МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Информационных систем

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Объектно-ориентированное программирование»

Студенты гр. 9374 _____ Богданова Е.М.
Калиненкова М.В.

Преподаватель _____ Егоров С.С.

Санкт-Петербург
2022

ЗАДАНИЕ НА ПРАКТИЧЕСКУЮ РАБОТУ

Создать консольное приложение, реализующее функции перечисленные в описании работы №1 (вычисление корней, вычисление значения, представление полинома в классической и канонических формах) на множестве рациональных чисел.

Рациональное число - это несократимая дробь a/b, где a и b целые, причем b > 0.

Приложение должно включать основной модуль, модуль «application», модуль «polinom» и модуль «rational».

Для этого в проект лабораторной работы №1 следует добавить модуль с описанием и реализацией класса рациональных чисел TRational. Класс TRational должен быть встроен в проект согласно диаграмме классов на рис.2. При этом основной модуль, модуль «application» и модуль «polinom» не должны изменяться. Изменения вносятся лишь в заголовочный файл number.h, где

```
typedef int number;
```

следует заменить на

```
#include «rational.h»
typedef TRational number;
```

В классе TRational следует определить только те члены класса и спецификации, которые необходимы для совместимости модулей проекта и реализации отношений, приведенных в ДК объектной модели.

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

СПЕЦИФИКАЦИИ КЛАССОВ

1. ***ConsoleApplication*** - класс для связи пользователя и других классов между собой.

private Scanner scanner	Объект класса Scanner , используется для ввода из консоли.
private CustomType polinomType	Тип полинома(пока только целочисленный).
private boolean exit	Переменная, используемая для проверки, нажата ли кнопка выход.
public ConsoleApplication()	Конструктор класса, в нем инициализируется объект типа Scanner.
public void exec()	Метод для запуска приложения.
private void chooseType()	Метод для выбора типа полинома.
private List<Integer> getCoefficient()	Метод для ввода коэффициентов полинома и проверки их на корректность.
private int getNumber()	Метод для проверки ввода целого числа(используется при выборе пункта меню).
private void menu()	Главный метод класса, который отвечает за обработку команд ввода.
private void createPolinom()	Метод для ввода значений коэффициентов и создания полинома.
private void printOptionsMenu()	Метод печатает к консоли возможные операции.

2. ***Polinom*** – класс для представления в программе полинома второй степени.

private T a	Переменная, хранящая значение

	коэффициента при x^2.
private T b	Переменная, хранящая значение коэффициента при x.
private T c	Переменная, хранящая значение свободного члена.
private T x1	Переменная, хранящая значение первого корня.
private T x2	Переменная, хранящая значение второго корня.
public Polinom(T a, T b, T c)	Конструктор класса, принимает коэффициенты.
public Roots<T> getRoots()	Метод, вычисляющий корни полинома. Если корни вычислить удалось, возвращает true, если нет, возвращает false и сообщает пользователю, что корней нет.
public void printEquation()	Метод выводит в консоль полином в виде a*x^2+b*x+c.
public void printCanonical()	Метод выводит в консоль полином в виде a(x-x1)(x-x2).
public T solveWithX(CustomNumber x)	Метод вычисляет значение полинома в заданной точке x.

3. ***Main*** – главный класс, содержит метод main, который является точкой входа.

public static void main(String[] args)	Точка входа в программу, где создается объект класса ConsoleApplication и вызывается его функция exec().

4. ***CustomNumber*** – абстрактный класс, представляющий число.

public abstract CustomNumber plus(CustomNumber a)	Абстрактный метод для сложения.
public abstract CustomNumber minus(CustomNumber a)	Абстрактный метод для вычитания.
public abstract CustomNumber mult(CustomNumber a)	Абстрактный метод для умножения.
public abstract CustomNumber mult(int a)	Абстрактный метод для умножения на целое число.
public abstract CustomNumber div(CustomNumber a)	Абстрактный метод для деления.
public abstract boolean biggerZero()	Абстрактный метод для проверки больше ли нуля число.
public abstract boolean isZero()	Абстрактный метод для проверки является ли число нулем.
public abstract CustomNumber sqrt()	Абстрактный метод для вычисления квадратного корня числа.
public abstract Number getValue()	Абстрактный метод для получения значения наследников CustomNumber
public abstract boolean isTheSame(double exactX)	Абстрактный метод для проверки корней.

5. ***CustomInteger*** – класс для представления целых чисел, наследуется от CustomNumber.

private int value	Переменная, хранящая значение числа.
public CustomInteger(int value)	Конструктор класса, принимает значение.
private CustomInteger convertToCustomInteger(CustomNumber a)	Метод кастует CustomNumber к CustomInteger.
@Override public CustomInteger plus(CustomNumber a)	Метод для сложения CustomInteger.

