ГВУЗ "Донецкий национальный технический университет"

кафедра Прикладной математики и информатики

Лабораторная работа №2

по курсу "Тестирование и качество ПО"

по теме "Качественное проектирование и кодирование программного обеспечения"

Выполнил студент гр. ІПЗ-12а Егоров А. А.

Проверил доц. каф. ПМИ Федяев О. И.

Донецк – 2015

1 ПОСТАНОВКА ЗАДАЧИ

1. Выбрать подход к разработке программного продукта (структурный или объектно-ориентированный). Вспомнить принципы программной реализации проекта на основе этого подхода. Выбрать технологию и нотацию для разработки и описания формальных моделей проектирования.

2. На основе анализа спецификаций требований определить архитектуру библиотеки, как программной системы.

3. Определить перечень формальных моделей, которые планируется построить на этапе проектирования. Специфицировать их в соответствии с синтаксисом и семантикой выбранной нотации.

2 СПЕЦИФИКАЦИЯ ФОРМАЛЬНОЙ МОДЕЛИ

Поток событий:

1. Пользователь библиотеки создает комплексное число с помощью одной из форм:

1.1 С помощью алгебраической формы

1.2 С помощью экспоненциальной формы

1.3 С помощью тригонометрической формы

2. Пользователь производит базовые операции на комплексными числами:

2.1 Выполняет сложение двух чисел

2.2 Выполняет вычитание одного числа из другого

2.3 Выполняет умножение двух чисел

2.4 Выполняет деление одного числа на другое.

3. Пользователь получает значение комплексного числа в виде строки:

