



## **PROGRAMACIÓN ORIENTADA A OBJETOS**

### **TEMA 2.1** CONCEPTOS DE CLASES Y OBJETOS

#### **ACTIVIDAD 2.1**

**GRUPO:** 2g2A

#### **NOMBRE DE LOS ALUMNOS:**

- ENRIQUEZ MONTALVO RIGOBERTO
- GIL RODRIGUEZ JONATHAN
- GUTIERREZ CRUZ ÁNGEL DE JESÚS
- MORALES VAZQUEZ JUAN DIEGO
- SOSA FIGUEROA BENJAMÍN DE JESÚS

**PROFESORA:** PATRICIA QUITL GONZALEZ

**FECHA DE ENTREGA:** 22/03/2023

**HORA DE CLASE:** 11:00-12:00

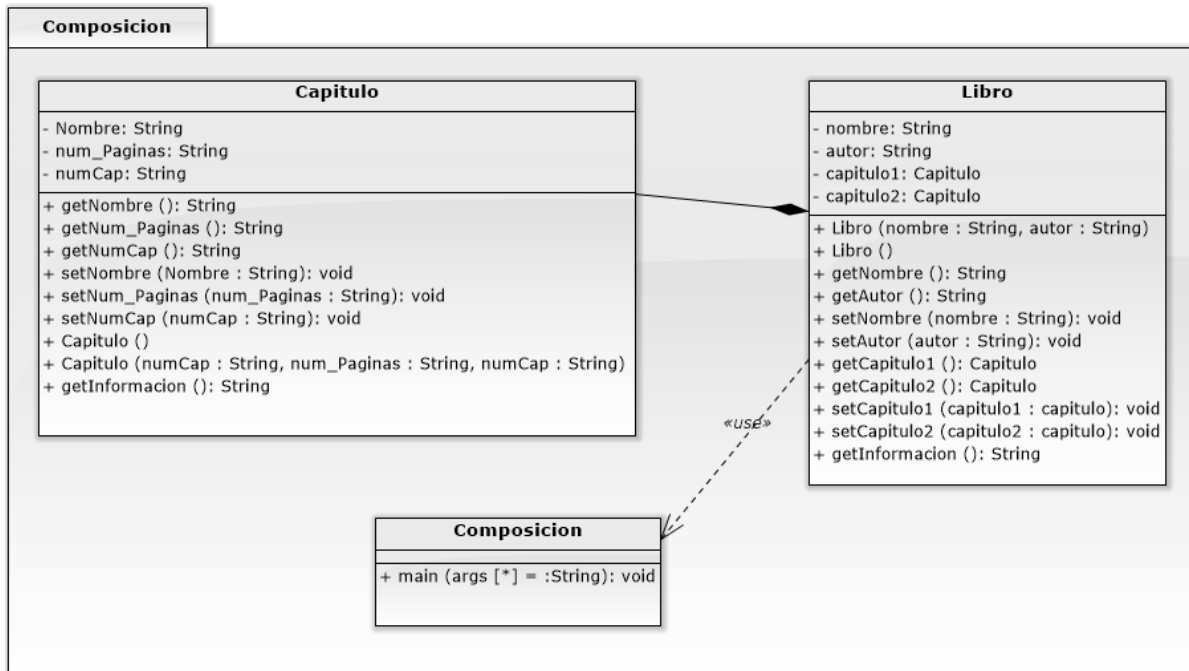


## PROBLEMA 1. COMPOSICIÓN 1

### Definición del problema

Programa que defina las propiedades de composición en las clases Libro y Capitulo, adicionando también una clase Composición la cual se encargue de crear el objeto libro y mostrar su información.

### Diseño de la solución mediante diagrama de clases (UML)



### Casos de prueba

```

Cual es nombre del capítulo 1?
vector
Cuantas páginas tiene el capítulo 1?
45
Cual es nombre del capítulo 2?
álgebra lineal
Cuantas páginas tiene el capítulo 2?
45
|
    
```



