Programación Avanzada

Alonso Bernal Alejandro  
López Piedracruz Marcos  
Luviano Murakawa Tsuioshi Alberto

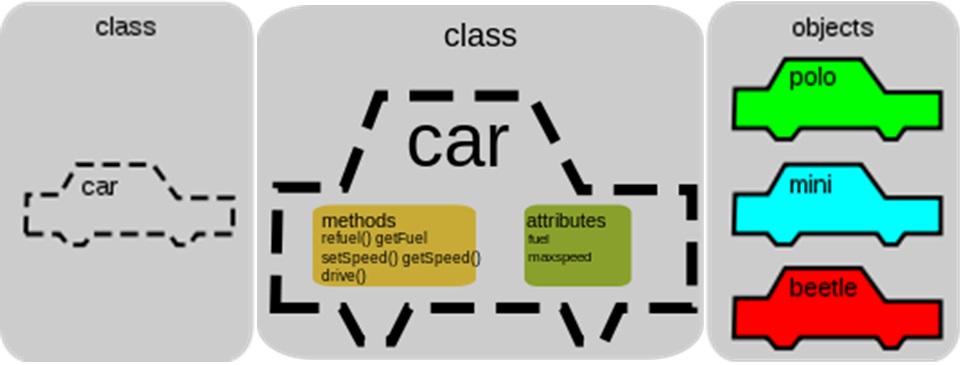
Practica 1

Clases

**Introducción**

El concepto de *clases* en la programación orientada a objetos hace referencia a un prototipo o “molde” de programación que permite al programador instanciar (crear) objetos (de los cuales se hará énfasis en la práctica n° 2) los cuales tendrán las mismas características que la clase original.

Un ejemplo muy común utilizado para entender las clases es el que se muestra a continuación:

Imaginemos una clase llamada automóvil, los rasgos que todos los automóviles comparten entre sí podrían ser algunos tales como color, velocidad máxima, capacidad de combustible, etc. Otra cosa que todos los automóviles tienen en común es el hecho de que todos pueden realizar acciones tales como desplazarse, llenar su deposito de combustible, maniobrar, etc. Pero a pesar de que todos tienen esas características en común, para cada modelo de automóvil habrá una diferencia entre ellos, ya sea si uno es de un color o de otro, si tiene una capacidad mayor para el combustible, o si alcanza una velocidad máxima distinta a la de otros modelos, una vez planteando todo esto se puede ver a una clase como la abstracción de lo que es un automóvil, a los atributos y los métodos de esta como los rasgos y acciones que comparten todos ellos, y un objeto siendo una instanciación de la clase automóvil al cual se le asignan valores específicos para todos los datos que existen dentro de nuestra clase.

La sintaxis general (aunque esta puede llegar a cambiar un poco dependiendo del lenguaje de programación utilizado) es la siguiente:

*<modificador>class<nombre\_clase>  
{  
<declaración\_atributo>  
<declaración\_constructor>  
<declaración\_método>  
}*

Como se mencionó anteriormente, todas las clases tienen miembros los cuales definen en su totalidad a los objetos que se instanciaran por medio de ella, así como también comportamientos que todos los objetos pueden realizar. Los miembros que definen a los objetos son conocidos como atributos y son declarados de la siguiente forma:

<modificador><tipo><nombre> [ = <valor\_inicial>];

Los comportamientos de un objeto definido a través de una clase son conocidos como métodos y son declarados con la siguiente sintaxis:

<modificador><tipo\_retorno><nombre> (<argumentos>) {<sentencia>}

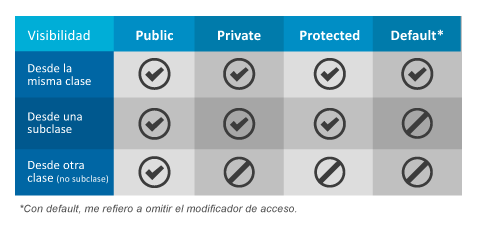
Dentro de todas las clases existe un método común el cual es llamado constructor de la clase.

El constructor es un miembro que implementa las acciones requeridas para inicializar la instancia de una clase. El constructor es invocado cuando se usa el operador new.

El constructor es parecido a los métodos, pero no tiene un tipo de retorno y su nombre es el mismo que el de la clase y también puede o no recibir parámetro como los métodos.

