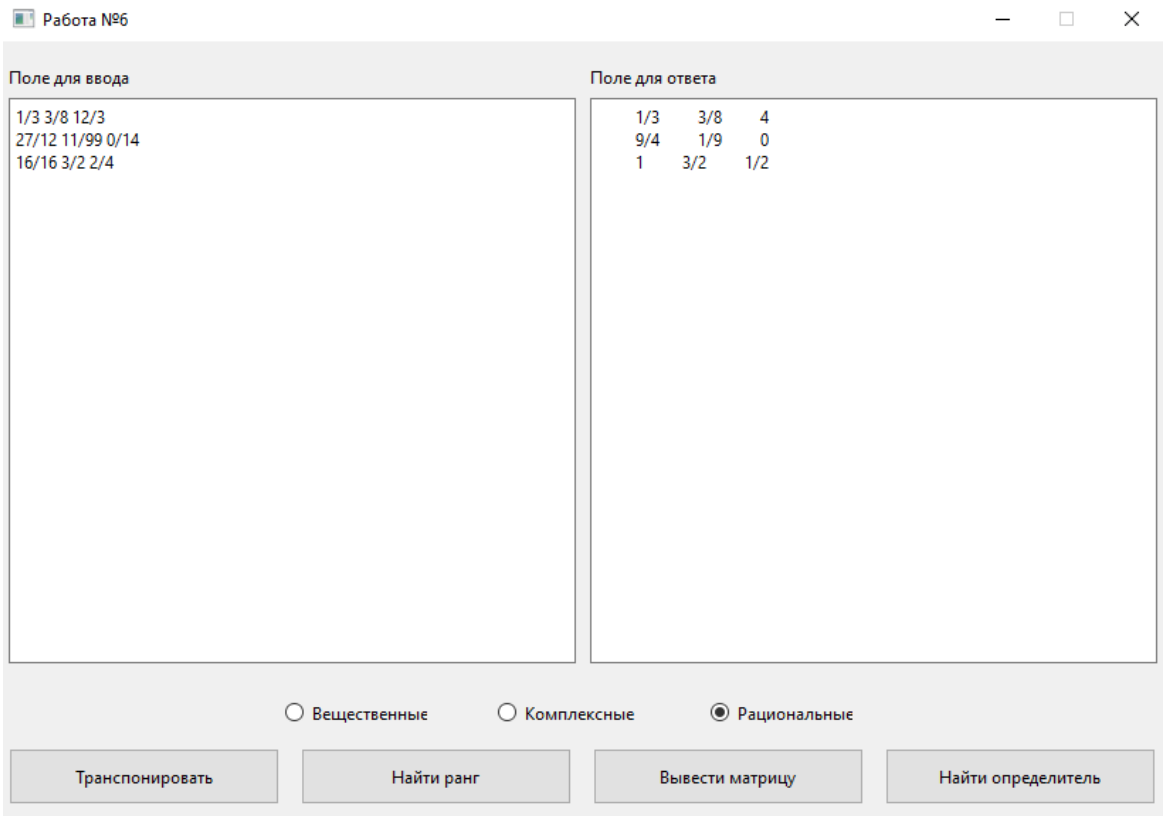

Рис. 10 - Тест 2.