@Override public CustomInteger minus(CustomNumber a)	Метод для вычитания CustomInteger.
@Override public CustomInteger mult(CustomNumber a)	Метод для умножения CustomInteger.
@Override public CustomInteger mult(int a)	Метод для умножения CustomInteger на целое число.
@Override public CustomInteger div(CustomNumber a)	Метод для деления CustomInteger.
@Override public boolean biggerZero()	Метод для проверки больше ли CustomInteger нуля.
@Override public boolean isZero()	Метод для проверки, равен ли CustomInteger нулю.
@Override public CustomInteger sqrt()	Метод для вычисления квадратного корня CustomInteger.
@Override public Double getValue()	Метод для получения значения числа.
@Override public boolean isTheSame(double exactX)	Метод для сравнения чисел.
@Override public String toString()	Метод для описания вида вывода в консоль CustomInteger.

6. ***CustomRational*** – класс для представления рациональных дробей, наследуется от CustomNumber.

private int numerator	Переменная, хранящая значение числителя.
private int denominator	Переменная, хранящая значение знаменателя.
public CustomRational(int numerator, int denominator)	Конструктор класса, принимает числитель и знаменатель.

private CustomRational convertToCustomRational(CustomNumber a)	Метод кастует CustomNumber к CustomRational.
@Override public CustomRational plus(CustomNumber a)	Метод для сложения CustomRational.
private CustomRational createAndReturnCustomRational(int num, int denom)	Метод для создания CustomRational
@Override public CustomRational minus(CustomNumber a)	Метод для вычитания CustomRational.
@Override public CustomRational mult(CustomNumber a)	Метод для умножения CustomRational.
@Override public CustomRational mult(int a)	Метод для умножения CustomRational на целое число.
@Override public CustomRational div(CustomNumber a)	Метод для деления CustomRational.
@Override public boolean biggerZero()	Метод для проверки больше ли CustomRational нуля.
@Override public boolean isZero()	Метод для проверки, равен ли CustomRational нулю.
@Override public CustomRational sqrt()	Метод для вычисления квадратного корня CustomRational.
@Override public Double getValue()	Метод для получения значения числа.
@Override public boolean isTheSame(double exactX)	Метод для сравнения чисел.

private List<Integer> reduceFraction(int num, int denom)	Метод для сокращения дроби.
private int getGCF(int a, int b)	Метод для получения НОДа двух чисел.
@Override public String toString()	Метод для описания вида вывода в консоль CustomRational.

7. ***CustomType*** – перечисление, содержит все возможные типы полинома.

CUSTOM_INT	Константа, соответствующая целочисленному типу.
CUSTOM_RATIONAL	Константа, соответствующая дроби.

8. ***NoRootsException*** – исключение, которое может возникнуть во время работы программы.

@Override public String getMessage()	Метод, в котором определяется сообщение выводимое в консоль.

Для выполнения данной работы был выбран высокоуровневый язык программирования Java.

Основным классом данной работы является класс ConsoleApplication. Он отвечает за:

1. Создание объекта класса Scanner, который, в свою очередь, отвечает за весь ввод пользователем данных.
2. Получение значения введённого полинома второй степени и создания на его основе экземпляра класса Polinom.
3. Запуск методов для нахождения корней и значения функции у созданного объекта класса Polinom.

4. Получение результатов методов, описанных в пункте выше и выводе их на экран консоли.

При запуске приложения пользователю предлагается выбрать тип полинома: целочисленный или рациональный. Далее создается полином по умолчанию с коэффициентами a=1, b=0, c=0 и выводится меню с возможными действиями.

Вторым созданным классом является класс Polinom. Данный класс является представлением полинома второй степени в коде со всеми необходимыми методами, которые необходимы по заданию.
Данный класс имеет публичный конструктор с тремя обязательными параметрами типа T, который должен быть наследником класса CustomNumber.

Для того, чтобы получить корни уравнения, используется публичный метод getRoots(), который не принимает аргументов, т.к. корни ищутся при уравнении, равном 0.

В данном методе вычисляется дискриминант, а также выполняется проверка: если дискриминант меньше нуля, то действительных корней не существует, что, в итоге, и будет выведено на экран. В противном случае корни будут вычислены.

Дополнительно реализовано вычисление более точных корней, для последующей их сверки с получившимися при вычислениях в целых числах или в рациональных дробях. В консоли также выводятся оба этих результата и информация о том, было ли произведено округление.