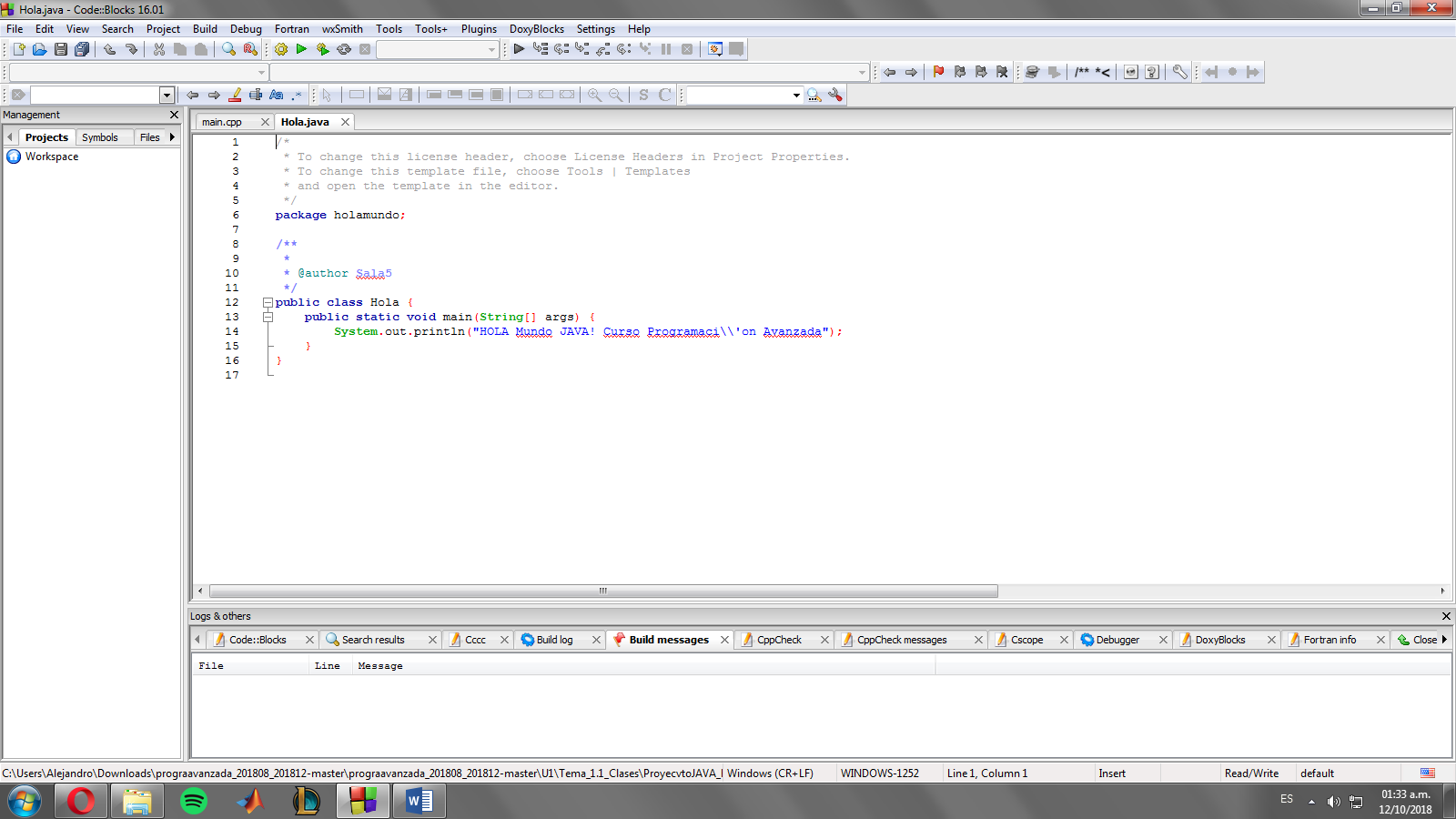
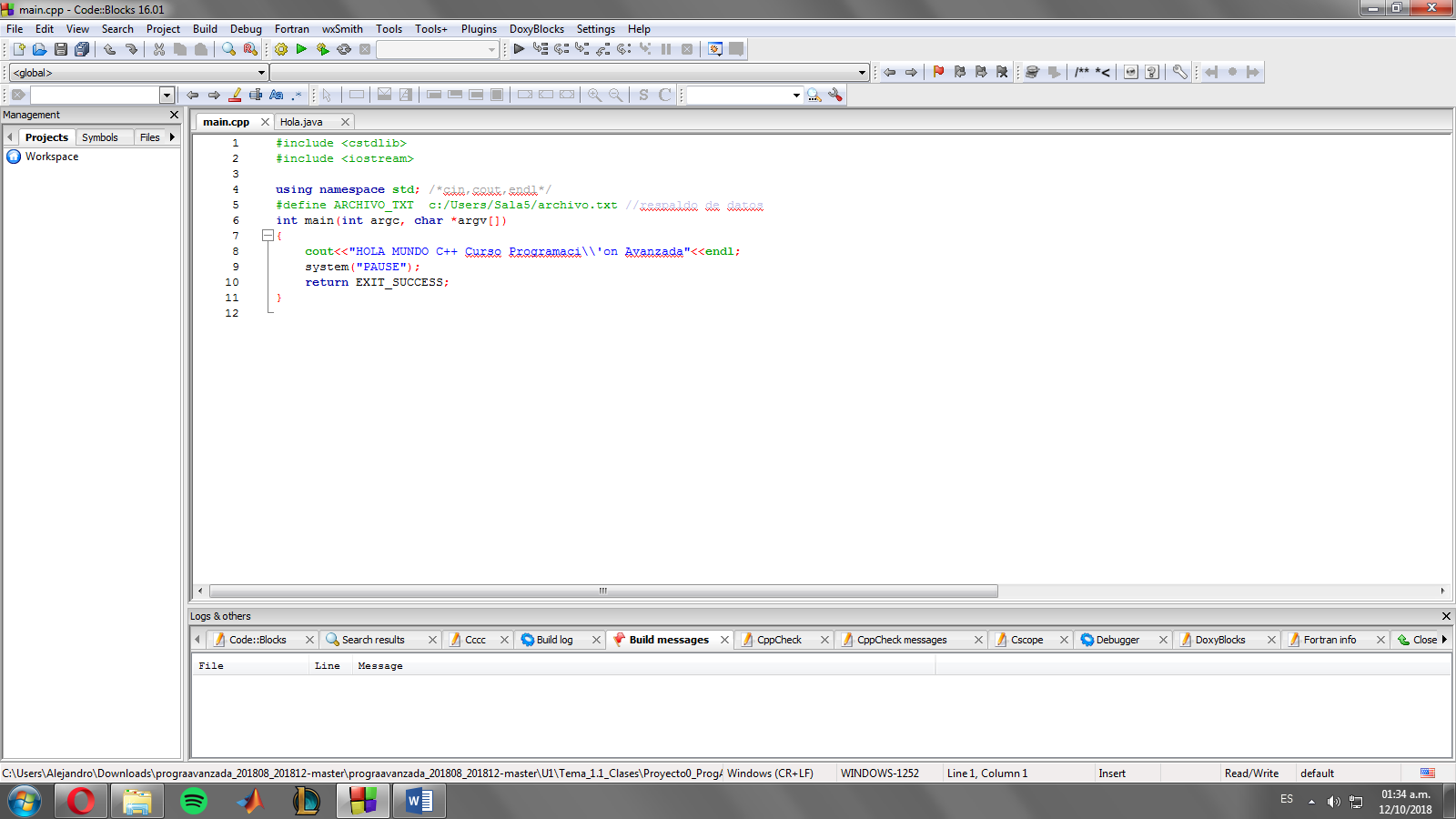
Además de todo lo anterior mencionado, una clase puede contener distintos tipos de variable, entre ellos pueden estar:

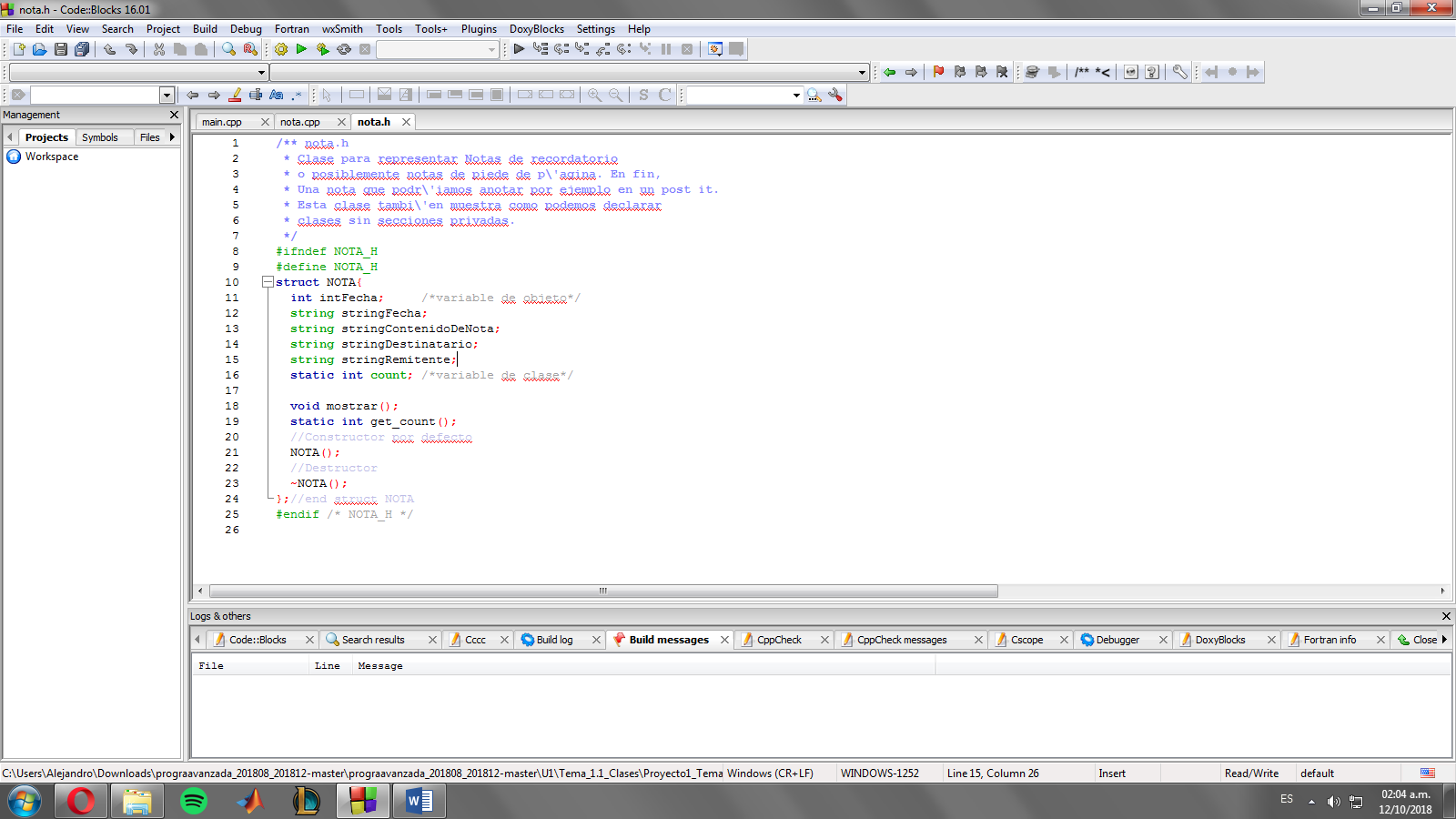
* **De instancia**: Son aquellas que se declaran dentro de una clase y que no existen hasta que se hace una instancia de esa clase (se crea un objeto) <modificador><tipo> nombre [ = <valor\_inicial>];
* **De referencia:** Son aquellas que hace referencia a otra clase. Esto se puede definir mejor diciendo que es una variable del tipo de otra clase, y cuando se haga la instancia tendremos: un objeto dentro de otro objeto. <modificador><Clase> nombre [ = <instancia>];
* **De clase:** Estas variables se declaran anteponiendo la palabra static a su declaración, y a diferencia de las variables de instancia, éstas no necesitan que se haga una instancia (que se cree un objeto), existen desde que se crea la clase. <modificador>static<tipo> nombre [ = <valor\_inicial>];