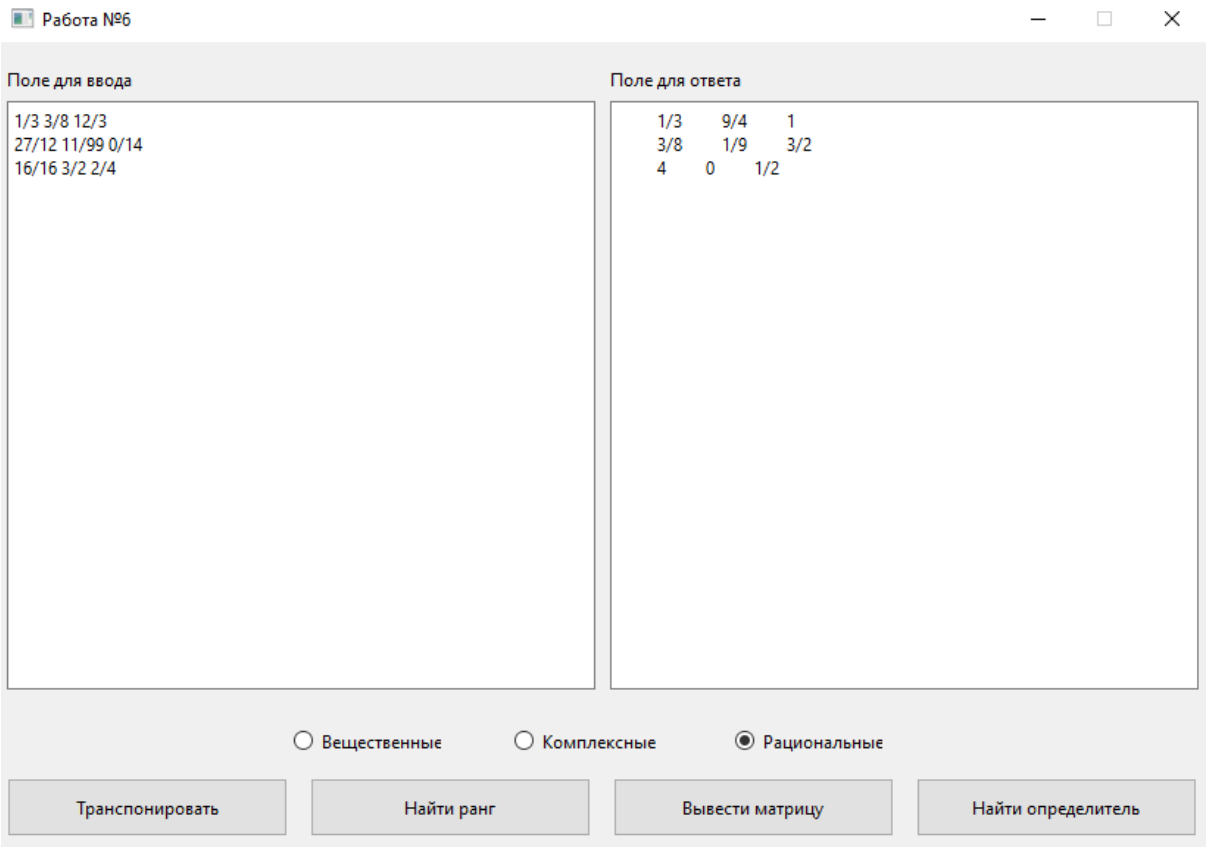

Рис. 11 - Тест 2.

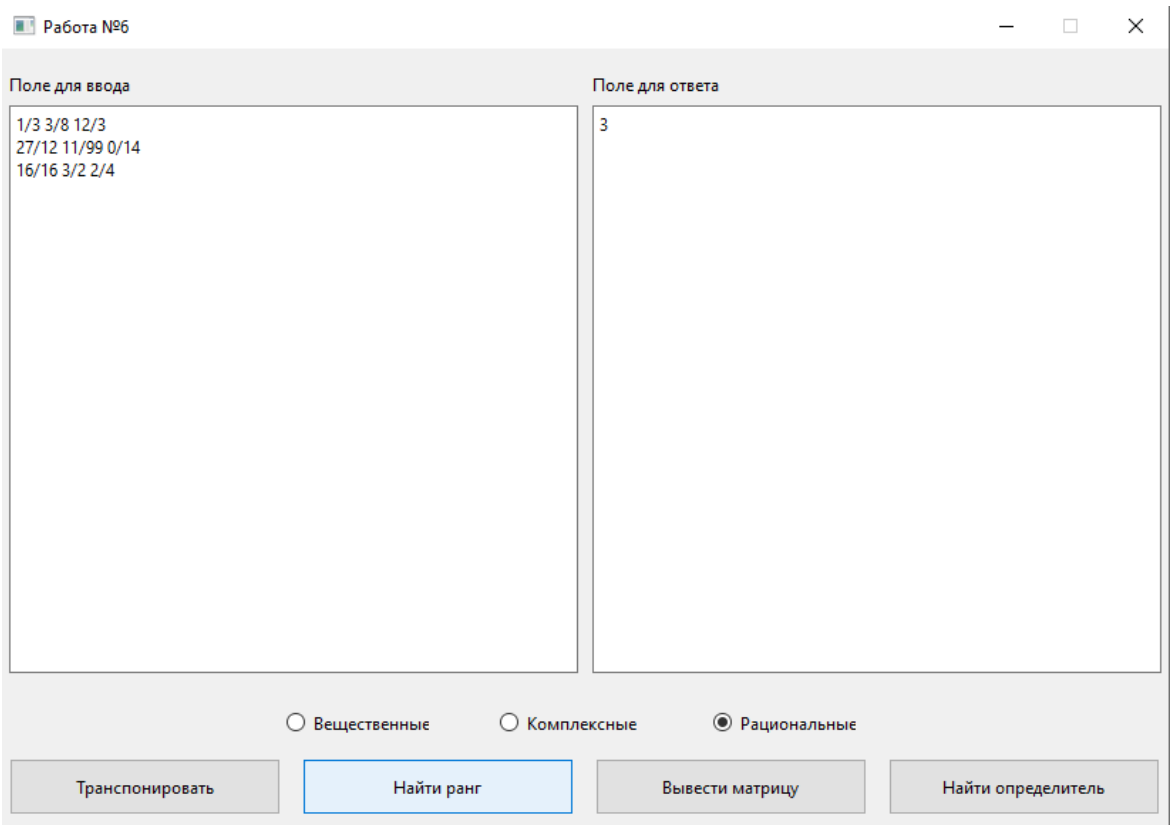

Рис. 12 - Тест 2.