3.1 Получает строку в каноническом виде

3.2 Получает строку в экспоненциальном виде

3.3 Получает строку в тригонометрическом виде

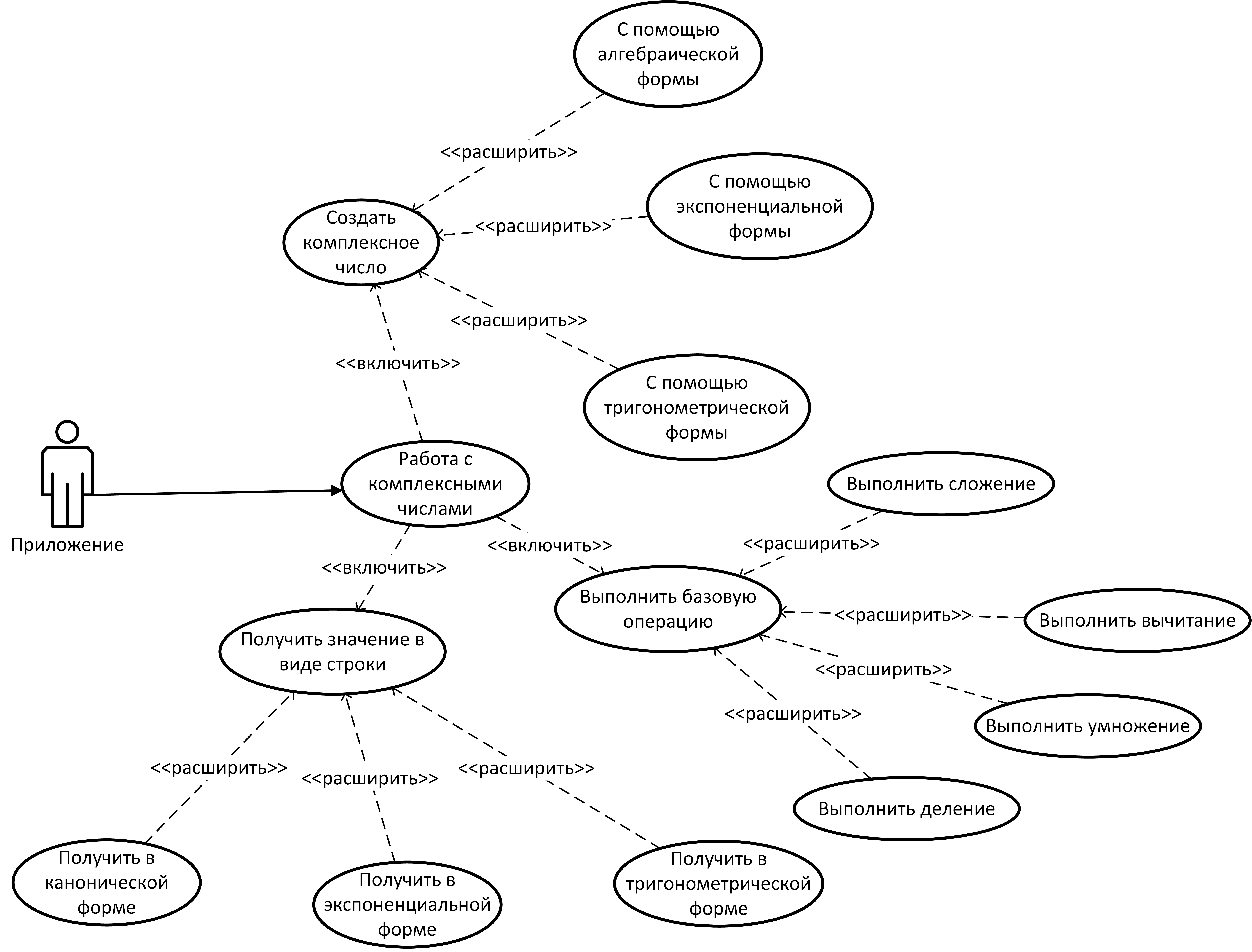


Рисунок 2.1 – Диаграмма прецедентов библиотеки комплексного числа

|  |  |
| --- | --- |
| **Имя** | Complex |
| **Документация** | Класс реализующий представление комплексного числа. |
| **Множественность** | N |
| **Иерархия** | java.lang.Object  java.lang.Number |
| **Суперкласс** | Number |
| **Интерфейса/ Реализация** | Serializable |
| **Использование** | - |
| **Поля** | Действительная часть (double),  Мнимая часть (double) |
| **Операции**  **Свойства** | Сложить (add),  Вычесть (sub),  Умножить (mul),  Разделить (div),  Создать с помощью алгебраической формы (fromCanonical),  Создать с помощью тригонометрической формы (fromPolar),  Создать с помощью экспоненциальной формы (fromExponent),  Получить строку в алгебраической форме (toCanonicalForm),  Получить строку в тригонометрической форме (toPolarForm),  Получить строку в экспоненциальной форме (toExponentForm)  … |
| Получить Действительную часть (getReal),  Получить Мнимую часть (getImag),  Получить Модуль (getModule),  Получить Аргумент (getArg) |
| **Устойчивость** | Статическая |
| **Объем памяти** | Минимум 16 байт |