#### Datos Libro:

Nombre libro: Como programar en Java  
Nombre autor: Deitel & Deitel

#### Datos capítulo 1:

Nombre capítulo: vector  
Num. hojas: 45

#### Datos capítulo 2:

Nombre capítulo: álgebra lineal  
Num. hojas: 45

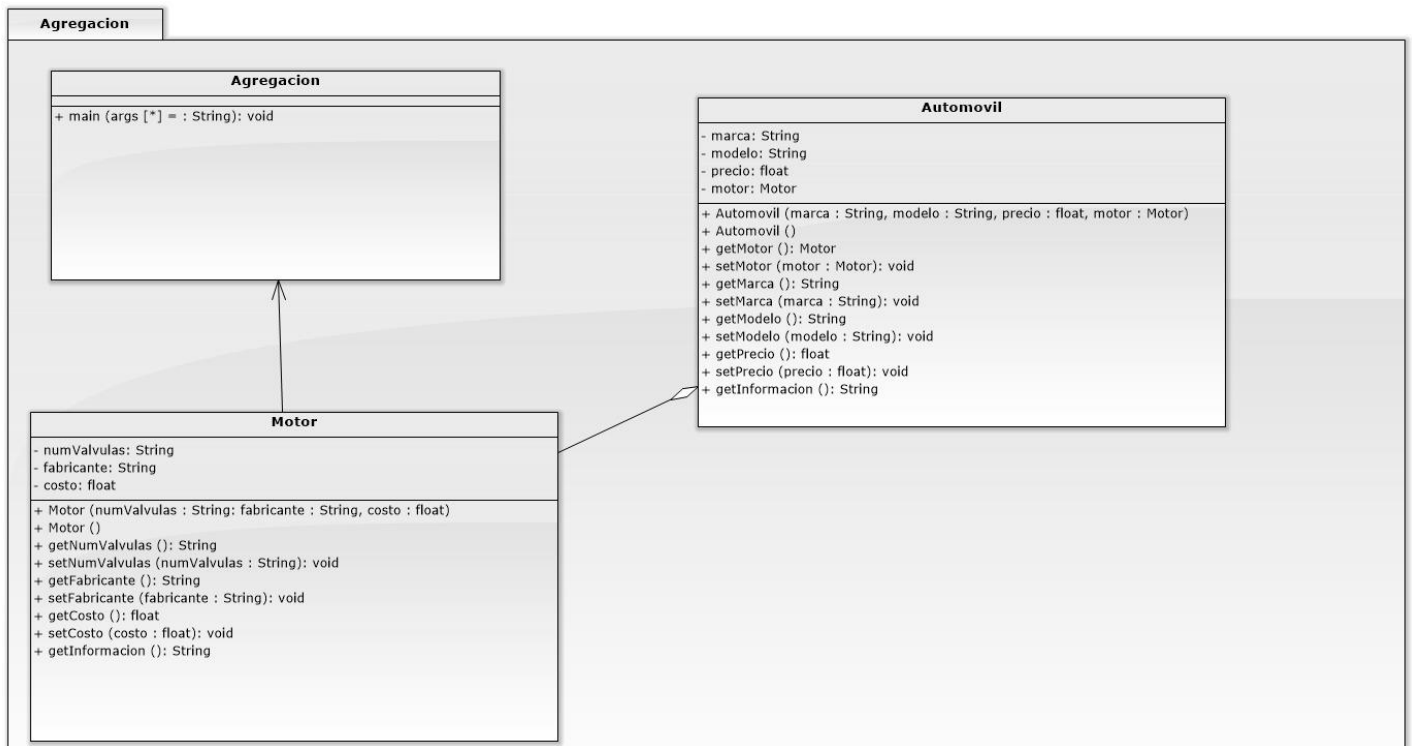
Aceptar

## PROBLEMA 2. AGREGACIÓN 1

### Definición del problema

Programa que defina las propiedades de agregación en las clases Automovil y Motor, adicionando también una clase Agregación la cual se encargue de agregar el motor “gasolina” al auto y muestre su información.

### Diseño de la solución mediante diagrama de clases (UML)



### Casos de prueba

```

: Salida - Agregacion (run)

run:
Ford
4
12500
Datos auto:

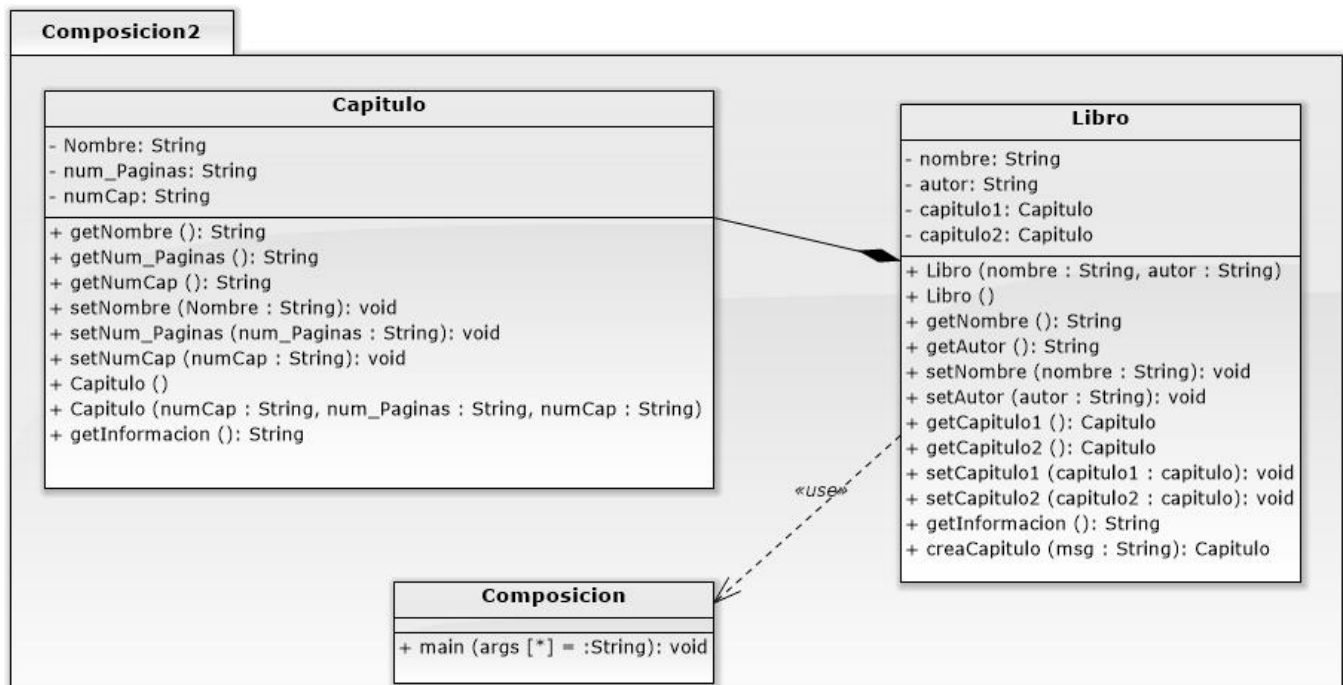
Marca: Toyota
Modelo: Corolla
Precio: 345900.0
Datos motor:
Fabricante: 4
Num. Valvulas: Ford
Costo: 12500.0
    
```

