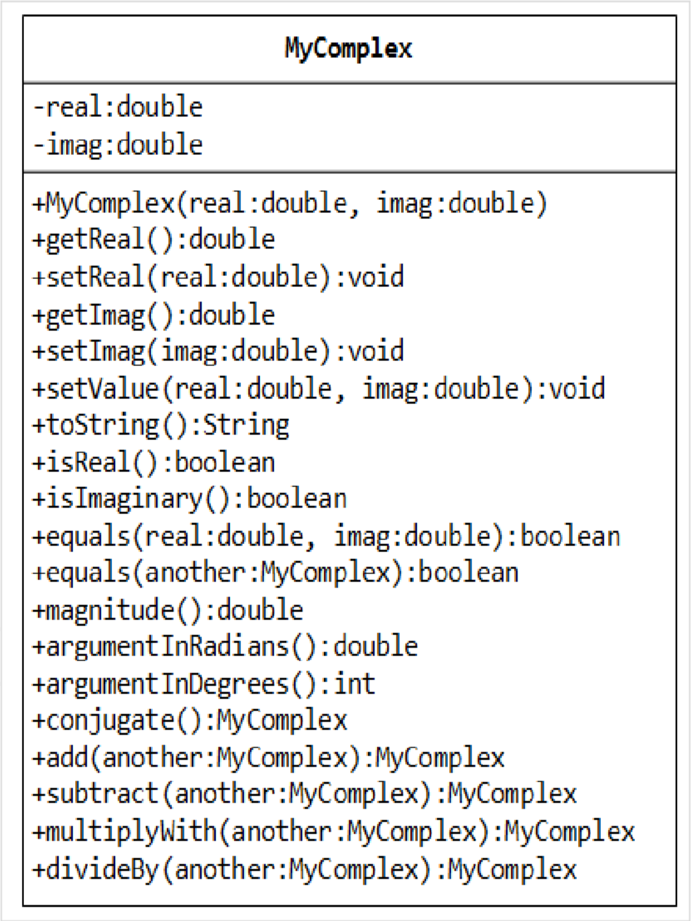
1.1 Exercise: The MyComplex class



```
MyComplex
-real:double
-imag:double
+MyComplex(real:double, imag:double)
+getReal():double
+setReal(real:double):void
+getImag():double
+setImag(imag:double):void
+setValue(real:double, imag:double):void
+toString():String
+isReal():boolean
+isImaginary():boolean
+equals(real:double, imag:double):boolean
+equals(another:MyComplex):boolean
+magnitude():double
+argumentInRadians():double
+argumentInDegrees():int
+conjugate():MyComplex
+add(another:MyComplex):MyComplex
+subtract(another:MyComplex):MyComplex
+multiplyWith(another:MyComplex):MyComplex
+divideBy(another:MyComplex):MyComplex
```

A class called MyComplex, which models complex numbers x+yi, is designed as shown in the class diagram. It contains:

- Two instance variable named real(double) and imag(double) which stores the real and imaginary parts of the complex number respectively.
- A constructor that creates a MyComplex instance with the given real and imaginary values.
- Getters and setters for instance variables real and imag.
- A method setValue() to set the value of the complex number.
- A toString() that returns "(x +yi)" where x and y are the real and imaginary parts respectively.
- Methods isReal() and isImaginary() that returns true if this complex number is real or imaginary, respectively. Hint:

```
return (imag == 0);     // isReal()
```

- A method equals(double real, double imag) that returns true if this complex number is equal to the given complex number of (real, imag).
- An overloaded equals(MyComplex another) that returns true if

  this complex number is equal to the given

  MyComplex instance another.
- A method magnitude()that returns the magnitude of this complex

```
magnitude(x+yi) = Math.sqrt(x2 + y2)
```

- number.
- Methods argumentInRadians() and argumentInDegrees() that returns the argument of this complex number in radians (in double) and degrees (in int) respectively.

```
arg(x+yi) = Math.atan2(y, x) (in radians)
```

Note: The Math library has two arc tangent methods, Math.atan(double) and Math.atan2(double, double). We commonly use the Math.atan2(y, x) instead of Math.atan(y/x) to avoid division by zero. Read the documentation of Math class in package java.lang.

- A method conjugate() that returns a new MyComplex instance containing the complex conjugate of this instance.

```
conjugate(x+yi) = x - yi
```

Hint:

```
return new MyComplex(real, -imag);       // construct a new instance and return the constructed
instance
```

- Methods add(MyComplex another) and subtract(MyComplex another) that adds and subtract this instance with the given MyComplex instance another, and returns a new MyComplex instance containing the result.

  Methods multiplyWith(MyComplex another) and divideBy(MyComplex another) that

```
(a + bi) + (c + di) = (a+c) + (b+d)i
(a + bi) - (c + di) = (a-c) + (b-d)i
```

- multiplies and divides this instance with the given MyComplex instance another, keep the result in this instance, and returns this instance.

```
(a + bi) * (c + di) = (ac - bd) + (ad + bc)i
(a + bi) / (c + di) = [(a + bi) * (c – di)] / (c2 + d2)
```

Hint:

```
return this;        // return "this" instance
```

You are required to:

1. Write the MyComplex class.
2. Write a test program to test all the methods defined in the class.
3. Write an application called MyComplexApp that uses the MyComplex class. The application shall prompt the user for two complex numbers, print their values, check for real, imaginary and equality, and carry out all the arithmetic operations.

Take note that there are a few flaws in the design of this class, which was introduced solely for teaching

```
Enter complex number 1 (real and imaginary part): 1.1 2.2 Enter
complex number 2 (real and imaginary part): 3.3 4.4

Number 1 is: (1.1 + 2.2i)
(1.1 + 2.2i) is NOT a pure real number
(1.1 + 2.2i) is NOT a pure imaginary number

Number 2 is: (3.3 + 4.4i)
(3.3 + 4.4i) is NOT a pure real number
(3.3 + 4.4i) is NOT a pure imaginary number

(1.1 + 2.2i) is NOT equal to (3.3 + 4.4i)
(1.1 + 2.2i) + (3.3 + 4.4i) = (4.4 + 6.6000000000000005i) (1.1 + 2.2i) - (3.3
+ 4.4i) = (-2.1999999999999997 + -2.2i)
```

- purpose: Comparing doubles in equal() using "==" may produce unexpected outcome. For example, (2.2+4.4)==6.6 returns false. It is common to define a small threshold called EPSILON (set to about 10^-8) for comparing

  floating point numbers.

- The method add(), subtract(), and conjugate() produce new instances, whereas multiplyWith()

  and

  divideBy() modify this instance. There is inconsistency in the design (introduced for teaching

  purpose).

- Unusual to have both argumentInRadians() and argumentInDegrees().