Série 2- Exercice 2 : Nombre Complexe

Problème 1

1. Définissez une **classe Complexe**, dont chaque instance représente un nombre complexe immuable de l'ensemble ℂ. Un objet complexe aura deux attributs, **une partie réelle** et **une partie imaginaire**: re+i im.

```
public class Complexe {
	// les attributs - la partie réelle et la partie imaginaire
	//Une fois le nombre complexe créé, sa valeur ne change pas.)
	private final double re;
	private final double im;
	}
```

2. La classe Complexe comportera **un constructeur par défaut** qui initialisera les deux attributs à zéro, ainsi qu'**un constructeur** qui initialisera un nombre complexe à partir de deux paramètres réels, et un troisième **constructeur par recopie**.

```
// les constructeurs
	// constructeur sans paramètre
	public Complexe(){
		re = im = 0.0;
	}
	// constructeur avec deux paramètres
	public Complexe(double re, double im){
		this.re = re;
		this.im = im;
	}
	// constructeur par copie
		public Complexe(Complexe x){
		this(x.re,x.im);
	}
```

3. Redéfinir la méthode **toString().** La méthode toString permet la conversion d'un objet de type complexe en une chaîne de caractères.

```
@Override
	public String toString(){
		if (im == 0) return re + "";
		if (re == 0) return im + "i";
		if (im <  0) return re + " - " + (-im) + "i";
		return re + " + " + im + "i";
	}
```

4. Ecrire les accesseurs de la classe (**Les getters**).

```
//Les accesseurs
	public double getRe() {
		return re;
	}
	public double getIm() {
		return im;
	}
```

5. Complétez la classe Complexe avec les opérations arithmétiques: **addition**, **soustraction**, **multiplication**, et **division**.

```
// méthode d'addition
public Complexe addition(Complexe z){
	return new Complexe(this.re+z.re,this.im+z.im);
```

```
	}
// méthode de soustraction
public Complexe soustraction(Complexe z){
	return new Complexe(this.re-z.re,this.im-z.im);
	}
// méthode de multiplication
public Complexe multiplication(Complexe z){
	return new Complexe(this.re*z.re-this.im*z.im,this.re*z.im+this.im*z.re);
	}
//méthode 1 de division
public Complexe division(Complexe c) {
	float reD = (float)((re*c.re+im*c.im)/(Math.pow(c.re,2)+(Math.pow(c.im,2))));
	float imD = (float)((c.re*im-re*c.im)/(Math.pow(c.re,2)+(Math.pow(c.im,2))));
	return new Complexe(reD, imD);
}
//méthode 2 de division
public Complexe division2(Complexe z){
	return new Complexe(this.multiplier(z.inverse()));
}
```

6. Ajoutez à la classe Complexe une méthode statique **Complexe additionner(Complexe a,Complexe b)**.

```
//méthode statique addition (
public static Complexe additionner(Complexe a, Complexe b) {
	return new Complexe(a.re+b.re,a.im+b.im);
}
```

7. Ecrire une méthode qui permet de calculer **le module** ainsi que l'argument d'un nombre complexe.

```
// Le module
public double module() {
	return Math.sqrt(re*re+im*im);
}
// L'argument ou angle thêta    tan x =im/re   angle x=artang (im/re)
public double argument() {
	 return Math.atan(im/re);  //Math.atan2(re, im);
}
```

8. Ecrire une méthode qui permet de **multiplier un nombre complexe par un scalaire**.

```
// multiplier un nombre complexe par un scalaire
 public Complexe multiplierParScalaire(double scal) {
	return new Complexe(scal * re, scal * im);
 }
```

9. Ecrire les méthodes **Inverse**, **conjugue** qui permettent de retourner l'inverse et le conjugué d'un nombre complexe.

```
// la méthode Inverse
public Complexe inverse () {
	return new Complexe(re/(re*re+im*im), -im /(re*re+im*im));
}
//la méthode conjugué
```

```java
public Complexe conjugue() {
	return new Complexe(re, -im);
}
```

10. Ecrire une méthode **exponentielle** qui calcule l'exponentielle d'un Complexe.

```java
//La méthode exponentielle  e^z=e^re*cos(im)+e^re*sin(im) i
public Complexe exponentielle() {
	return new Complexe(Math.exp(re)*Math.cos(im),Math.exp(re)*Math.sin(im));
}
```