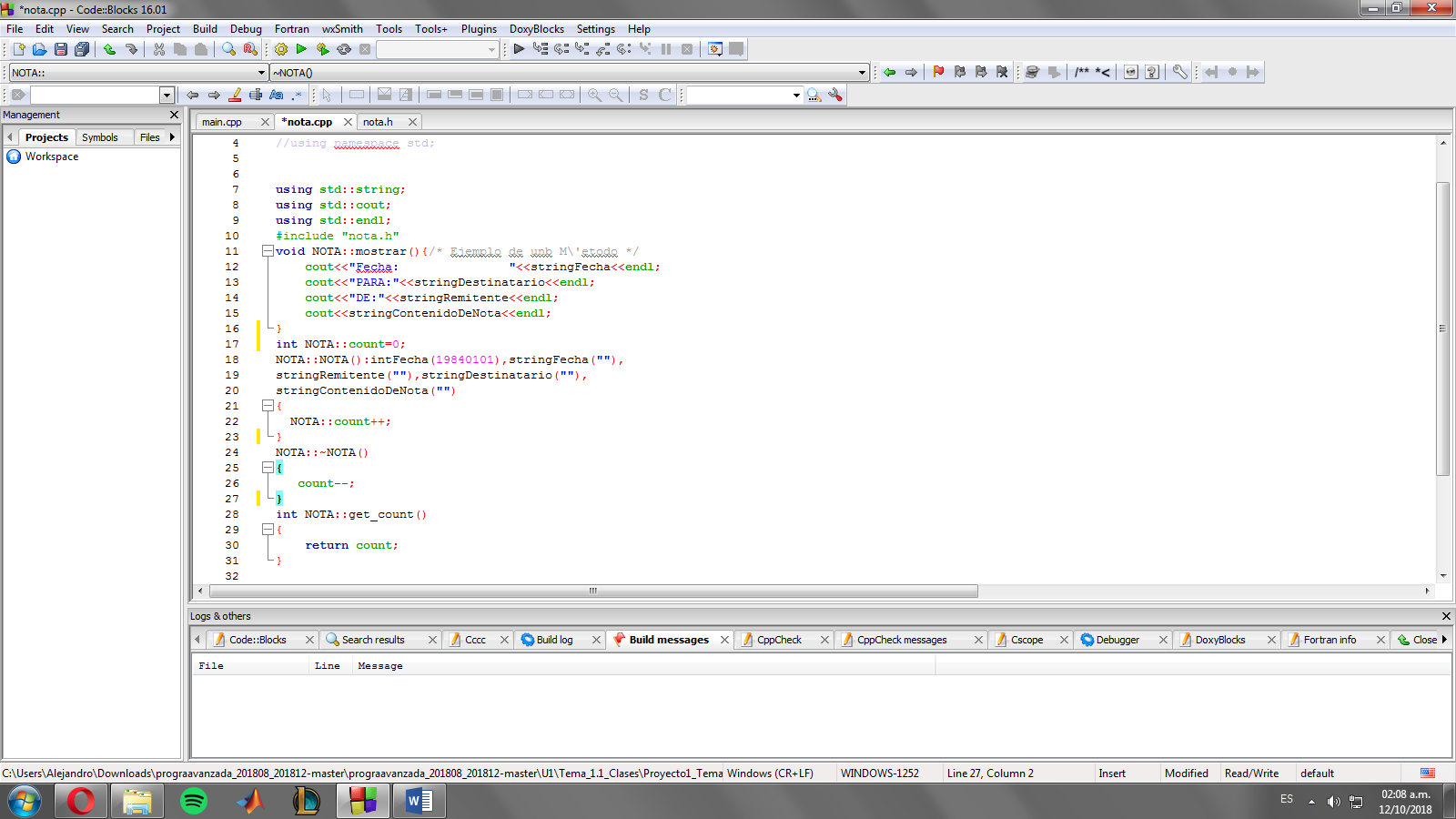
Dentro de todas las sintaxis anteriores se puede observar un llamado “modificador”, estos funcionan para determinar el acceso que se tiene a un tipo de atributo o método.

