**Pseudo Code**

CLASS ComplexNumber INHERITS FROM RealNumber

CLASS FIELDS magPart

Default Constructor: IMPORT None, EXPORT None

super

imagPart = 1.0

Alternate Constructor: IMPORT inNumber, inImagPart, EXPORT None

Super🡨inNumber

imagPart=inImagPart

Copy Constructor: IMPORT inComplexNumber (ComplexNumber), EXPORT None

Super🡨inComplexNumber

imagPart = inComplexNumber.getImagPart

Accessor getImagPart: IMPORT None, EXPORT imagPart

Accessor equals: IMPORT inComplexNumber(ComplexNumber), EXPORT isEqual

tolerance of 0.00001

IF super.equals(inComplexNumber) AND |imagPart - inComplexNumber.getImagPart| < 0.00001 THEN

isEqual = true

ELSE

isEqual = false

END IF

Accessor toString: IMPORT None, EXPORT outStr

outStr = super.toString, “+”, imagPart, “i” (round imagPart to four decimal places)

Mutator setComplexNumber: IMPORT inNumber, inImagPart, EXPORT None

super.setComplexNumber← inNumber

imagPart = inImagPart

Mutator add: IMPORT inComplexNumber (ComplexNumber), EXPORT None

number = super.getNumber + inComplexNumber.getNumber

imagPart = imagPart + inComplexNumber.getImagPart

Mutator subtract: IMPORT inComplexNumber (ComplexNumber), EXPORT None

number = super.getNumber - inComplexNumber.getNumber

imagPart = imagPart - inComplexNumber.getImagPart

Mutator multiply: IMPORT inComplexNumber (ComplexNumber), EXPORT None

tempRealNumber = getNumber

number = (tempRealNumber x inComplexNumber.getNumber – imagPart x inComplexNumber.getImagPart)

imagPart = (imagPart x inComplexNumber.getNumber + tempRealNumber x inComplexNumber.getImagPart)

Mutator divide: IMPORT inComplexNumber (ComplexNumber), EXPORT None

tempRealNumber = getNumber

IF ((inComplexNumber.getNumber>0) AND (inComplexNumber.getImagPart>0)) THEN

otherNumber = inComplexNumber.getNumber

otherImagPart = inComplexNumber.getImagPart

number = (tempRealNumber x otherNumber + imagPart x otherImagPart) ÷ ((otherNumber x otherNumber) + (otherImagPart x otherImagPart))

imagPart = (imagPart x otherNumber - tempRealNumber x otherImagPart) ÷ (otherNumber x otherNumber + otherImagPart x otherImagPart)

END IF

**Java Code**

public class ComplexNumber extends RealNumber

{

private double imagPart;

public ComplexNumber()

{

super();

imagPart=1.0;

}

public ComplexNumber(double inNumber,double inImagPart)

{

super(inNumber);

imagPart=inImagPart;

}

public ComplexNumber(ComplexNumber inComplexNumber)

{

super(inComplexNumber);

imagPart=inComplexNumber.getImagPart();

}

public double getImagPart()

{

return imagPart;

}

public boolean equals(ComplexNumber inComplexNumber)

{

boolean isEqual;

double tolerance=0.00001;

if(super.equals(inComplexNumber) && Math.abs(imagPartinComplexNumber.getImagPart())<tolerance)

{

isEqual=true;

}

else

{

isEqual=false;

}

return isEqual;

}

public String toString()

{

String outStr;

outStr=super.toString() + " + " + Math.round(imagPart\*10000.0)/10000.0 + " i ";

return outStr;

}

public void setComplexNumber(double inNumber, double inImagPart)

{

super.setRealNumber(inNumber);

imagPart=inImagPart;

}

public void add(ComplexNumber inComplexNumber)

{

setRealNumber(super.getNumber()+inComplexNumber.getNumber());

imagPart=imagPart+inComplexNumber.getImagPart();

}

public void subtract(ComplexNumber inComplexNumber)

{

setRealNumber(super.getNumber()-inComplexNumber.getNumber());

imagPart=imagPart-inComplexNumber.getImagPart();

}

public void multiply(ComplexNumber inComplexNumber)

{

double tempRealNumber;

tempRealNumber=getNumber();

setRealNumber(tempRealNumber\*inComplexNumber.getNumber()-imagPart\*inComplexNumber.getImagPart());

imagPart=(imagPart\*inComplexNumber.getNumber()+tempRealNumber\*inComplexNumber.getImagPart());

}

public void divide(ComplexNumber inComplexNumber)

{

double otherNumber;

double otherImagPart;

double tempRealNumber;

tempRealNumber=getNumber();

if((inComplexNumber.getNumber()>0) && (inComplexNumber.getImagPart()>0))

{

otherNumber=inComplexNumber.getNumber();

otherImagPart=inComplexNumber.getImagPart();

   setRealNumber((tempRealNumber\*otherNumber+imagPart\*otherImagPart)/(Math.pow(otherNumber,2)+Math.pow(otherImagPart,2)));

imagPart=(imagPart\*otherNumber-tempRealNumber\*otherImagPart)/(Math.pow(otherNumber,2)+Math.pow(otherImagPart,2));

}

}

}