## PROBLEMA 3. COMPOSICIÓN 2

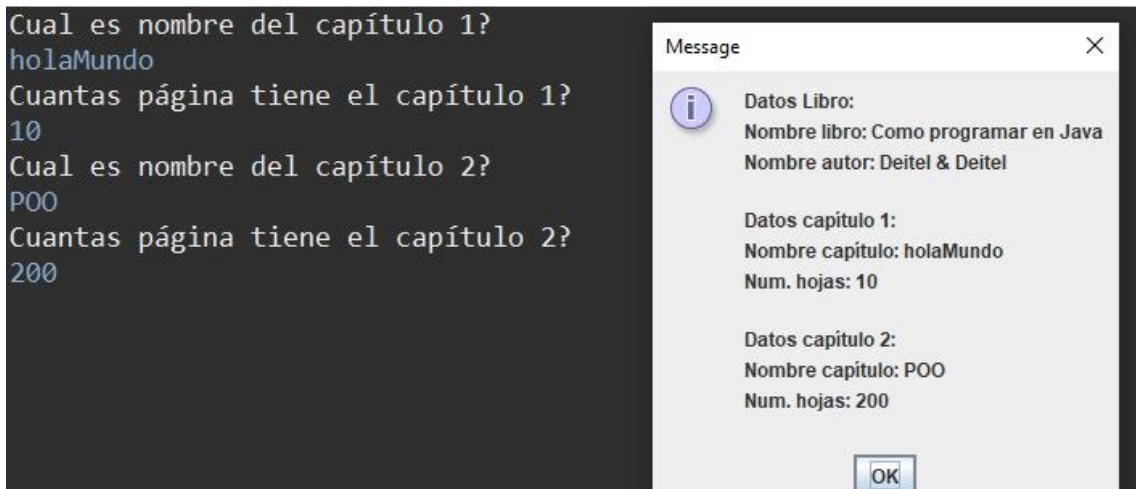
### Definición del problema

Programa que defina las propiedades de composición en las clases Libro y Capitulo, adicionando también una clase Composición la cual se encargue de crear el objeto libro y mostrar su información. Añade el método creaCapitulo en la clase libro para crear un capítulo y que mejore las propiedades de composición.

### Diseño de la solución mediante diagrama de clases (UML)



### Casos de prueba

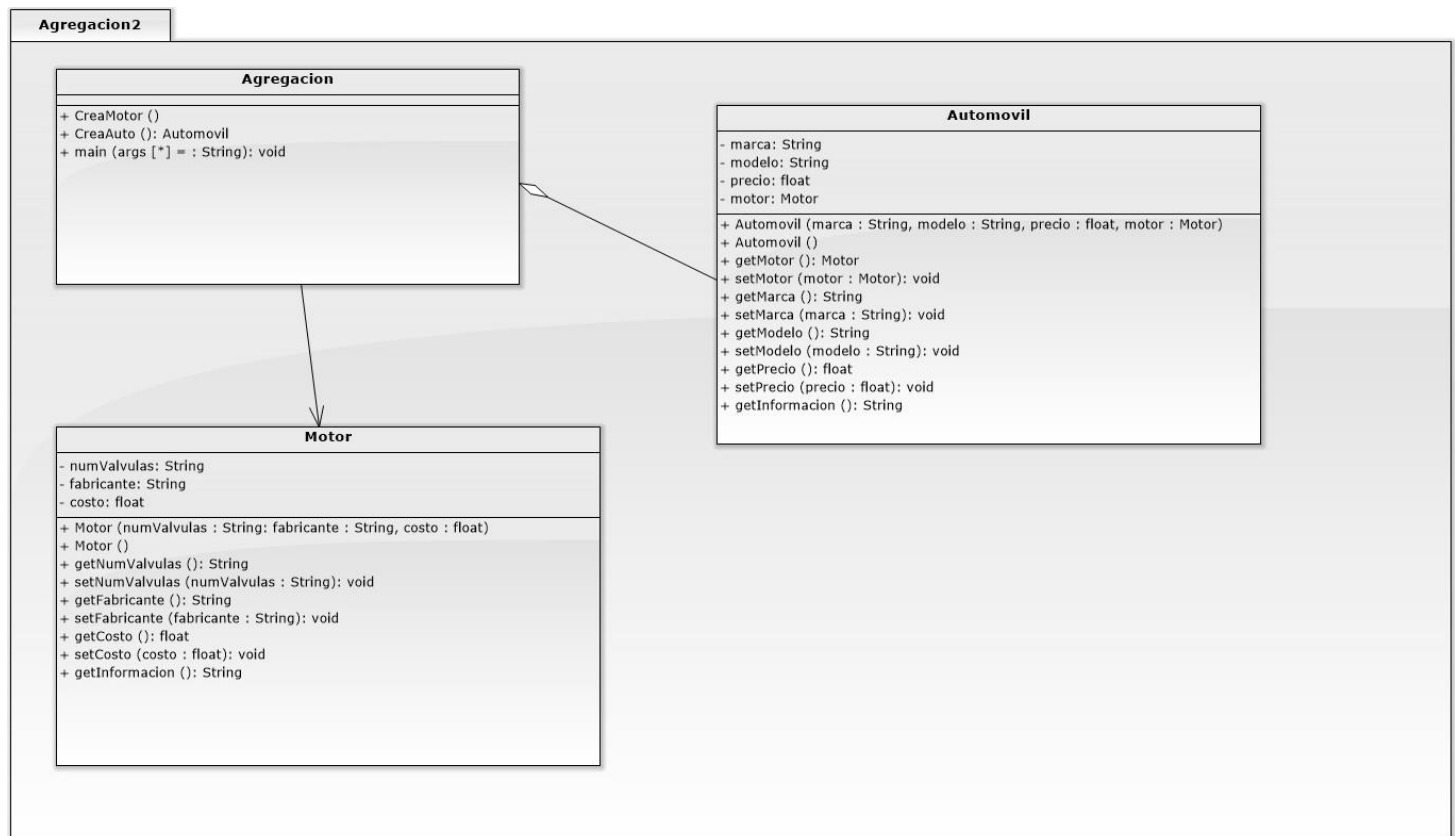


## PROBLEMA 4. AGREGACIÓN 2

### Definición del problema

Programa que defina las propiedades de agregación en las clases Automovil y Motor, adicionando también una clase Agregación la cual se encargue de agregar el motor “gasolina” al auto y muestre su información. Creando dos métodos en la clase agregación para crear el motor y crear el auto.