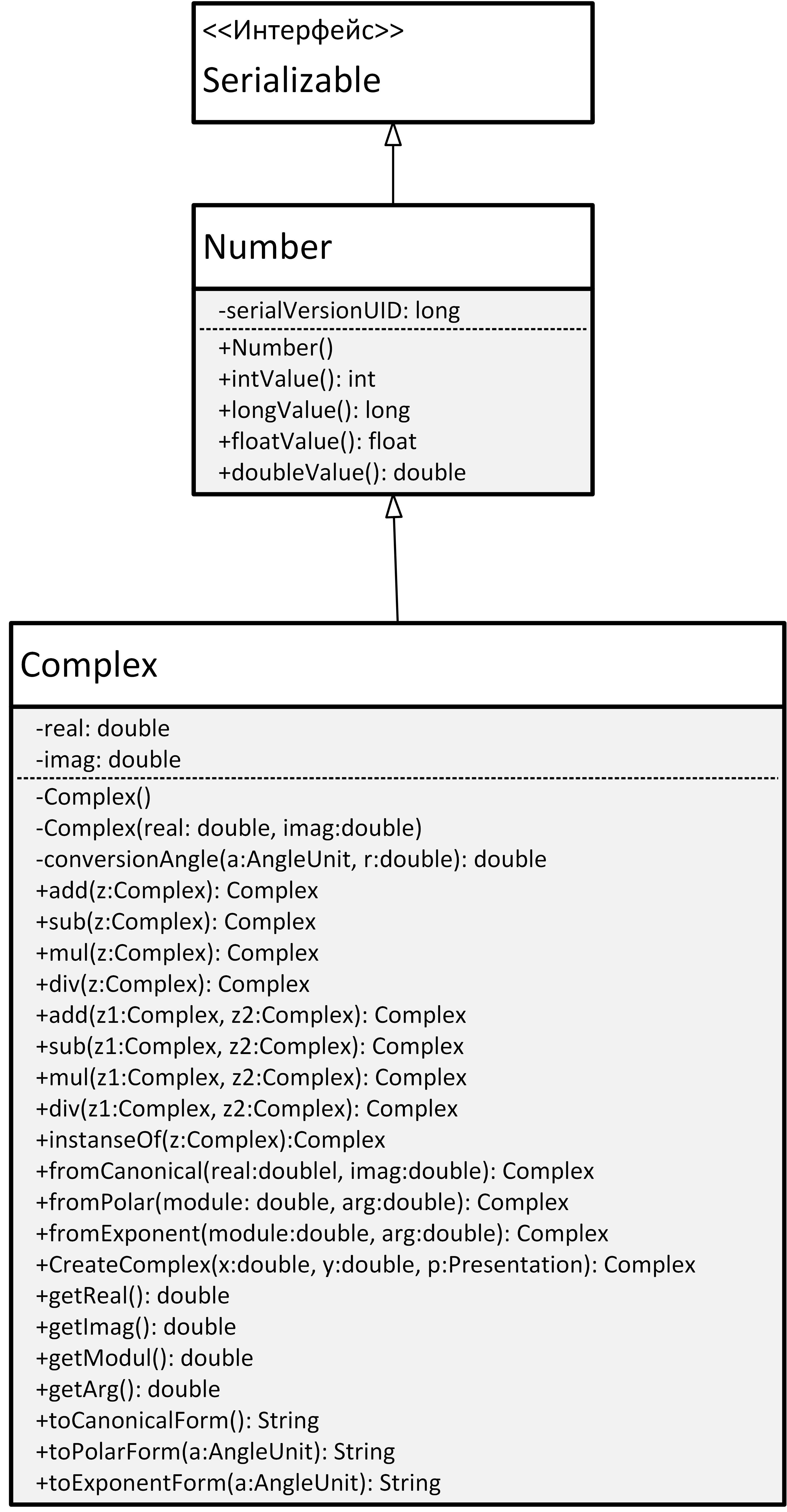


Рисунок 2.2 – Диаграмма классов

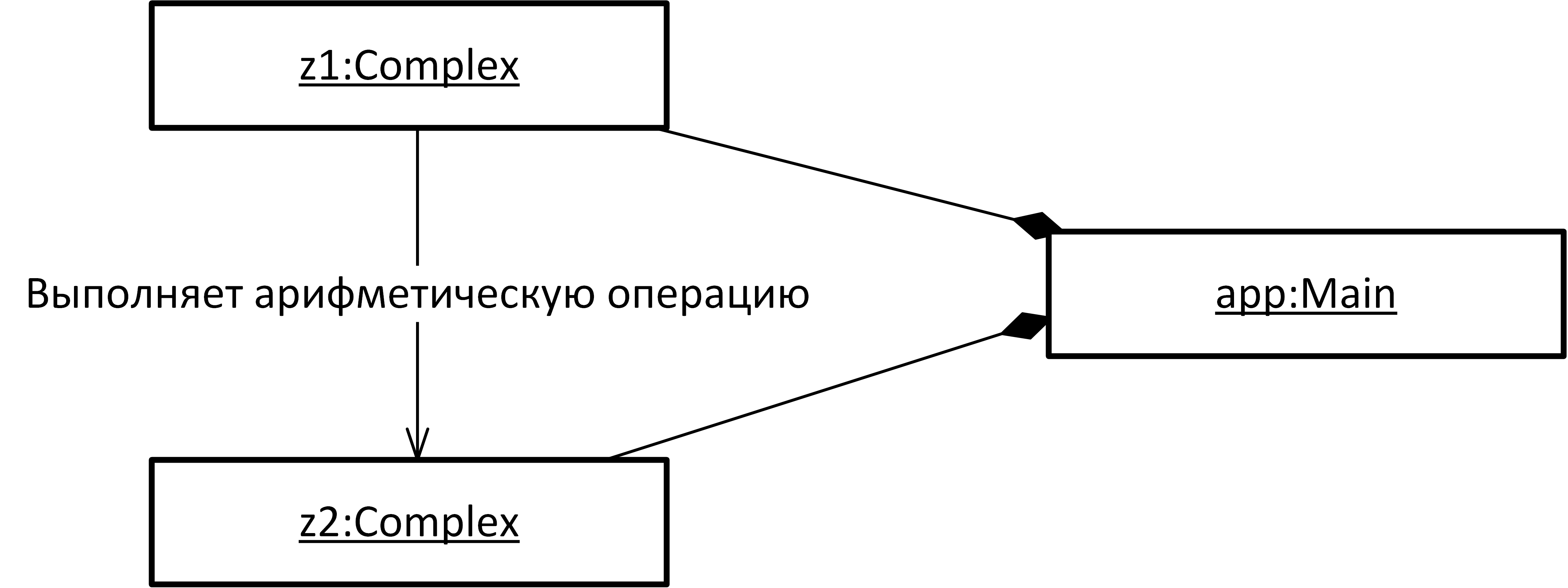


Рисунок 2.3 – Диаграмма объектов

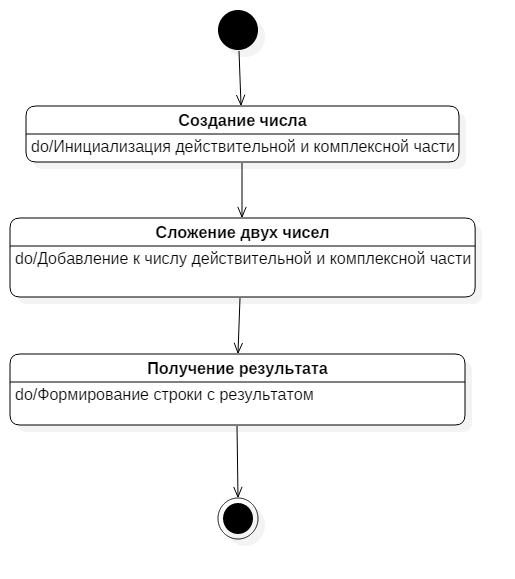


Рисунок 2.4 – Диаграмма состояния объекта класса Complex

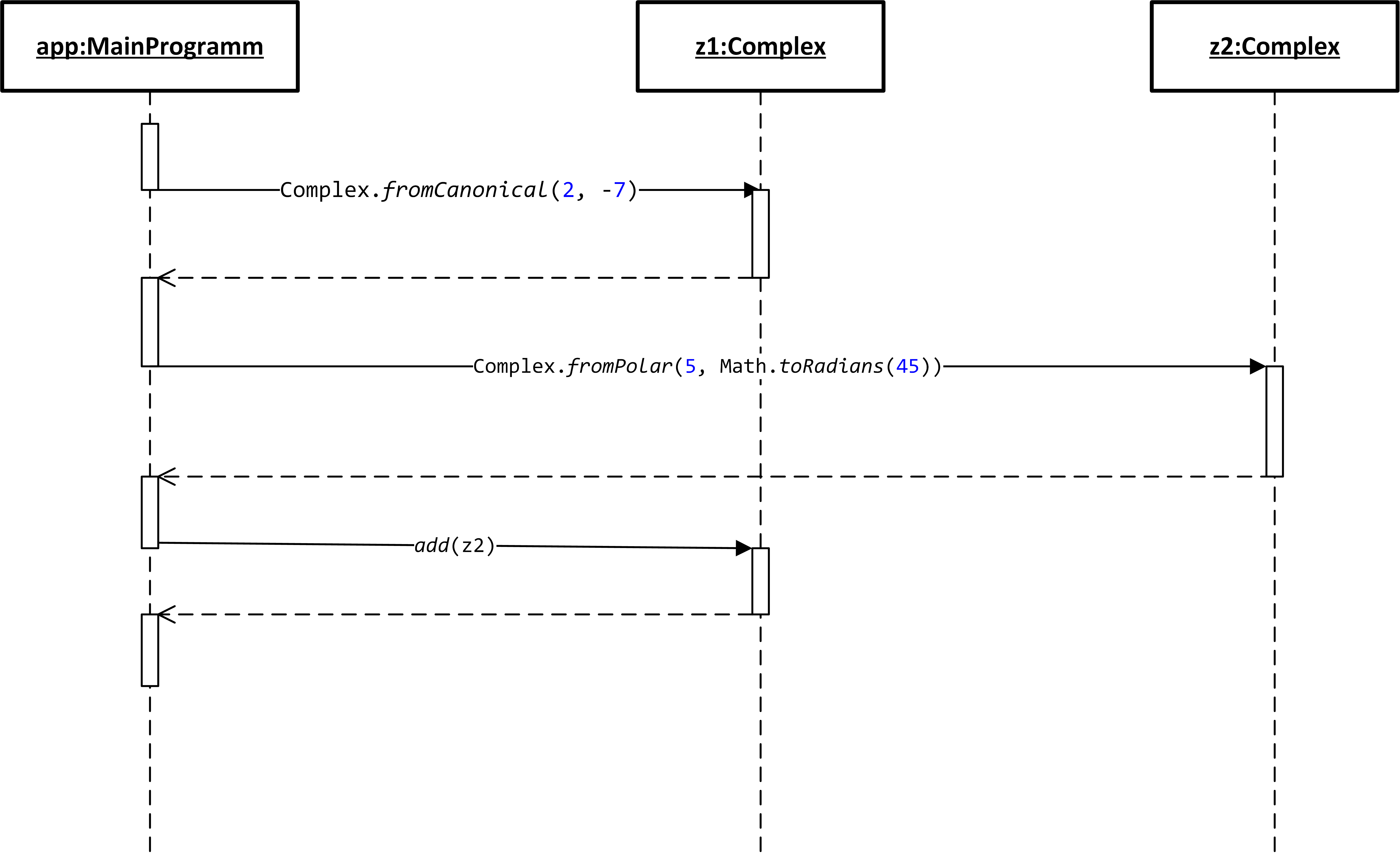


Рисунок 2.4 – Временная диаграмма взаимодействия приложения с библиотекой

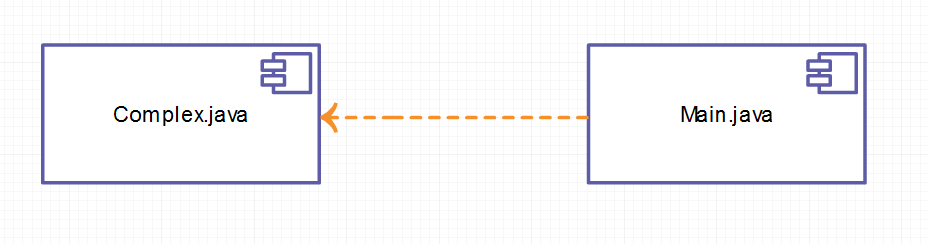