Есть два варианта вывода самого полинома: в форме a*x^2+b*x+c и в канонической форме a(x-x1)(x-x2), за это отвечают методы printEquation() и printCanonical() соответственно. В первом просто записываем все коэффициенты, иксы и знаки в одну строку, которую после выводим на экран. Во втором – проверяем, вычислены ли корни, если они не вычислены, сообщаем об этом пользователю, если вычислены, соединяем все в одну строку и выводим ее в консоль.

Для поиска значения полинома в конкретной точке, используется метод solveWithX(), который принимает x.

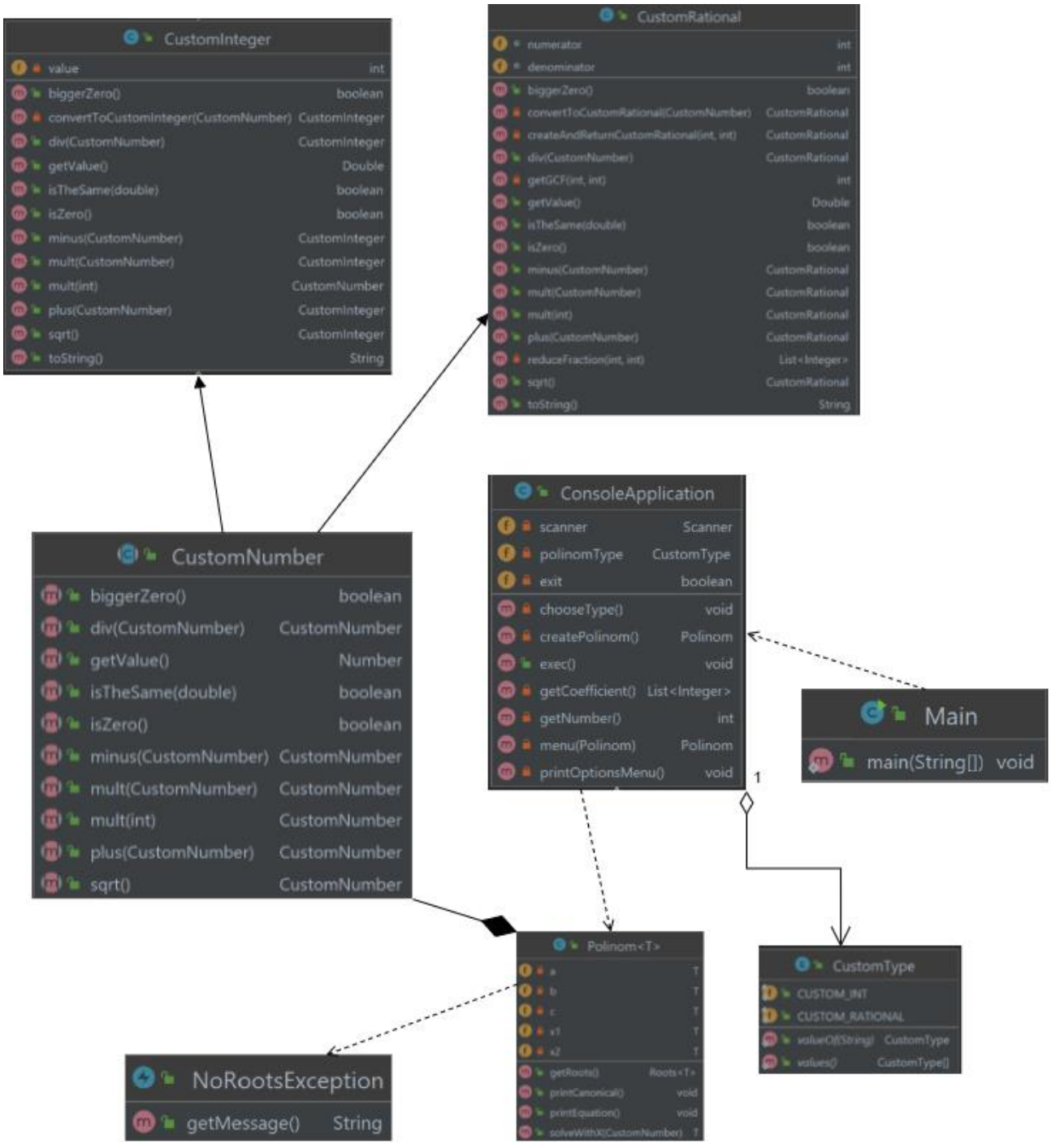
Для возможности создания полиномов разных типов и переопределения арифметических операций используется абстрактный класс CustomNumber. В первой работе требовалось реализовать работу с целочисленным типом, поэтому был создан класс CustomInteger, он унаследован от CustomNumber. Этот класс является оберткой над обычным интом и нужен только, чтобы переопределить арифметические операции и вывод в консоль(метод toString()). В этой работе для работы с дробными коэффициентами был создан класс CustomRational для тех же целей, что и предыдущий.