### Diseño de la solución mediante diagrama de clases (UML)



### Casos de prueba

```

Valvulas?
16
Fabricante?
ToyotaGroup
Costo?
4399.99
Datos auto:
  
```

```

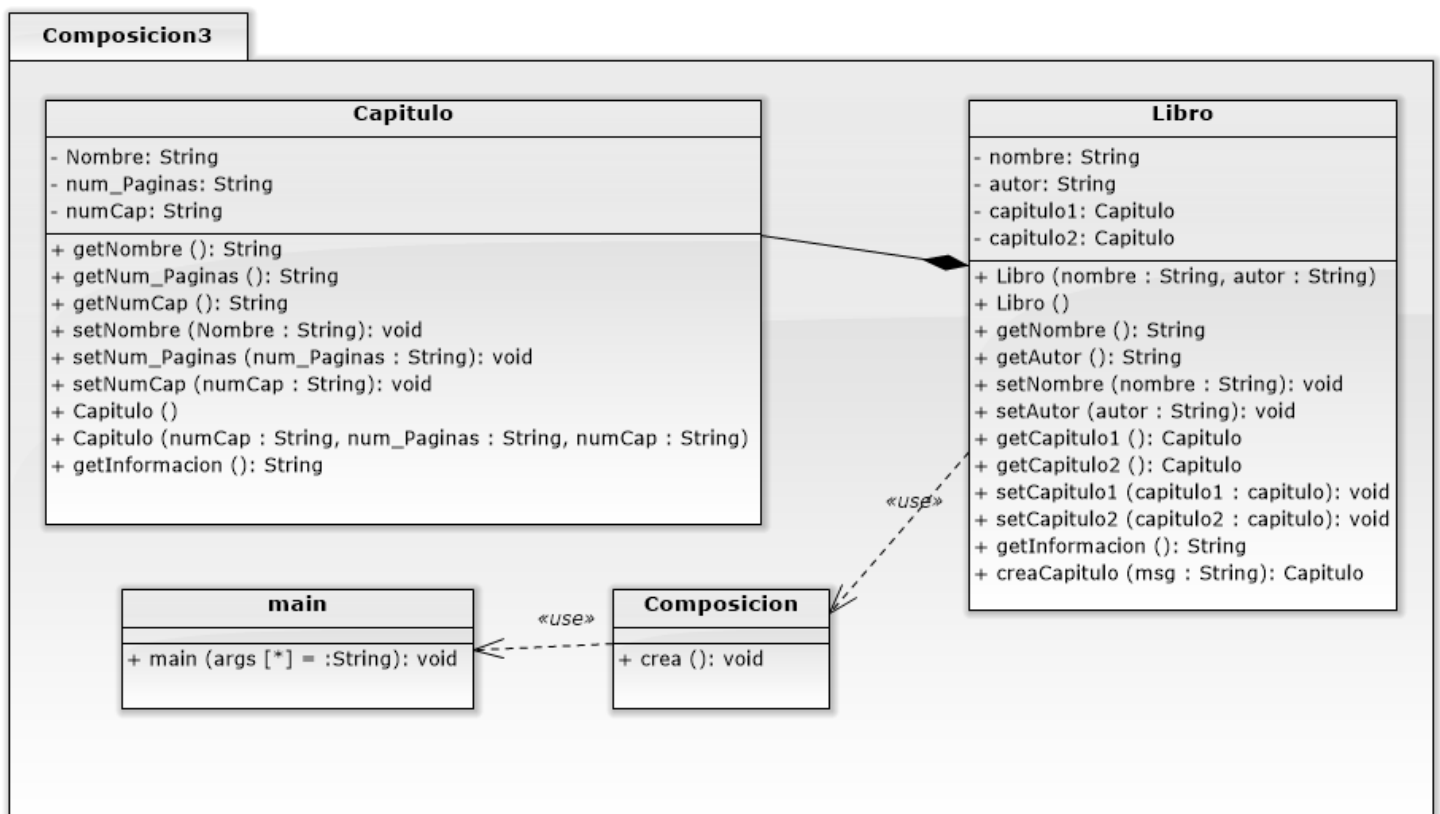
Marca: Toyota
Modelo: Corolla
Precio: 345900.0
Datos motor:
Fabricante: ToyotaGroup
Num. Valvulas: 16
Costo: 4399.99
  
```

## PROBLEMA 5. COMPOSICIÓN 3

### Definición del problema

Programa que defina las propiedades de composición en las clases Libro y Capitulo, adicionando también una clase Composición la cual se encargue de crear el objeto libro y mostrar su información. Añade el método creaCapitulo en la clase libro para crear un capítulo y que mejore las propiedades de composición. Además, crea una clase main la cual solo se encargue de llamar a la clase composición la cual tiene dentro el método crea().