Рисунок 2.5 – Диаграмма компонентов

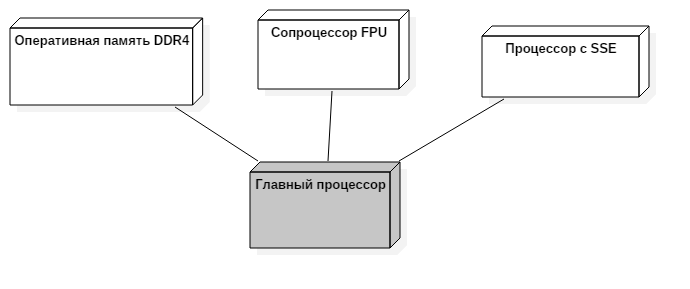


Рисунок 2.6 – Диаграмма развертывания

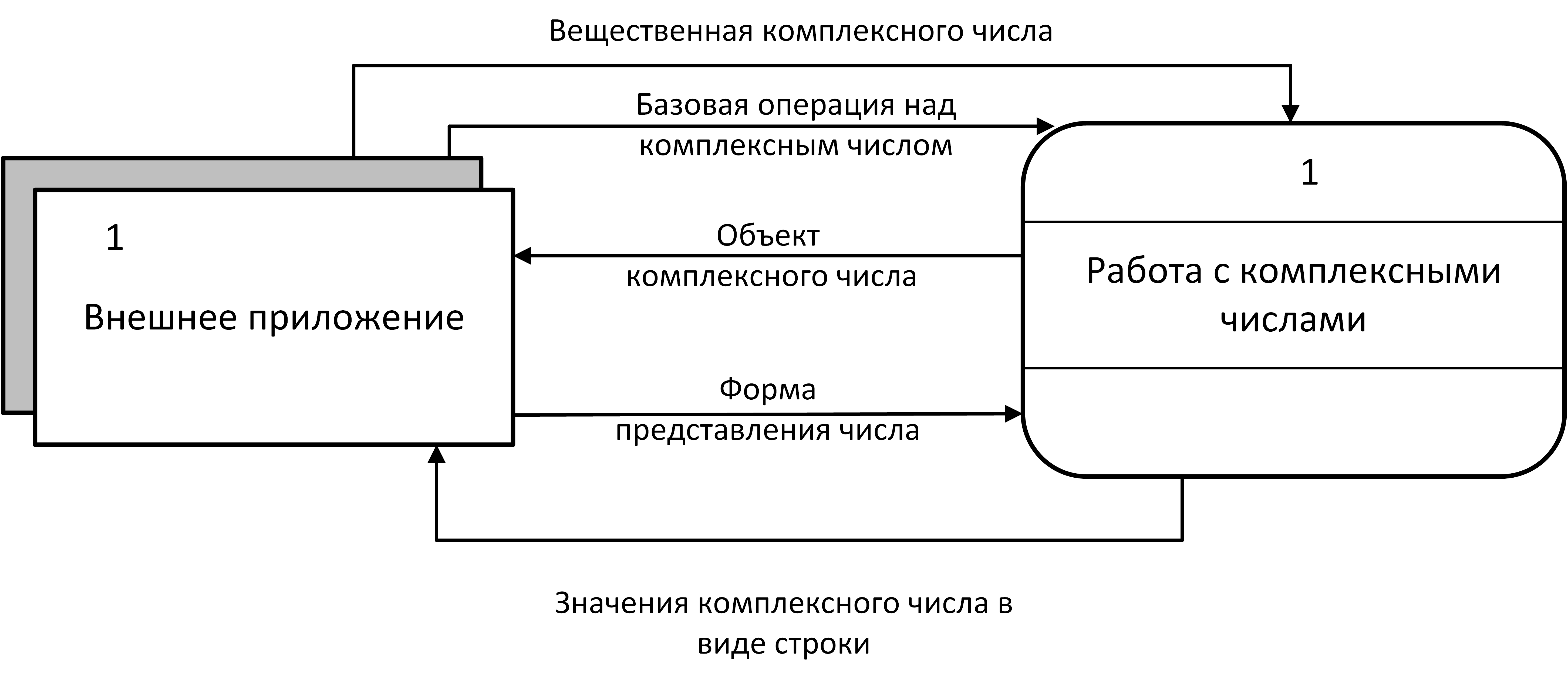


Рисунок 2.7 – Диаграмма потоков данных верхнего уровня

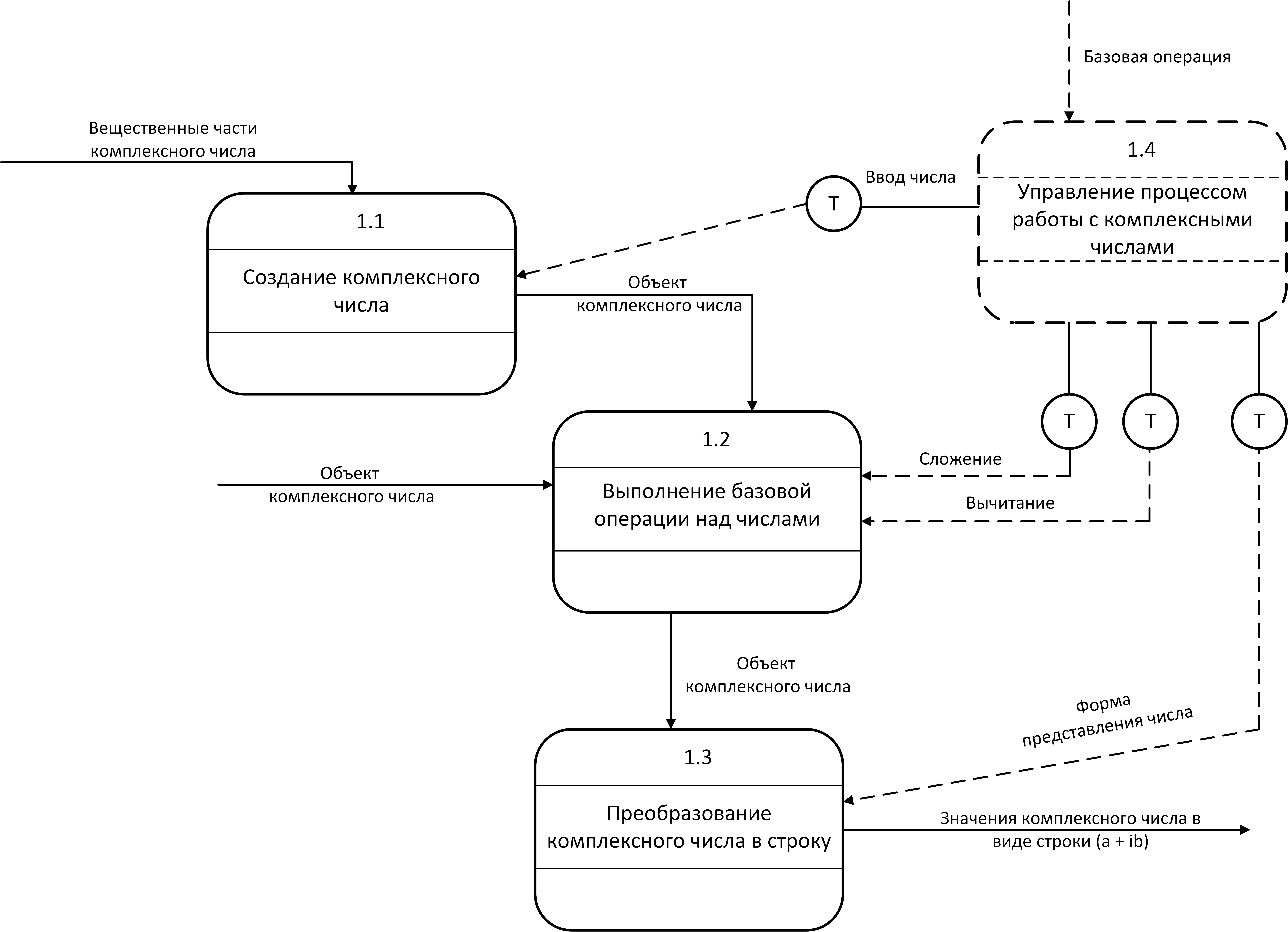


Рисунок 2.8 – Детализирующая диаграмма потоков данных библиотеки комплексных чисел

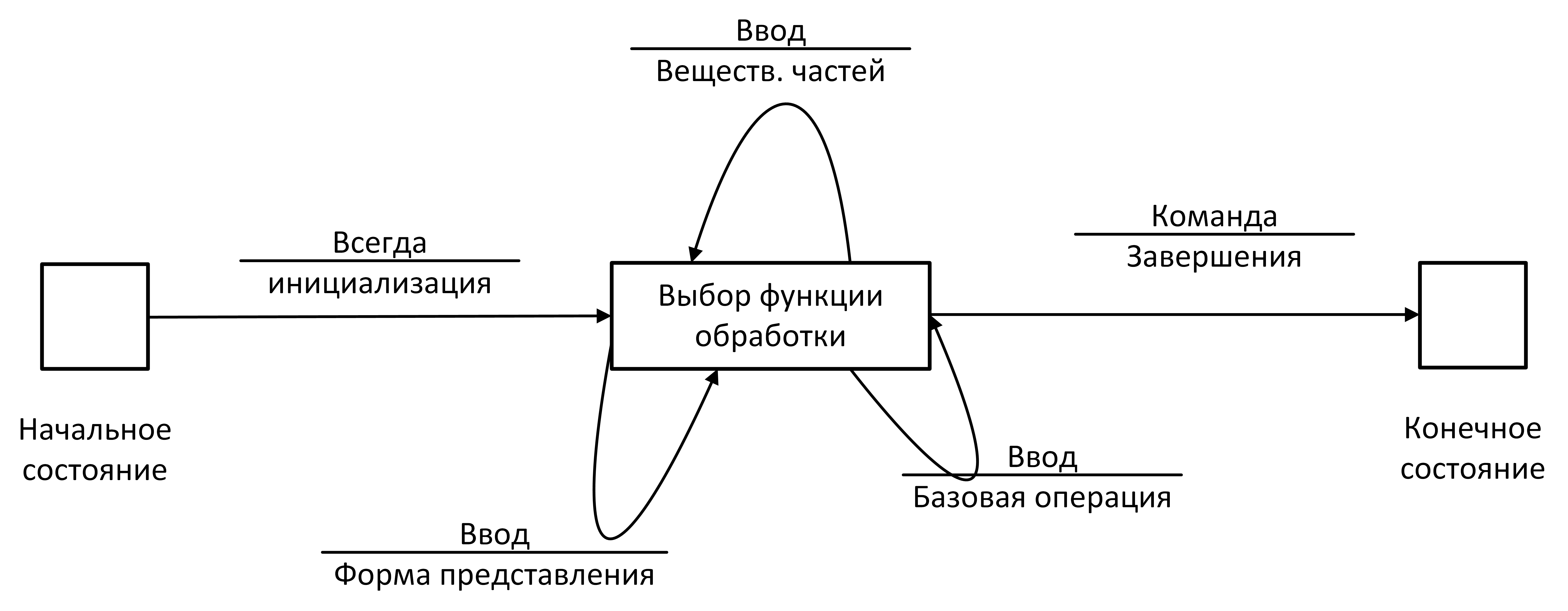


Рисунок 2.9 – Диаграмма переходов состояний библиотеки комплексных чисел

3 ФИЗИЧЕСКАЯ МОДЕЛЬ

3.1 Исходный код библиотеки

**package** yegorov.math;  
  
*/\*\*  
 \* Created by Yegorov Artem <yegorov0725@yandex.ru>  
 \*/***public class** Complex **extends** Number {  
 **public enum** Presentation {  
 ***CANONICAL***,  
 ***POLAR***,  
 ***EXPONENT*** }  
  
 **public enum** AngleUnit {  
 ***DEGREE***,  
 ***RADIAN***,  
 ***GRAD*** }

**private double real**;  
 **private double imag**;  
  
  
 **private** Complex() {  
 **this**.**real** = 0d;  
 **this**.**imag** = 0d;  
 }  
  
 **private** Complex(**double** real, **double** imag) {  
 **this**.**real** = real;  
 **this**.**imag** = imag;  
 }  
  
 */\*\*  
 \* Addition operations  
 \** ***@param z*** *other complex number  
 \** ***@return*** *Complex object  
 \*/* **public** Complex add(Complex z) {  
  
