Trabajo Práctico 2
Clases y Objetos

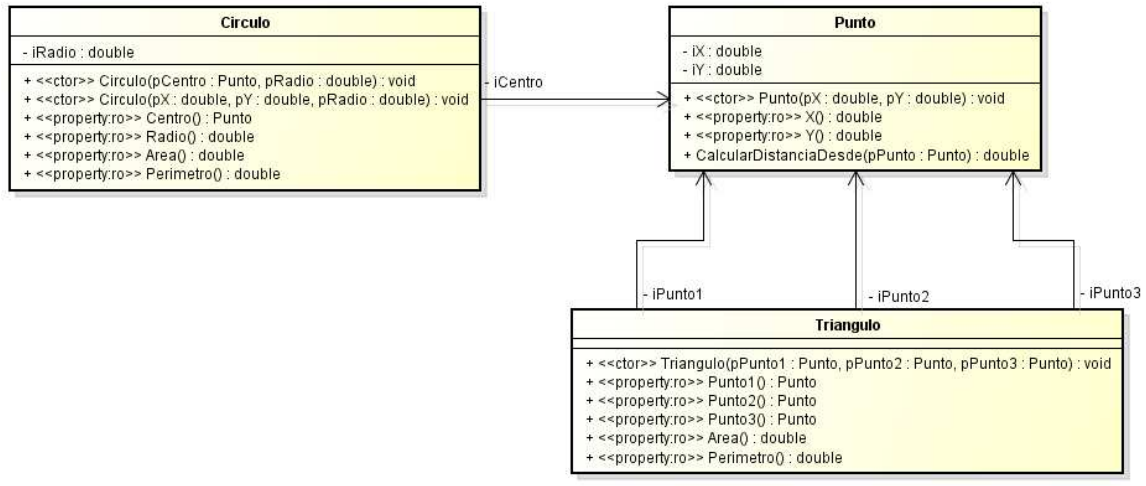
Todos los ejercicios deben estar debidamente documentados, utilizando las etiquetas XML que ofrece .NET para comentar tipos y miembros. Además, todos los identificadores tienen que tener nombres significativos.

Ejercicio 1

Se recibe un requerimiento de un cliente el cual quiere una aplicación de consola con la cual pueda obtener de una forma fácil los valores de área y perímetro de figuras geométricas. Las figuras requeridas son punto, círculo y triángulo. El cliente quiere que se presente un menú para seleccionar la figura y que después se pueda ingresar los valores de los ejes X e Y que definen el tamaño de las mismas sin importar la unidad de medida.

También se recibe un diagrama de clases del diseñador de su equipo de desarrollo el cual define objetos de dominio que se deben crear para satisfacer los requerimientos. Además se indica que se debe utilizar el patrón de diseño GRASP *Controlador de Fachada* para desacoplar la lógica del dominio de la interfaz.

Desarrolle una aplicación que satisfaga el requerimiento de basándose en el diseño y a las indicaciones realizadas.

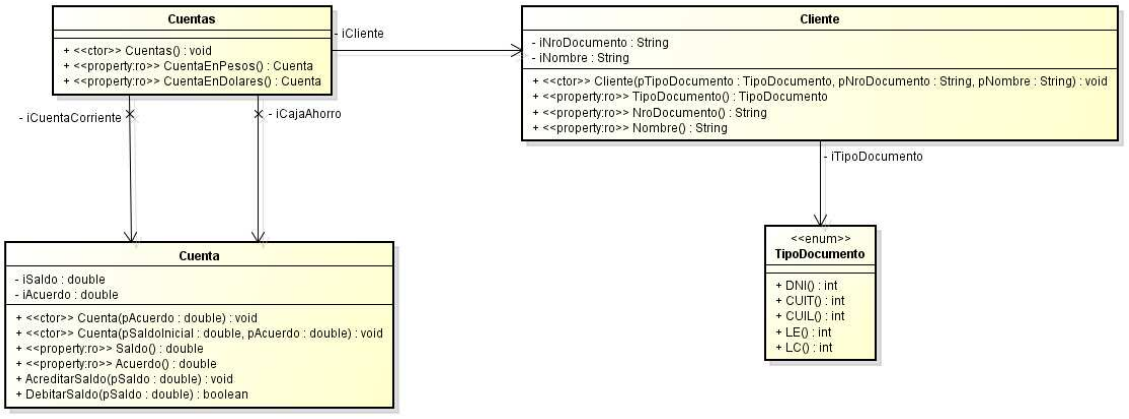


Ejercicio 2

Se recibe un requerimiento para que se desarrolle una aplicación de consola que permita mostrar información y realizar operaciones sobre las cuentas de un cliente en forma

interactiva por consola.

El cliente solamente puede operar con dos cuentas, una cuenta corriente y una caja de ahorro, y seleccionando una de ellas se pueda generar movimientos de débito y crédito con la posibilidad de mostrar el saldo cuando se requiera así como también de realizar transferencias entre ambas cuentas. En el caso de que al momento de debitar saldo de una cuenta este no sea suficiente y el acuerdo de descubierto sea superado, entonces se debe informar al usuario de esta situación.