### Diseño de la solución mediante diagrama de clases (UML)



## Casos de prueba

```

Cual es nombre del capítulo 1?
Hola
Cuántas páginas tiene el capítulo 1?
20
Cual es nombre del capítulo 2?
Mundo
Cuántas páginas tiene el capítulo 2?
30
        
```

Message X

**Datos Libro:**  
Nombre libro: Hola Mundo  
Nombre autor: Enriquez Rigo

**Datos capítulo 1:**  
Nombre capítulo: Hola  
Num. hojas: 20

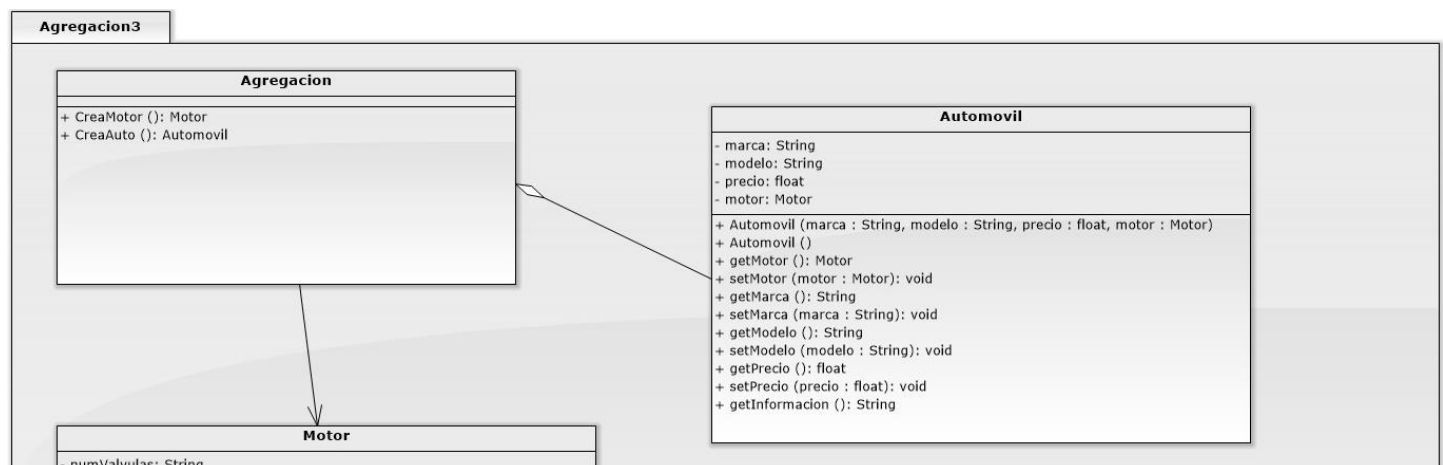
**Datos capítulo 2:**  
Nombre capítulo: Mundo  
Num. hojas: 30

## PROBLEMA 6. AGREGACIÓN 3

### Definición del problema

Programa que defina las propiedades de agregación en las clases Automovil y Motor, adicionando también una clase Agregación la cual se encargue de agregar el motor “gasolina” al auto y muestre su información. Creando dos métodos en la clase agregación para crear el motor y crear el auto. Además, crea una clase main la cual solo se encargue de llamar a la clase Motor.

### Diseño de la solución mediante diagrama de clases (UML)





```
- numValvulas: String
- fabricante: String
- costo: float

+ Motor (numValvulas : String, fabricante : String, costo : float)
+ Motor ()
+ getNumValvulas (): String
+ setNumValvulas (numValvulas : String): void
+ getFabricante (): String
+ setFabricante (fabricante : String): void
+ getCosto (): float
+ setCosto (costo : float): void
+ getInformacion (): String
```

```
Main

+ main (args [*] = : String): void
```

## Casos de prueba

```
Valvulas?
18
Fabricante?
Toyota
Costo?
5000.99
Datos auto:

Marca: Toyota
Modelo: Corolla
Precio: 345900.0

Datos motor:
Fabricante: Toyota
Num. Valvulas: 18
Costo: 5000.99
```