Перечисление CustomType пока содержит два значения CUSTOM_INT и CUSTOM_RATIONAL обозначающие целочисленный и рациональный тип соответственно. Эта конструкция используется для проверки типа введенных коэффициентов.

Класс Main содержит статический метод main(), который является точкой входа в программу. В нем мы создаем объект типа ConsoleApplication и вызываем его метод exec().

Язык Java был выбран потому, что у него очень удобный функционал работы с классами (в отличие от C++), при максимальной приближённости синтаксиса к C++ по сравнению с аналогами (Python, например).

ДИАГРАММА КЛАССОВ



По заданию требовалось отношение композиции между полиномом и рациональным числом. Это сделать легко, можно просто назначить переменным такой тип. Но в таком случае нельзя будет передать целое число, не трогая код. Отношение композиции построено между классом полинома и родителем CustomRational и CustomInteger.

ПРИМЕРЫ РАБОТЫ ПРИЛОЖЕНИЯ

Пример 1.
Решим уравнение 6/27*x^2+7/9*x+1/3=0
И вычислим значение при x=9/2

```
a = 6/27
b = 7/9
c = 1/3
1.Ввести коэффициенты a, b, c
2.Рассчитать корни и вывести в консоль
3.Ввести x и рассчитать значения полинома
4.Текстовое представление полинома в форме a*x^2 + b*x + c
5.Вывод тестового представления полинома в канонической форме a(x - x1)(x - x2)
6.Выход
2
(exact x1 = -0,50000)
(exact x2 = -3,0000)
x1 принадлежит заданному множеству
x2 принадлежит заданному множеству
x1 = -1/2
x2 = -3
```

```
1.Ввести коэффициенты a, b, c
2.Рассчитать корни и вывести в консоль
3.Ввести x и рассчитать значения полинома
4.Текстовое представление полинома в форме a*x^2 + b*x + c
5.Вывод тестового представления полинома в канонической форме a(x - x1)(x - x2)
6.Выход
3
При x = 0: 1/3
x = 9/2
При x = 9/2: 25/3
```

Результат работы программы совпадает с ожидаемым.

Пример 2.

Решим уравнение 3/20*x^2+5/2*x-42/5=0
И вычислим значение при x = 7/3

```
a = 3/20
b = 5/2
c = 42/5
1.Ввести коэффициенты a, b, c
2.Рассчитать корни и вывести в консоль
3.Ввести x и рассчитать значения полинома
4.Текстовое представление полинома в форме a*x^2 + b*x + c
5.Вывод тестового представления полинома в канонической форме a(x - x1)(x - x2)
6.Выход
2
(exact x1 = -4,6667)
(exact x2 = -12,000)
x1 принадлежит заданному множеству
x2 принадлежит заданному множеству
x1 = -14/3
x2 = -12
```

```
1.Ввести коэффициенты a, b, c
2.Рассчитать корни и вывести в консоль
3.Ввести x и рассчитать значения полинома
4.Текстовое представление полинома в форме a*x^2 + b*x + c
5.Вывод тестового представления полинома в канонической форме a(x - x1)(x - x2)
6.Выход
3
При x = 0: 42/5
x = 7/3
При x = 7/3: 301/20
```

Результат работы совпадает с ожидаемым.

ВЫВОД

В данной практической работе были разобраны основы объектно-ориентированного программирования, инициализация, вызов методов, присвоение переменных, разграничение доступа к полям и методам, скоупы классов, создание констант, внутренних классов, также разобрано наследование классов, композиция.

Основной частью данной программы являются методы взаимодействия экземпляров класса CustomRational. Так как в данной практической работе не используются встроенные числовые переменные языка, необходимо было переопределить данные операторы для нашего кастомного класса. Ввиду отсутствия в языке Java возможности переопределять операторы для работы с классами, единственным выходом стало создавать специальные методы для совершения над ними операций. Данная альтернатива является достаточно понятным вариантом и отлично вписываются в парадигмы ООП. Были получены навыки реализации данных методов в коде, а также навыки сокращения рациональных чисел.

Также были рассмотрены переопределение методов родительских классов (toString()) в языке Java и основные навыки парсинга строк для получения корректного ввода пользователя.