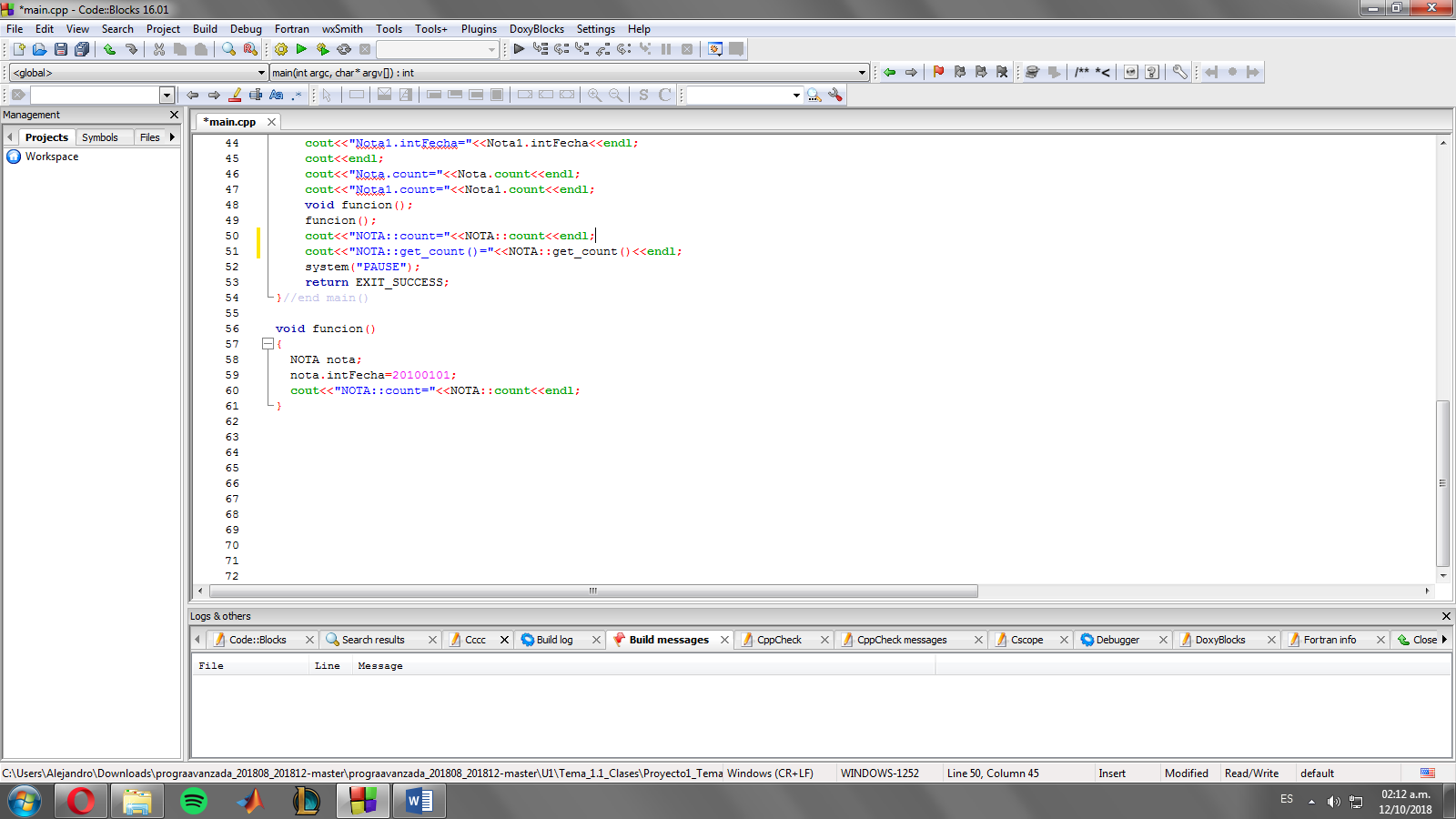
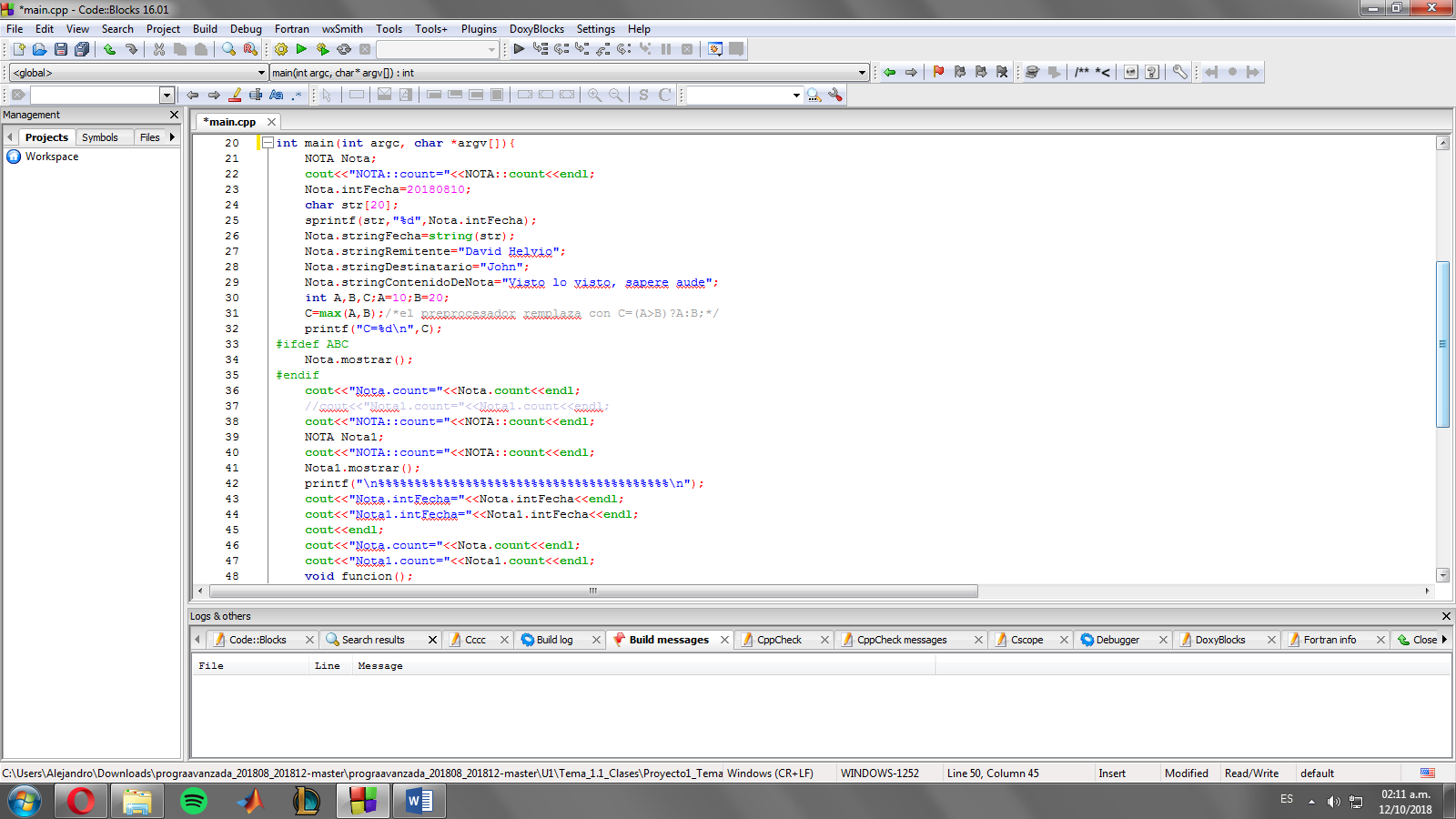
Los modificadores de acceso y los privilegios que contienen se pueden observar en la siguiente tabla:

**Desarrollo**

Los primeros archivos incluidos dentro del desarrollo de esta practica son las declaraciones más básicas para cualquier primer programa: un “hola mundo”, estos están redactados en dos diferentes lenguajes java y C++, los códigos respectivos son los siguientes:

Para expresar el uso de clases dentro de este proyecto se hizo una clase muy básica llamada nota, esta fue declarada como una estructura (lo cual comparte algunas similitudes con una clase), dentro del código se muestran los atributos de la estructura y los métodos que contiene:

Además de la declaración del código del cual se derivarán los demás objetos, el código incluido contiene las acciones que ejecutaran los métodos, además de un código extra para el constructor de la clase el cual funcionara para darse cuenta cuando se haga un llamado al mismo.

Finalmente, en el código principal se crea un objeto de tipo nota al cual se le asignan valores para que el código efectué las acciones definidas en el código anterior.

**Conclusiones:**

Dentro de la programación general, la programación orientada a objetos da lugar a una gran variedad de ventajas, en primer lugar, permitiendo que el código sea reutilizable ya que mediante el modelado de distintos tipos de clases el programador puede hacer que su proyecto utilice algunas partes anteriores para dar lugar a un componente del programa que a pesar de tener una gran similitud con otras partes pueda a llegar a tener un comportamiento totalmente diferente.

Otra de las grandes ventajas del uso de clases y objetos es que resulta una manera de programar bastante sencilla ya que el concepto de abstracción tiende a ser mas sencillo si solamente se tiene que pensar en la “esencia” de las partes principales del código y partir de la idea que se plantea definir la manera en la que se relacionan todas esas partes entre sí.   
Además la facilidad de suprimir o añadir partes a dicho código principal da lugar a la posibilidad de crear cambios significativos a nuestro código sin la necesidad de cambiar completamente la estructura del proyecto.