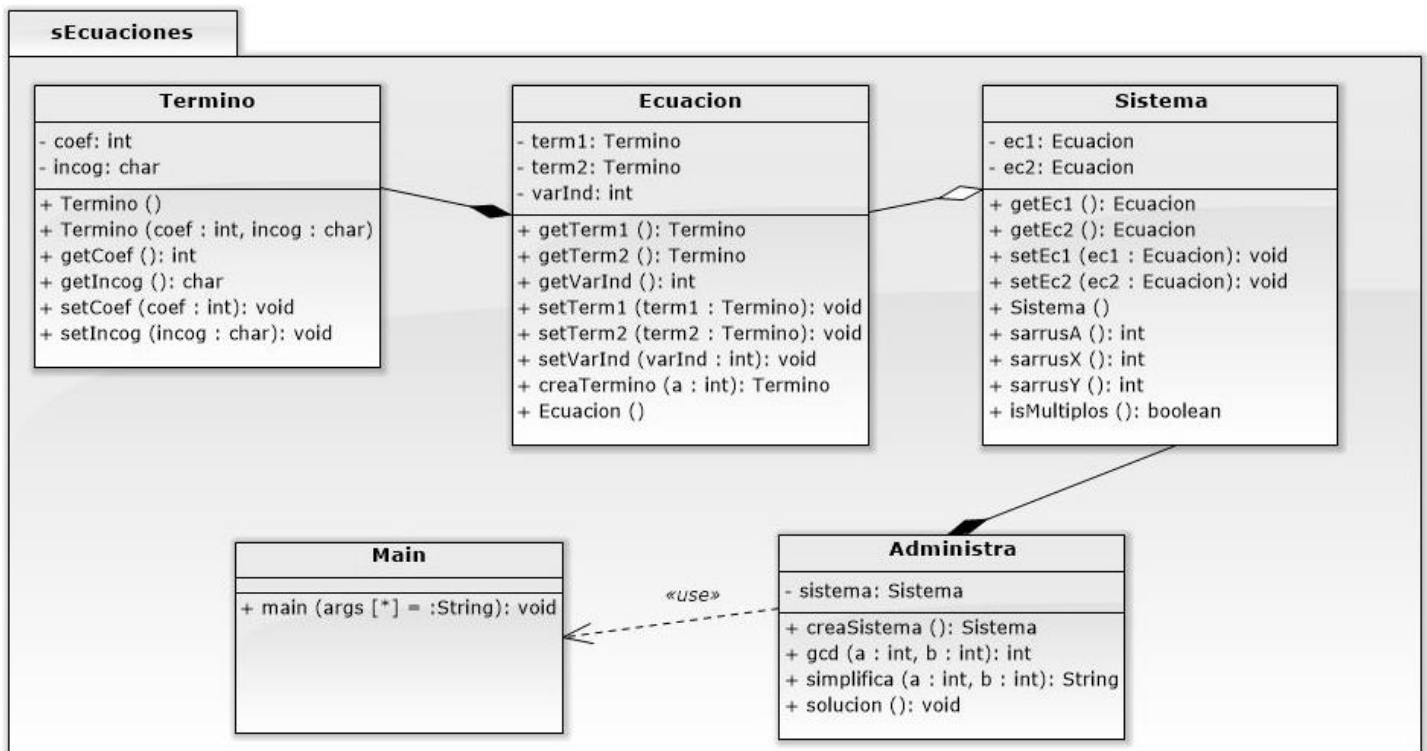


## PROBLEMA 7. REGLA DE CRAMER

### Definición del problema

Realiza un programa para calcular la solución de un sistema de ecuaciones de 2x2 utilizando la regla de Cramer, aplicando también la regla de Sarrúz para calcular el determinante de la matriz.

### Diseño de la solución mediante diagrama de clases (UML)



### Casos de prueba

Input X

?

Coeficiente de termino 1 ?

2

OK

Cancel

Input X

?

Incognita de termino 1 ?

x

OK

Cancel

Input

Coeficiente de termino 2 ?

3

OK Cancel

Input

Incognita de termino 2 ?

y

OK Cancel

Input

Termino Independiente ?

20

OK Cancel

Input

Coeficiente de termino 1 ?

1

OK Cancel

Input

Incognita de termino 1 ?

x

OK Cancel

Input

Coeficiente de termino 2 ?

-2

OK Cancel

Input

Incognita de termino 2 ?

y

OK Cancel


Input

Termino Independiente ?