11. Redéfinir la méthode **equals()** pour comparer et vérifier si deux nombres complexes sont égaux ou non.

```java
@Override
public boolean equals(Object x) {
	if (x == null)  return false;
	if (this.getClass() != x.getClass()) return false;
	Complexe autre = (Complexe) x;
	return (this.re == autre.re) && (this.im == autre.im);
}
```

12. Ajoutez une méthode main faisant quelques tests élémentaires des complexes et leurs opérations.

```java
public class Main {
	public static void main(String[] args) {
		Complexe a=new Complexe();
		Complexe b=new Complexe(2,4);
		Complexe c= new Complexe(2,4);
		System.out.println(a);
		System.out.println(b);
		System.out.println(c);
		System.out.println("(("+c+") + ("+b+"))="+c.additionner(b));
		System.out.println("(("+c+") + ("+b+"))="+Complexe.additionner(c,b));
		System.out.println("(("+c+") - ("+b+"))="+c.soustraire(b));
		System.out.println("(("+c+") x ("+b+"))="+c.multiplier(b));
		System.out.println("(("+c+") / ("+b+"))="+c.diviser(b));
		System.out.println("Argument de "+b+"="+b.argument());
		System.out.println("Module "+b+"="+ b.module());
		System.out.println("Exponentielle " +b+"="+ b.exponentielle());
		System.out.println(b+" égualité "+c+" : "+b.equals(c));
	}
}
```

Problème 2

En utilisant les principes de la POO résoudre une équation du second degré de la forme : $\mathbf{ax^2+ bx+ c=0}$ dans $\mathbb{C}$. (a, b et c les coefficients du polynôme).

1. Définir une classe **Equation**.

```java
public class Equation {
	private double a,b,c;
```

```
}
```

2. Ecrire **un constructeur à trois paramètres réels** a, b et c qui correspondent aux coefficients du polynôme à résoudre. Ce constructeur affectera les valeurs passées en paramètre aux attributs a, b et c et calculera la valeur du discriminant **delta**.

```
public class Equation {
	private double a,b,c,delta;
	private Complexe z1, z2;
public Equation(double a,double b,double c) {
	this.a=a;
	this.b=b;
	this.c=c;
	delta=b*b-4*a*c;
 }
}
```

3. Ecrire une méthode **afficherDiscriminant()** qui affiche la valeur du discriminant.

```
public void afficherDiscriminant() {
	System.out.println("le Discriminant de l'equation est : " + delta);
}
```

4. Ecrire une méthode **résoudre()** qui résout l'équation dans ℂ. (Utilisez la classe Complexe afin de représenter les racines de l'équation du second degré).

```
public class Equation {
	private double a,b,c,delta;
	private Complexe z1, z2;
	…
public void resoudre(){
// si le discriminant est positif, les solutions sont réelles
	if (delta==0) z2=z1=new Complexe((-b)/(2*a),0);
	else
	if (delta > 0.0){
	z1 = new Complexe((-b - Math.sqrt(delta)) /(2*a),0);
	z2 = new Complexe((-b + Math.sqrt(delta))/(2*a),0);
	}
	else {
// si le discriminant est négatif les solutions sont complexes
	z1 = new Complexe(-b/(2*a),Math.sqrt(-delta)/(2*a));
	z2 = new Complexe(-b/(2*a),(Math.sqrt(-delta)/(2*a))*-1);
	}
}
}
```

5. Ecrire une méthode **afficherSolutions()** qui affiche les solutions de l'équation.

```
public void afficherSolutions(){
	if (delta == 0)
			System.out.println("La racine est " + z1);
	else
		if (delta > 0) {
		// on affiche les racines réelles
			System.out.println("La première racine est " + z1);
			System.out.println("La deuxième racine est " + z2);
			}
```

```
            else {
            // on affiche les solutions complexes
                  System.out.println("La première racine est = "+z1);
                  System.out.println("La deuxième racine est = "+z2);
                  }
}
```

6. Tester votre classe.

```
public class Main {
      public static void main(String[] args) {
            Scanner sc=new Scanner(System.in);
            System.out.println("Equation deuxième degré");
            System.out.println("Donner les coefficients de l'équation:");
            System.out.print("a=");
            double a=sc.nextDouble();
            System.out.print("b=");
            double b=sc.nextDouble();
            System.out.print("c=");
            double c=sc.nextDouble();
            Equation equation = new Equation(a,b,c);
            System.out.println("Votre equation est ( "+a+"x^2+"+b+"x+"+c+"= 0 )");
            equation.afficherDiscriminant();
            System.out.println("Solution de l'equation est :");
            equation.resoudre();
            equation.afficherSolutions();
      }
}
```