 **this**.**real** += z.getReal();  
 **this**.**imag** += z.getImag();  
  
 **return this**;  
 }  
  
 */\*\*  
 \* Subtraction operations  
 \** ***@param z*** *other complex number  
 \** ***@return*** *Complex object  
 \*/* **public** Complex sub(Complex z) {  
  
 **this**.**real** -= z.getReal();  
 **this**.**imag** -= z.getImag();  
 **return this**;  
 }  
 */\*\*  
 \* Multiplication operations  
 \** ***@param z*** *other complex number  
 \** ***@return*** *Complex object  
 \*/* **public** Complex mul(Complex z) {  
  
 **double** real = **this**.**real** \* z.getReal() - **this**.**imag** \* z.getImag();  
 **double** imag = **this**.**imag** \* z.getReal() + **this**.**real** \* z.getImag();  
  
 **this**.**real** = real;  
 **this**.**imag** = imag;  
  
 **return this**;  
 }  
  
 */\*\*  
 \* Division operations  
 \** ***@param z*** *other complex number  
 \** ***@return*** *Complex object  
 \*/* **public** Complex div(Complex z) {  
  
 **double** d = **this**.**imag** \* **this**.**imag** + z.getImag() \* z.getImag();  
 **double** real = (**this**.**real** \* z.getReal() + **this**.**imag** \* z.getImag()) / d;  
 **double** imag = (**this**.**imag** \* z.getReal() - **this**.**real** \* z.getImag()) / d;  
  
 **this**.**real** = real;  
 **this**.**imag** = imag;  
  
 **return this**;  
 }

*/\*\*  
 \* Addition operations (not mutable input parameters)  
 \** ***@param z1*** *- First complex number  
 \** ***@param z2*** *- Second complex number  
 \** ***@return*** *New complex number with with the result of the operation  
 \*/* **public static** Complex add(Complex z1, Complex z2) {  
 **return** Complex.*instanseOf*(z1).add(z2);  
 }  
  
 */\*\*  
 \* Subtraction operations (not mutable input parameters)  
 \** ***@param z1*** *- First complex number  
 \** ***@param z2*** *- Second complex number  
 \** ***@return*** *New complex number with with the result of the operation  
 \*/* **public static** Complex sub(Complex z1, Complex z2) {  
 **return** Complex.*instanseOf*(z1).sub(z2);  
 }  
  
 */\*\*  
 \* Multiplication operations (not mutable input parameters)  
 \** ***@param z1*** *- First complex number  
 \** ***@param z2*** *- Second complex number  
 \** ***@return*** *New complex number with with the result of the operation  
 \*/* **public static** Complex mul(Complex z1, Complex z2) {  
 **return** Complex.*instanseOf*(z1).mul(z2);  
 }  
 */\*\*  
 \* Division operations (not mutable input parameters)  
 \** ***@param z1*** *- First complex number  
 \** ***@param z2*** *- Second complex number  
 \** ***@return*** *New complex number with with the result of the operation  
 \*/* **public static** Complex div(Complex z1, Complex z2) {  
 **return** Complex.*instanseOf*(z1).div(z2);  
 }  
  
  
 */\*\*  
 \* Copy the input object  
 \** ***@param z*** *- Complex number  
 \** ***@return*** *New complex number  
 \*/* **public static** Complex instanseOf(Complex z) {  
 **return new** Complex(z.getReal(), z.getImag());  
 }  
  
 */\*\*  
 \** ***@param real*** *- Real part of the complex number  
 \** ***@param imag*** *- Imaginary part of the complex number  
 \** ***@return*** *New complex number  
 \*/* **public static** Complex fromCanonical(**double** real, **double** imag) {  
 **return new** Complex(real, imag);  
 }  
  
 */\*\*  
 \** ***@param module*** *- Absolute value (or modulus or magnitude) of a complex number sqrt(real^2 + imag^2)  
 \** ***@param arg*** *- The argument of z (in many applications referred to as the "phase") is the angle of the radius OP  
 \* with the positive real axis,  
 \** ***@return*** *New complex number  
 \*/* **public static** Complex fromPolar(**double** module, **double** arg) {  
 *// http://h4e.ru/obshchie-svedeniya/145-primery-reshenij-kompleksnykh-chisel-kalkulyator*  **double** real = module \* Math.*cos*(arg);  
  **double** imag = module \* Math.*sin*(arg);  
  
  **return new** Complex(real, imag);  
}

*/\*\*  
 \** ***@param module*** *Absolute value (or modulus or magnitude) of a complex number sqrt(real^2 + imag^2)  
 \** ***@param arg*** *- The argument of z (in many applications referred to as the "phase") is the angle of the radius OP  
 \* with the positive real axis,  
 \** ***@return*** *New complex number  
 \*/* **public static** Complex fromExponent(**double** module, **double** arg) {  
  
 Complex c = Complex.*fromPolar*(module, arg);  
  
 **return** c;  
 }  
 */\*\*  
 \** ***@param x*** *- Real part if Presentation Canonical, else absolute value (or modulus or magnitude)  
 \** ***@param y*** *- Imaginary part if Presentation Canonical, else the argument of complex number  
 \** ***@param p*** *- Presentation of complex number (Canincal, Polar, Exponent)  
 \** ***@return*** *New complex number  
 \*/* **public static** Complex CreateComplex(**double** x, **double** y, Presentation p) {  
 **switch** (p) {  
 **case *CANONICAL***:  
 **return** *fromCanonical*(x, y);  
 **case *POLAR***:  
 **return** *fromPolar*(x, y);  
 **case *EXPONENT***:  
 **return** *fromExponent*(x, y);  
 **default**:  
 **return new** Complex();  
 }  
 }