3

OK Cancel

Message

  $x = 7, y = 2$

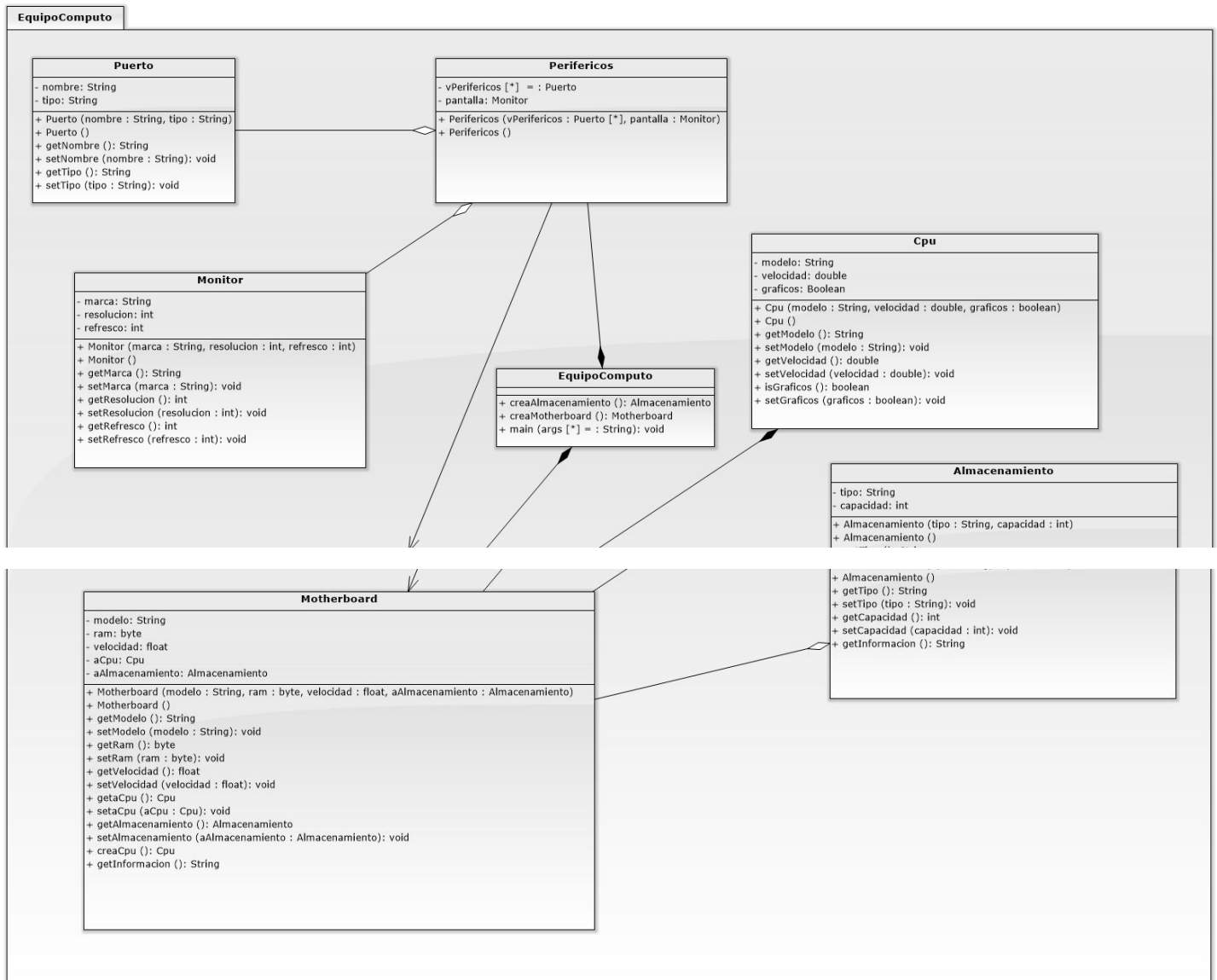
OK

## PROBLEMA 8. EQUIPO DE CÓMPUTO

### Definición del problema

Realiza un programa el cual incluya los elementos que tiene un equipo de cómputo, utilizando composición o agregación según corresponda la relación de cada componente. Mostrar la información del equipo de cómputo.

### Diseño de la solución mediante diagrama de clases (UML)



## Casos de prueba

The image displays a series of Java Swing dialog boxes used for data entry and a final message box. The input boxes are titled 'Input' and contain a green question mark icon. The message box is titled 'Message' and contains an information icon.

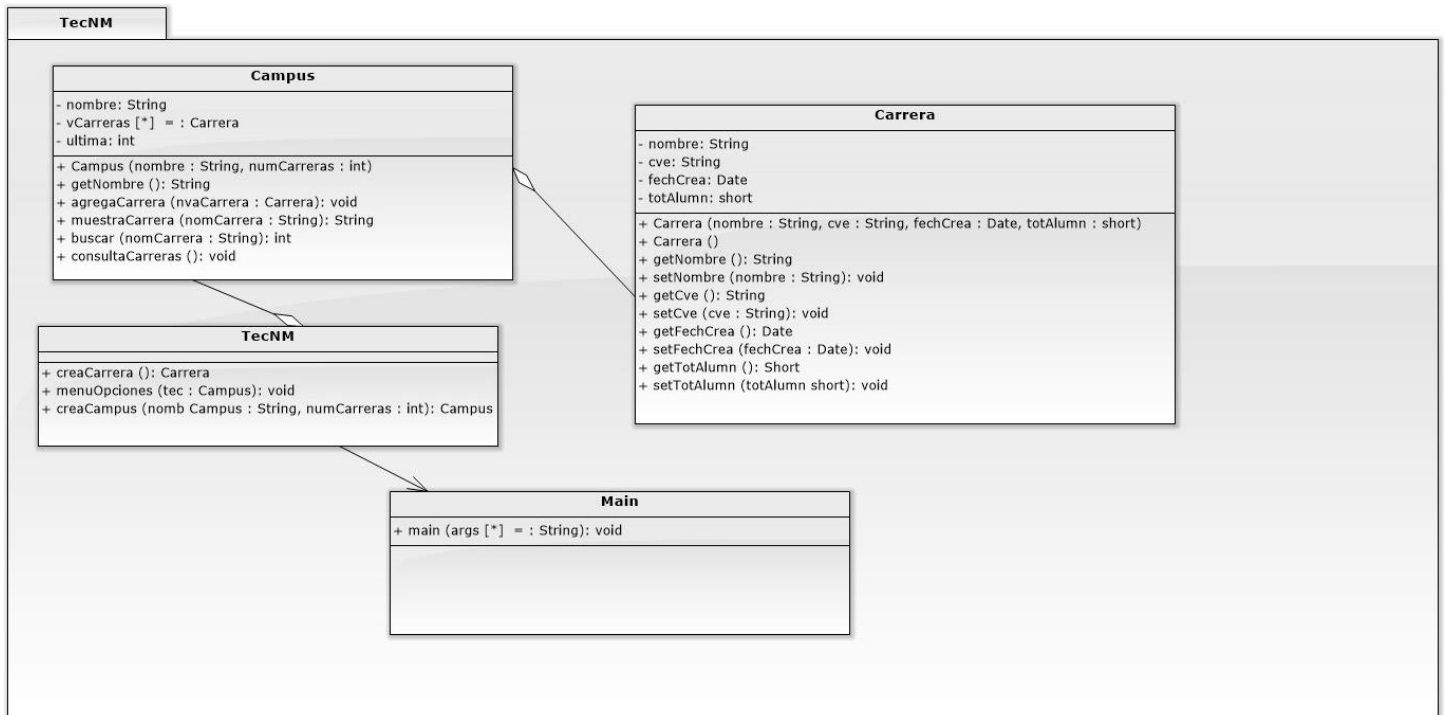
- Input Dialog 1:** "Cual es el modelo de la MotherBoard?" with the text "Intell" entered.
- Input Dialog 2:** "Cuanta RAM tiene la MotherBoard?" with the text "4" entered.
- Input Dialog 3:** "Cual es la velocidad de la MotherBoard?" with the text "2.4" entered.
- Input Dialog 4:** "Tipo de almacenamiento?" with the text "ssd" entered.
- Input Dialog 5:** "Cual es su capacidad?" with the text "125" entered.
- Input Dialog 6:** "Cual es el modelo del CPU?" with the text "intel" entered.
- Input Dialog 7:** "Cual es la velocidad del CPU?" with the text "2.4" entered.
- Input Dialog 8:** "Tiene tarjeta grafica incluida?" with the text "true" entered.
- Message Dialog:** "Datos de la MotherBoard" with the following details:
  - Modelo: Intell
  - RAM: 4
  - Velocidad: 2.4
  - Modelo del CPU: intel
  - Velocidad del CPU: 2.4000000953674316
  - Tiene graficos: true
  - Datos Almacenamiento:
  - Tipo de almacenamiento: ssd
  - Capacidad de almacenamiento: 125

## PROBLEMA 9. UNIVERSIDAD TecNM

### Definición del problema

Realiza un programa el cual haga uso de un CRUD para administrar los distintos campus del TecNM y las carreras del campus las cuales se almacenarán en un vector de tipo Carrera.