Por parte del diseñador se recibe el diagrama de clases del dominio y las siguientes indicaciones:

- *TipoDocumento* es un enumerado con los diferentes tipos de documentos existentes.
- La clase *Cliente* representa el cliente al que pertenecen las cuentas.
- La clase *Cuenta* representa una cuenta bancaria, la cual posee un saldo en pesos de la cuenta, el acuerdo de descubierto también en pesos, y responde a los siguientes mensajes:
  - *Saldo:* obtiene el saldo actual de la cuenta.
  - *Acuerdo:* obtiene el acuerdo de descubierto de la cuenta.
  - *AcreditarSaldo:* suma a la cuenta el saldo indicado.
  - *DebitarSaldo:* debita el saldo indicado de la cuenta, siempre y cuando se cuente con saldo suficiente o bien el acuerdo de descubierto cubra el monto a debitar, en cuyo caso devuelve *true*. Si el saldo de la cuenta no es suficiente y el acuerdo de descubierto es superado, entonces el saldo no es debitado y devuelve *false*.
- La clase *Cuenta* posee dos constructores, donde uno de ellos solo permite establecer el acuerdo de descubierto inicializando el saldo inicial de la cuenta en cero, y el otro constructor permite establecer tanto el acuerdo de descubierto como el saldo inicial.
- La clase *Cuentas* mantiene las dos cuentas (caja de ahorro y cuenta corriente) que posee un cliente.

Utilizar el patrón de diseño GRASP *Controlador de Fachada* para desacoplar la lógica de negocio de la interfaz de usuario.

Ejercicio 3

Diseñe y desarrolle una aplicación que permita jugar al juego del "ahorcado". La misma deberá satisfacer al menos los siguientes requerimientos:

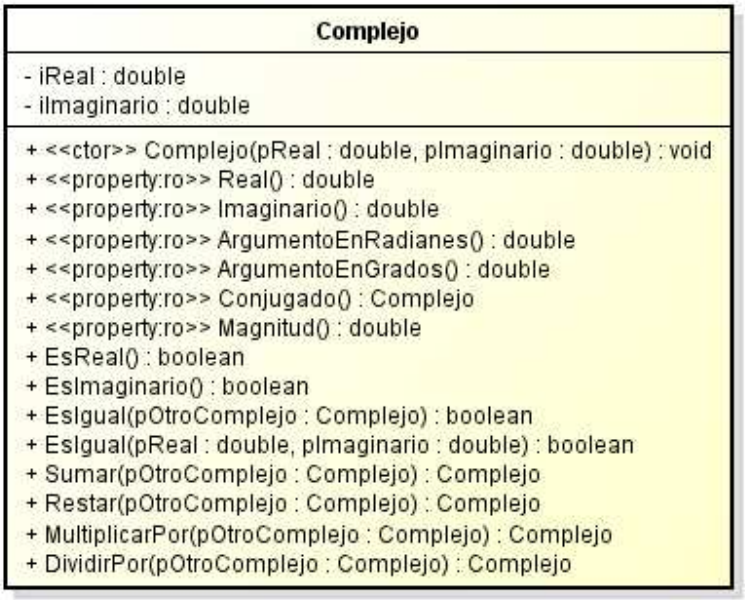
- La palabra a adivinar en una partida debe seleccionarse aleatoriamente de un arreglo de 30 palabras.
- Por defecto la cantidad máxima de fallos para no perder el juego es 10, pero se debe permitir modificar dicho valor.
- Cada partida debe permitir registrar los datos del jugador, la fecha y hora de inicio y fin de la misma, la duración y si el jugador ganó o no la partida.
- Se deben listar las cinco partidas ganadas que tengan el menor tiempo de duración.

Utilice el patrón de diseño GRASP *Controlador de Fachada* para desacoplar la lógica del sistema de la interfaz.

Ejercicio 4

Se debe construir una clase que sirva para representar un número complejo y que se puedan realizar operaciones sobre la misma. Se recibe el diagrama de clases mostrado a continuación con las siguientes indicaciones:

- El único constructor debe asignar valores a sus atributos.
- Las propiedades de solo lectura *Real* e *Imaginario* devuelven los valores atributos respectivamente.
- Los métodos *EsReal* e *EsImaginario* devuelven verdadero o falso según si el número es real o imaginario respecto a cada caso.
- Los métodos *EsIgual* compara el valor del número complejo con otro suministrado como parámetro.
- La propiedad de solo lectura *Magnitud* devuelve la magnitud del número complejo.
- Las propiedades de solo lectura *ArgumentoEnRadianes* y *ArgumentoEnGrados* devuelven el valor del argumento en radianes y grados respectivamente.
- El método *Conjugado* devuelve otra instancia de la clase conteniendo el complejo conjugado de esta instancia.
- Los métodos *Sumar* y *Restar* devuelven otra instancia de la clase conteniendo el resultado de la operación respectivamente.
- Análogamente los métodos *MultiplicarPor* y *DividirPor* devuelven otra instancia de la clase conteniendo el resultado de la operación respectivamente.



Las instancias de la clase creada deben ser *inmutables*, por lo que debe investigar dicho concepto y la forma de implementarlo correctamente utilizando C#.

Ejercicio 5

Se recibe una indicación para construir una clase para representar fechas, que como mínimo responda a los siguientes mensajes:

- Agregar días.
- Agregar meses.
- Agregar años.
- Devolver el componente día de la fecha.
- Devolver el componente mes de la fecha.
- Devolver el componente año de la fecha.
- Devolver el nombre del día de la semana (Lunes, Martes, etc.) de la fecha.
- Indicar si el componente año de la fecha es bisiesto.
- Comparar dos fechas.

Las instancias de la clase creada deben ser *inmutables*.