- Sobrecarga de constructores y métodos-
La clase NumComplejo

Contenido

1. Objetivos

En esta práctica de laboratorio aprenderemos:

1. Cómo pasar un problema real a un programa en Java (abstracción). Para ese propósito, un concepto que tú ya conoces será traducido a un programa en Java, que contiene definiciones de clases.
2. Cómo definir varios constructores, con una lista de parámetros distintos (cantidad y/o tipos).
3. La implementación de métodos para una clase.
4. Distinguir entre los métodos que sólo leen la información y los que escriben (cambian) variables de instancia o atributos/campos/propiedades de clase. Es muy importante hacer esta distinción y entender lo que realmente hace un método.
5. Como sobrescribir métodos (mismo nombre, distinto comportamiento).
6. Cómo incluir la documentación Javadoc en Java.

2. La clase NumComplejo

Tendrás que implementar una clase llamada ***NumComplejo*** que representará un número complejo formado por una parte real y una parte imaginaria.
Tendrás que crear un nuevo Proyecto llamado *Lab2Proje*ct. Dentro de este proyecto, crearás un nuevo paquete llamado *PaqComplejo*, este paquete contendrá, entre otras, la clase NumComplejo descrita a continuación.

2.1. Elementos de la clase NumComplejo

La clase ***NumComplejo*** permitirá operaciones típicas con números complejos: suma, resta, producto, etc.

- **Atributos**:
  Tendrá dos atributos, uno que contenga la parte real y otro que contenga la parte imaginara.

- **Constructores**:
  - Constructor sin argumentos: permitirá crear un número complejo con la parte real y la parte imaginaria con valor 0.
  - Otro constructor que recibirá dos valores de **tipo entero,** uno para la parte real y otro para la parte imaginaria.
  - Otro constructor que recibirá dos valores de **tipo doble,** uno para la parte real y otro para la parte imaginaria.
  - Un constructor que permitirá crear un **NumComplejo** a partir de otro **NumComplejo**.

- **Métodos:**
  - Los métodos a los que llamamos *setters y getters,* para establecer y obtener respectivamente los valores de cada uno de los dos atributos (4 métodos en total).
  - Un método llamado **sumar** que sume dos números complejos.
  - Un método llamado **restar** que reste dos números complejos.
  - Un método llamado **multiplicar** que multiplique dos números complejos.
  - Un método llamado **multiplicar** que multiplique un escalar por un número complejo.
  - Un método con el formato **public String toString()** que devuelva el número complejo con un formato agradable.
  - Un método llamado **comparar** que nos devuelva true si dos **NumComplejo** son iguales y false en caso contrario.