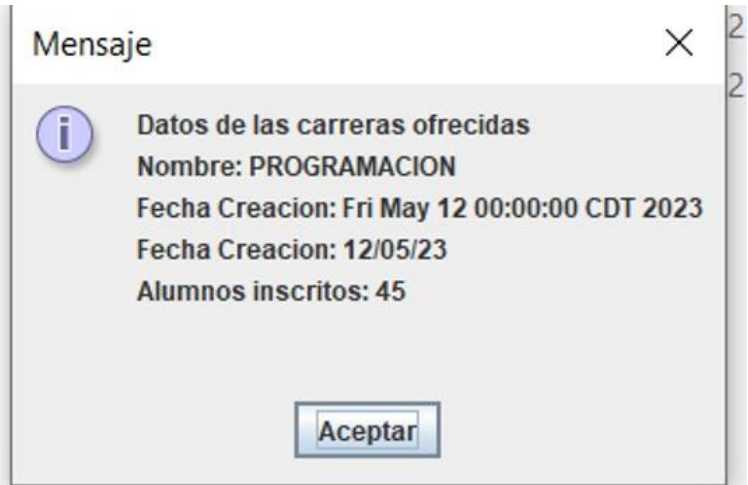
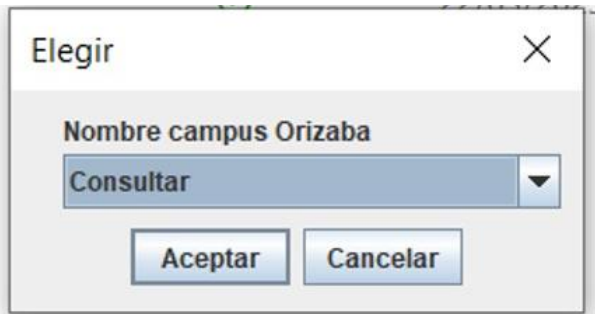
## Diseño de la solución mediante diagrama de clases (UML)



## Casos de prueba

The following screenshots show the input steps for adding a new career:

- Entrada:** Nombre de la carrera a dar alta?
- Entrada:** Clave de la carrera a dar de alta?
- Entrada:** Numero de alumnos que tiene la carrera?
- Entrada:** Fecha creacion dd/mm/yyyy



## Conclusiones

Enríquez Montalvo Rigoberto: En esta nueva unidad reforcé los conocimientos previos obtenidos en la primera, además de agregar dos nuevos conceptos: agregación y composición, dónde el primero se caracteriza por tener una dependencia ajena al programa general, es decir, su presencia es opcional; mientras que, el segundo se muestra como un requisito fundamental para que el programa jale correctamente con base a la función que este haga.

Gil Rodriguez Jonathan: En este tema nuevo aprendí cómo funciona la agregación y la composición en los diagramas UML y en parte como funcionan en la programación orientada a objetos, ya que estos conceptos se usan principalmente para que los objetos creados se asemejen un poco más a lo que es la realidad, lo cual es el objetivo principal de la programación orientada a objetos a mi parecer. Con los programas realizados en clase aprendí que el concepto de agregación se usa principalmente cuando las clases tienen poca dependencia unos de otros, y estos se representan con un diamante blanco. El concepto de composición se usa cuando dos clases tienen una fuerte dependencia, ya que una no podría existir sin la otra y se representa con un diamante de color negro. Estos dos conceptos nos ayudan a que nuestras clases estén mejor definidas, y nos servirán mucho en los temas que continúan.

Gutierrez Cruz Ángel de Jesus: En este nuevo tema de POO pudimos observar un nuevo comportamiento de las clases, al agregar los constructores con parámetros los cuales agregarán una gran dependencia de unas clases a otras, lo cual permite, crear objetos dentro de otras



clases, con el fin de que si el objeto de esta se elimina, se eliminé todo el conjunto de datos pertenecientes a la misma, y no solo se borre una pequeña parte de la misma.

Morales Vázquez Juan Diego: En este tema nuevo del paradigma orientado a objeto vimos un más sobre la naturaleza de las clases y los objetos, vimos lo que es agregación y composición, dos subtemas que a simple vista me confundieron, ya que van de la mano de la abstracción, pero llevándolo más profundo llegando a comprender la naturaleza del objeto a modelar, a base de programación y práctica logre comprender más de este proceso de llevar un objeto real a este paradigma y representarlo mediante diagramas de clases donde resaltamos lo que es composición y agregación.

Sin duda será algo que en posteridad ocuparemos y complementa mucho la comprensión de diagramas al momento de pasarlos a código, me parece interesante que un dilema muy grande dentro del paradigma orientado a objetos sea la abstracción, cosa que se puede tomar a la ligera y a primeras parece "fácil" pero es importante analizar lo que se quiere hacer antes de programar para que este proceso sea más sencillo.

Sosa Figueroa Benjamín de Jesus: En este tema entendí 2 nuevos conceptos principalmente que es la agregación y composición, además de poder implementar una nueva solución a diferentes códigos, como los métodos sobrecargados que no sabía que se podían realizar, esto me abrió un panorama más amplio para resolver diferentes problemas, claro con complejidades referente a los constructores principalmente, pero con su debido estudio y esfuerzo fue posible afrontar los problemas propuestos.