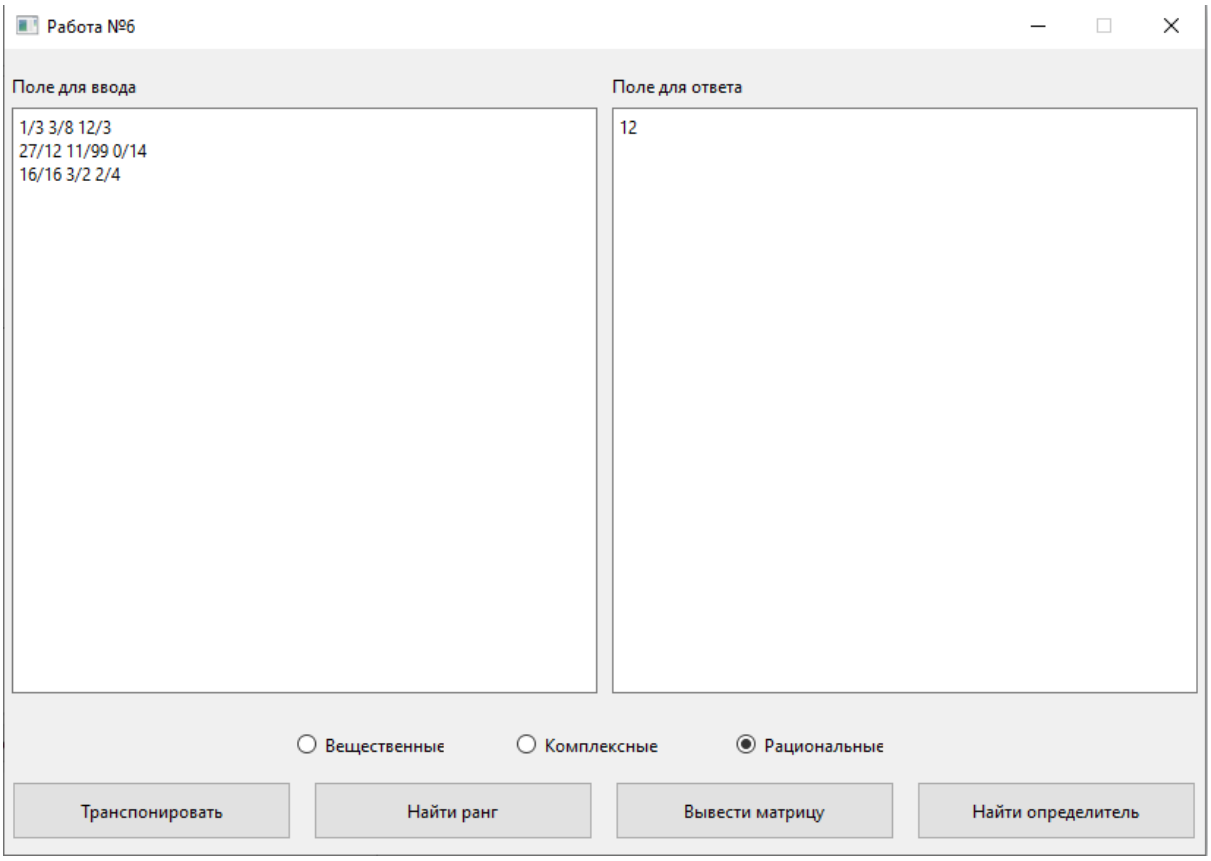

Рис. 13 - Тест 2.

6. Вывод.

В ходе работы мы создали распределенное приложение, включающее клиентскую и серверную части, взаимодействующие посредством сетевого обмена сообщениями через класс TCommunicator**.**

Был использованы классы TRationalNumber и TComplex. TRationalNumber имеет набор основных операторов для работы с рациональными числами, а TComplex производит работу с комплексными числами. Был использован создан класс TMatrix. TMatrix имеет функционал для работы с матрицами. А также создан класс TSquareMatrix, который наследовался от TMatrix, но были переопределены некоторые методы и созданы дополнительные (метод для нахождения определителя и ранга). На основе этого была рассмотрена работа с наследованием.

Был реализован графический пользовательский интерфейс с использованием классов QWidget, QPushButton, QTextEdit и QLabel. А также рассмотрена работа со слотами и сигналами.