Ejemplo de diseño de una clase (I)

- **Ejemplo:** diseño de una clase para manipular números complejos.

  - **atributos:** un número complejo queda totalmente definido por dos valores reales, la parte imaginaria y la parte real, que serán los atributos de la clase. Estos atributos se declaran privados parar ocultar la implementación al exterior y por lo tanto, la clase incorporará métodos para establecer y acceder al valor de los atributos (ver apartado de métodos).

```
public class Complejo
{
    private double re ;
    private double im;
    ...
}
```

  - **constructores:** el constructor de la clase está sobrecargado de manera que los constructores proporcionados son cuatro: un constructor por defecto (sin parámetros), un constructor con dos parámetros (la parte real y la parte compleja), un constructor que convierte un número real en un complejo y un constructor de copia.

Ejemplo de diseño de una clase (II)


```
public Complejo(double r, double i)
   {
      re = r;
      im = i;
   }
public Complejo()
   {
      this(0.0, 0.0);
   }
public Complejo(double d)
   {
      re = d;
      im = 0.0;
   }
public Complejo(Complejo c)
   {
      re = c.re;
      im = c.im;
   }
```

→ Constructor con parámetros

→ Constructor por defecto (sin parámetros)

→ Constructor a partir de un número real

→ Constructor de copia

Ejemplo de diseño de una clase (III)

- **Métodos (i):** los métodos más básicos que proporciona la clase *Complejo* son los métodos de establecimiento y acceso a los atributos privados.

```
public void setRe(double r)
{
    this.re = r;
}
public void setIm(double i)
{
    this.im = i;
}
public double getRe()
{
return this.re;
}
public double getIm()
{
return this.im;
}
```

Establecimiento de la parte real

Establecimiento de la parte imaginaria

Consulta de la parte real

Consulta de la parte imaginaria

Ejemplo de diseño de una clase (IV)

- **Métodos (ii):** funciones de obtención del módulo y argumento de un complejo.

```
public double modulo()  {return Math.sqrt(this.re * this.re + this.im * this.im);}
public double argumento() {
    double arg;
    if (re == 0)
    {
        if (im == 0)
            arg = 0;
        else if (im>0)
                arg = Math.PI/2;
            else arg = 3*Math.PI/2;
     }
     else
     {
        arg = Math.atan(im/re);
        if ( ((re <= 0) & (im >= 0)) | ((re <= 0) & (im <= 0)) )
            arg += Math.PI;
        if ( (re >= 0) & (im <= 0) )
            arg += 2*Math.PI;
     }
    return arg; }
```

Función para la obtención del módulo

Argumento en radianes

Estudio de los casos en los que la parte real es nula.

Estudio de los casos en los que la parte real no es nula.

Ejemplo de diseño de una clase (V)

- **Métodos (iii):** la clase *Complejo* proporciona un conjunto de métodos para realizar operaciones aritméticas (suma, resta, multiplicación y división). Estos métodos se han implementado como métodos de clase.

```
public static Complejo sumar(Complejo c1, Complejo c2)
  {
     double r, i;
     r = c1.re + c2.re;
     i = c1.im + c2.im;
     Complejo c3 = new Complejo(r,i);
     return c3;
  }

  public static Complejo restar(Complejo c1, Complejo c2)
  {
     double r, i;
     r = c1.re - c2.re;
     i = c1.im - c2.im;
     Complejo c = new Complejo(r,i);
     return c;
  }
```

Ejemplo de diseño de una clase (VI)


```
public static Complejo multiplicar(Complejo c1, Complejo c2)
  {
     double r, i;
     r = c1.re * c2.re - c1.im * c2.im;
     i = c1.re * c2.im + c1.im * c2.re;
     Complejo c = new Complejo(r,i);
     return c;
  }

  public static Complejo dividir(Complejo c1, Complejo c2)
  {
     double r, i;
     if (c2.modulo()==0)
        return new Complejo (c1);
     double modulo, argumento;
     modulo = c1.modulo() / c2.modulo();
     argumento = c1.argumento() - c2.argumento();
     r = modulo * Math.cos(argumento);
     i = modulo* Math.sin(argumento);
     Complejo c = new Complejo(r,i);
     return c;
  }
```

Si el divisor es 0+0i, se devuelve el dividendo. Lo más correcto sería generar una excepción.