*/\*\*  
 \* Real part of the complex number  
 \*/* **public double** getReal() {  
 **return real**;  
 }  
  
 */\*\*  
 \* Imaginary part of the complex number  
 \*/* **public double** getImag() {  
 **return imag**;  
 }  
  
 */\*\*  
 \* Absolute value (module) of the complex number (sqrt(real^2 + imag^2))  
 \*/* **public double** getModule() {  
 **return** Math.*sqrt*(**real** \* **real** + **imag** \* **imag**);  
 }  
  
 */\*\*  
 \* Argument is a function operating on complex numbers.  
 \* It gives the angle between the positive real axis to the line joining the point to the origin  
 \*/* **public double** getArg() {  
 *// https://en.wikipedia.org/wiki/Complex\_number#Polar\_form* **double** arcTan = Math.*atan*(**imag** / **real**); *// Math.atan2(imag, real);* **double** result;  
  
 **if**(**real** > 0) result = arcTan;  
 **else if**(**real** < 0 && **imag** >= 0) result = arcTan + Math.***PI***;  
 **else if**(**real** < 0 && **imag** < 0) result = arcTan - Math.***PI***;  
 **else if**(**real** == 0 && **imag** > 0) result = Math.***PI*** / 2.0;  
 **else if**(**real** == 0 && **imag** < 0) result = -Math.***PI*** / 2.0;  
 **else** result = Double.***NaN***;  
 **return** result;  
 }  
  
  
 */\*\*  
 \* Present a complex number in canonical form  
 \** ***@return*** *String canonical form, example: 6,00 + i\*3,00  
 \*/* **public** String toCanonicalForm() {  
 **return** String.*format*(**"%.5f %c i\*%.5f"**, **real**, **imag** >= 0 ? **'+'** : **'-'**, Math.*abs*(**imag**));  
 }  
  
 */\*\*  
 \* Convert radian to chosen angle unit.  
 \** ***@return*** *Value angle in chosen unit  
 \*/* **private double** conversionAngle(AngleUnit angleUnit, **double** radians) {  
 **switch** (angleUnit) {  
 **case *DEGREE***:  
 **return** Math.*toDegrees*(radians);  
 **case *RADIAN***:  
 **return** radians;  
 **case *GRAD***:  
 **return** radians \* (200d / Math.***PI***);  
 **default**:  
 **return** Double.***NaN***;  
 }  
 }  
  
  
 */\*\*  
 \* Present a complex number in polar form  
 \** ***@param angleUnit*** *- The unit of measurement of the angle at which you want to represent the number  
 \** ***@return*** *String polar (trigonometric) form, example: 2\*(cos(0,52) + i\*sin(0,52))  
 \*/* **public** String toPolarForm(AngleUnit angleUnit) {  
 **double** arg = conversionAngle(angleUnit, getArg());  
  
 **return** String.*format*(**"%.5f\*(cos(%.5f) %c i\*sin(%.5f))"**,  
 getModule(),  
 arg,  
 **'+'**,  
 arg);  
 }  
  
 */\*\*  
 \* Present a complex number in exponent form  
 \** ***@param angleUnit*** *- The unit of measurement of the angle at which you want to represent the number  
 \** ***@return*** *String exponent form, example: 2,00\*e^(i\*0,53)  
 \*/* **public** String toExponentForm(AngleUnit angleUnit) {  
 **double** arg = conversionAngle(angleUnit, getArg());  
  
 **return** String.*format*(**"%.5f\*e^(i\*%.5f)"**, getModule(), arg);  
 }  
  
 @Override  
 **public** String toString() {  
 **return** String.*format*(**"Complex{real=%f, imag=%f}"**, **real**, **imag**);  
 }

@Override  
 **public int** intValue() {  
 **return** (**int**) getModule();  
 }  
  
 @Override  
 **public long** longValue() {  
 **return** (**long**) getModule();  
 }  
  
 @Override  
 **public float** floatValue() {  
 **return** (**float**) getModule();  
 }  
  
 @Override  
 **public double** doubleValue() {  
 **return** getModule();  
 }  
}

3.2 Пример использования главных функций библиотеки

**package** test;  
  
**import** yegorov.math.Complex;  
  
*/\*\*  
 \* Created by Admin  
 \*/***public class** Main {  
 **public static void** main(String[] args) {  
  
 Complex c = Complex.*fromCanonical*(2, -7);  
 Complex c2 = Complex.*fromPolar*(5, Math.*toRadians*(45));  
  
 c.add(c2);  
  
 Complex z = Complex.*mul*(c, Complex.*fromExponent*(5, Math.***PI***));  
  
 System.***out***.println(c.toCanonicalForm());  
 System.***out***.println(z.toPolarForm());  
 System.***out***.println(c2.toExponentForm());  
  
  
 }  
}

*/\*\* Вывод программы  
 \* 5,53553 - i\*3,46447*

*\* 32,65144\*(cos(-2,13001) + i\*sin(-2,13001))*

*\* 5,00000\*e^(i\*0,78540)  
 \*/*