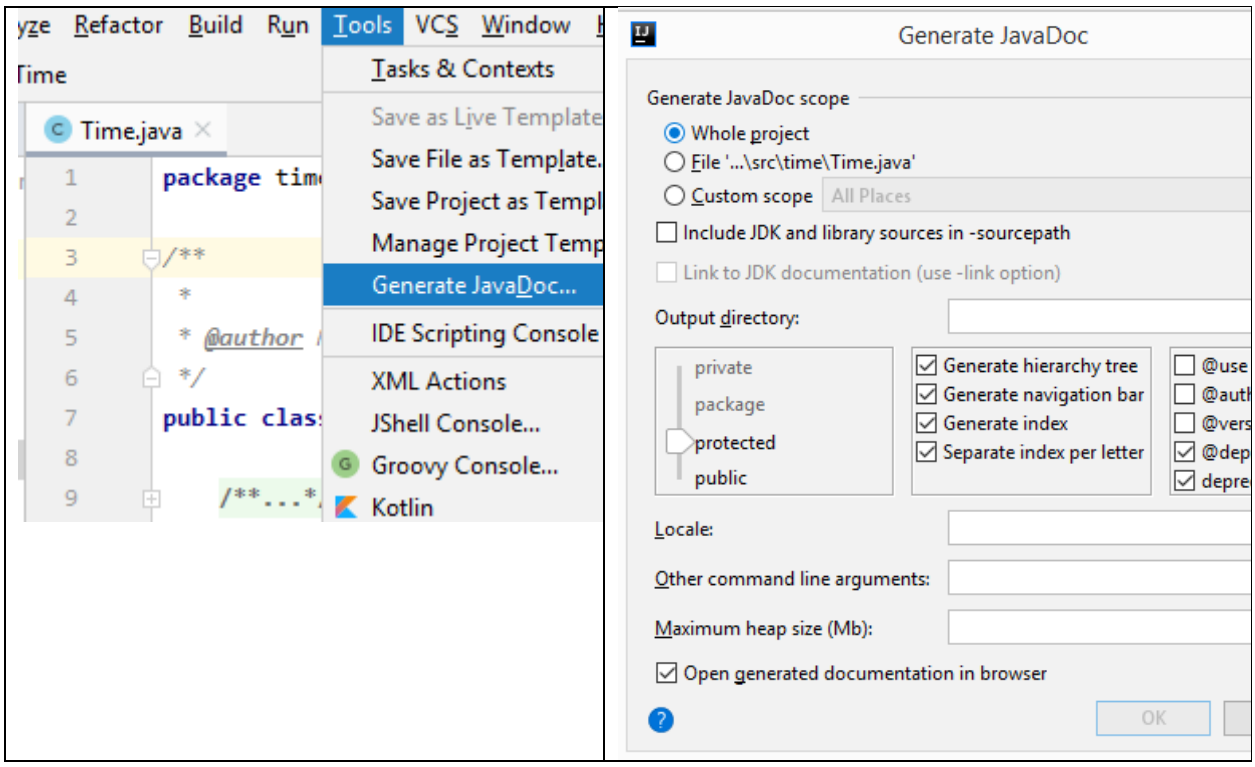
3. Probar la clase NumComplejo

Tendrás que crear una clase principal llamada *NumComplejoMain*, que usará la clase creada previamente y debe verificar su correcto funcionamiento. Crea distintas instancias de la clase **NumComplejo** y llama a todos los métodos para ver que funcionan correctamente. Es importante que compruebes que los constructores son correctos. No olvides añadir más comprobaciones para probar cualquier posible método y circunstancias excepcionales.

4. Generar Javadoc

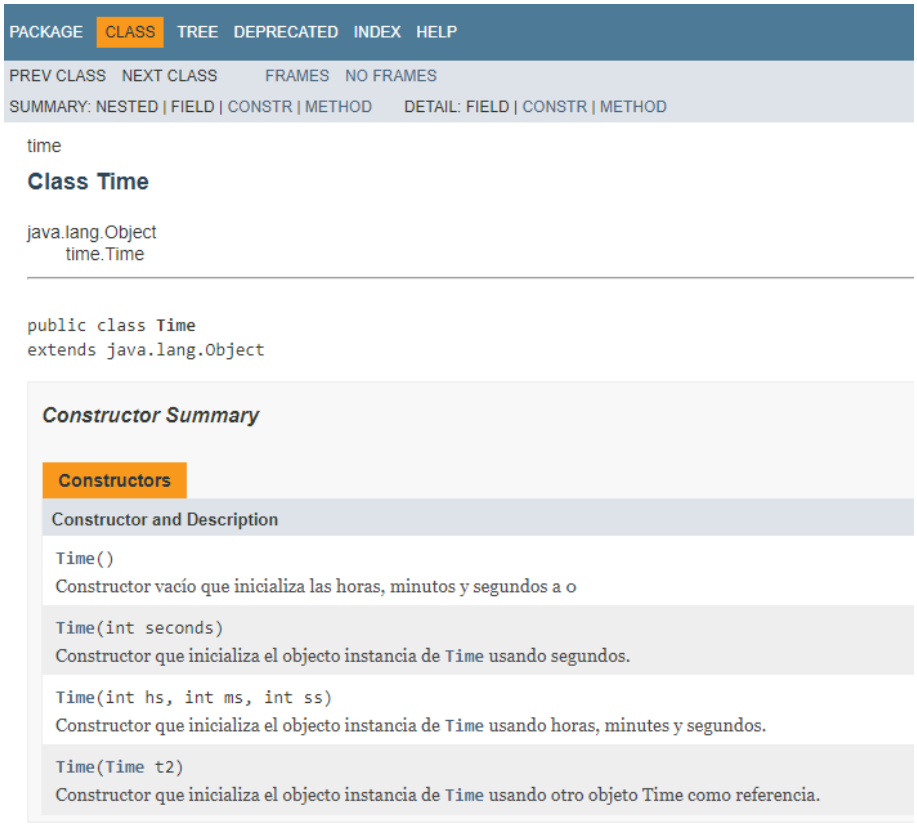
Con ayuda de este enlace y otra información online, por ejemplo, http://www.oracle.com/technetwork/articles/java/index-137868.html, tienes que incluir comentarios de *Javadoc* a tu clase *NumComplejo*.