Ejemplo de diseño de una clase (VII)

- **Métodos (iv):** funciones de obtención del opuesto y el conjugado de un número complejo.

```
public static Complejo opuesto(Complejo c)
  {
     Complejo elOpuesto = new Complejo(-c.re, -c.im);
     return elOpuesto;
  }

  public static Complejo conjugado(Complejo c)
  {
     Complejo elConjugado = new Complejo(c.re, -c.im);
     return elConjugado;
  }
```

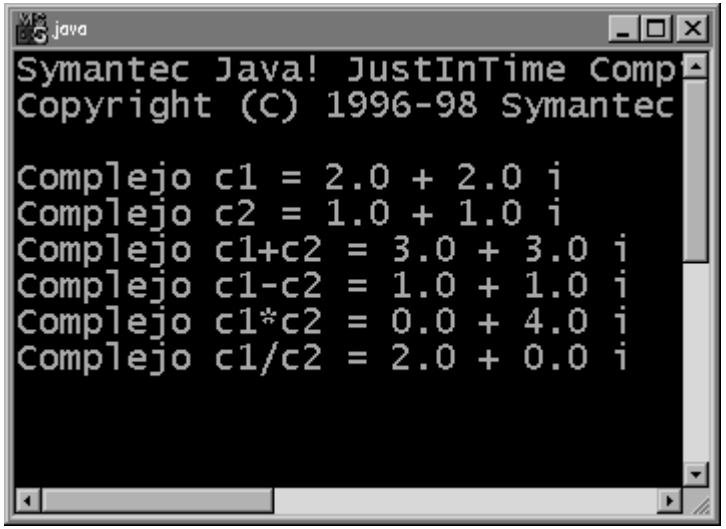
- Reescritura del método **toString** heredado de la clase **Object**.

```
public String toString()
  {
    return (Double.toString(re)+ " + " + Double.toString(im)+ " i ");
  }
```

Ejemplo de diseño de una clase (VIII)

- Ejemplo de uso de la clase.

```
public class MainClass
{
   public static void main (String[]  args)
   {
      Complejo c1, c2, c3;
       c1 = new Complejo (2,2);
       c2 = new Complejo(1,1);
❶  System.out.println("Complejo c1 = " + c1);
❷  System.out.println("Complejo c2 = " + c2);
       c3 = Complejo.sumar(c1,c2);
❸  System.out.println("Complejo c1+c2 = " + c3);
       c3 = Complejo.restar(c1,c2);
❹  System.out.println("Complejo c1-c2 = " + c3);
       c3 = Complejo.multiplicar(c1,c2);
❺  System.out.println("Complejo c1*c2 = " + c3);
       c3 = Complejo.dividir(c1,c2);
❻  System.out.println("Complejo c1/c2 = " + c3);
}}
```



Ejemplo de diseño de una clase (IX)

- **comentarios:** en el proceso de desarrollo de una clase (análisis, diseño e implementación) se toman una serie de decisiones basadas en múltiples criterios. En función de las decisiones adoptadas, el producto final (la clase) será distinto en algunos aspectos. Algunos ejemplos de decisiones tomadas en el desarrollo de la clase *Complejo* son los siguientes:

  - Las operaciones aritméticas de suma, resta, multiplicación y división se han implementado como métodos de clase para que se parezcan en la forma a la notación infija, es decir, con notación infija, el complejo *c3* como resultado de suma de dos complejos *c1 y c2* se expresaría *c3=c1+c2*. Con el método de clase que se ha implementado sería *c3=Complejo.sumar(c1,c2)*. Sin embrago, si se hubiera decidido implementar la suma de complejos como un método de instancia sería *c3=c1.sumar(c2)*, y esta forma es menos semejante a la notación infija.
  - Se ha decidido representar internamente el número complejo mediante su parte real e imaginaria, pero también se podía haber representado mediante su módulo y su argumento.

Ejemplo de diseño de una clase (X)

- La clase proporciona un método que devuelve el módulo del número. Se podría haber decidido calcular el módulo en el momento de la creación del número complejo y guardarlo en un atributo, y devolver este atributo cuando se necesite el módulo del complejo. Si se desarrolla la clase para una aplicación en la que se va a necesitar intensivamente el módulo del número complejo, entonces se optará por calcular ese módulo una sola vez y almacenarlo en un atributo de la clase.