Para generar la documentación Javadoc desde IntelliJ, solo usa Tools → Generate javadoc. Podremos entonces configurar la generación de la documentación, dando por ejemplo, la carpeta donde se va a guardar la documentación generada (preferiblemente en la carpeta del proyecto; también podemos generar la documenación del proyecto entero o de un archivo). Si quieres que Javadoc acepte tildes en ***Other command line arguments*** tienes que añadir esta secuencia: *-encoding utf8 -docencoding utf8 -charset utf8*



Para más información ver el enlace https://www.jetbrains.com/help/idea/working-with-code-documentation.html

Aquí tienes dos capturas de pantalla para que puedas hacerte una idea de cómo podría ser la documentación generada por Javadoc para una clase llamada **Time** ya implementada:



PACKAGE CLASS TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

time

Class Time

java.lang.Object
time.Time

```
public class Time
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
`Time()` Constructor vacío que inicializa las horas, minutos y segundos a o
`Time(int seconds)` Constructor que inicializa el objecto instancia de `Time` usando segundos.
`Time(int hs, int ms, int ss)` Constructor que inicializa el objecto instancia de `Time` usando horas, minutes y segundos.
`Time(Time t2)` Constructor que inicializa el objecto instancia de `Time` usando otro objeto Time como referencia.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
Time	addition(Time t2) Método que suma un objeto Time al objeto, y devuelve el resultado
void	additionTwo(Time t2) Método que modifica el objeto, sumándole otro objeto Time pasado por parámetro **No devuelve nada, modifica el objeto actual**
Time	difference(Time t2) Método que realiza la diferencia entre mi objeto Time y elparámetro, y devuelve el resultado
void	differenceTwo(Time t2) Método que modifica el objeto, sumándole otro objeto Time pasado por parámetro **No devuelve nada, modifica el objeto actual**
boolean	equals(java.lang.Object o) Método que compara el objeto con el pasado por parámetro
int	getHours() Devuelve las horas del objeto
int	getMinutes() Devuelve los minutos del objeto
int	getSeconds() Devuelve los segundos del objeto
void	setHours(int hours) Da un valor mayor que o a las horas del objeto
void	setMinutes(int minutes) Da un valor (mayor o igual que o y menor que 60) a los minutos del objeto
void	setSeconds(int seconds) Da un valor (mayor o igual que o y menor que 60) a los segundos del objeto
int	toSeconds() Método que devuelve los segundos totales del objeto Time actual
java.lang.String	toString() Método que devuelve el String con la información del objeto en formato hh:mm:ss

Methods inherited from class java.lang.Object

clone, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

5. Conjunto de números complejos.

Ahora implementarás la clase **VariosNumComplejos** (en el mismo paquete de antes*)*, que tendrá un atributo para almacenar un array de instancias de ***NumComplejo***. Este array representará un conjunto de números complejos. Esta clase tendrá los métodos siguientes:

- Constructor **`VariosNumComplejos (int n)`:** para crear un nuevo objeto cuyo array tenga tamaño n, y los números complejos del array del array serán creados aleatoriamente
- **`NumComplejo sumaTodos()`:** Devolverá un objeto de tipo NumComplejo que tendrá como valor la suma de todos los números complejos del array.
- **`String toString()`:** usará el toString de la clase ***NumComplejo*** para recorrer todos los elementos del array y mostrarlos.
- Debes incluir también métodos “setter y getter” para ***VariosNumComplejos***.

En la clase principal tendrás que crear al menos un objeto de esta nueva clase y probar todos los métodos.