Seminario de Lenguajes (.NET)

Práctica 6

1) Derive de la clase Persona definida en la práctica anterior la clase Alumno, agréguele un campo privado llamado promedio. Haga accesible en la clase Alumno los campos definidos en Persona. Defina los constructores necesarios. Modifique el programa del ejercicio 6 de la práctica anterior para tratar ahora con alumnos. El usuario tipeará las entradas como:

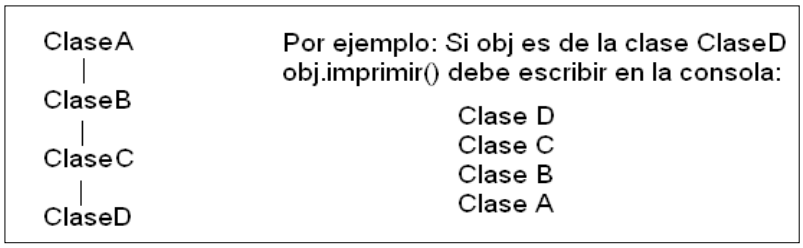
"Nombre<TAB>Documento<TAB>Edad<TAB>Promedio<ENTER>"
o bien:
"Nombre<TAB>Documento<TAB>fecha de nacimiento<TAB>Promedio <ENTER>".

Redefina el método imprimir() heredado de Persona para que se imprima en la consola los datos ALUMNO: Nombre (Edad) <TAB> DNI <TAB> promedio
El listado debe tener la siguiente apariencia:

```
1) ALUMNO: Juan Perez (26)    2998745      7.8
2) ALUMNO: Jose García (25)   3065412      8.4
...
```

2) Modifique el programa anterior para trabajar tanto con Personas como con Alumnos. Para la carga de datos, el usuario antepondrá "A<TAB>" si lo que va a ingresar es un alumno, o "P<TAB>" si se trata de una persona. Utilice un arreglo de personas para guardar tanto personas como alumnos.

3) Cree una jerarquía de clases como la que se indica en el esquema y defina en todas ellas el método imprimir() que escribe en la consola la jerarquía desde la claseA hasta aquella a la que pertenece el objeto en forma invertida.



4) Codifique un programa que cree un arreglo de objetos ClaseA que contenga objetos ClaseC, y objetos ClaseD. Recorra el arreglo llamando al método imprimir() sólo para los objetos ClaseC.

5) Conteste las siguientes preguntas:

a) ¿Para qué sirve un constructor?

b) ¿Para qué sirve un destructor?

c) ¿Cuál es la diferencia entre **`Protected`** y **`Private`**?

d) ¿Para qué sirve invocar el método **`collect`** de la clase **`GC`**?

6) ¿Por qué no funciona el siguiente código? Cómo puede solucionarlo.

```
class Auto{

	double velocidad;

	public virtual void acelerar(){
		Console.WriteLine("Velocidad = {0}", velocidad+=10);
	}

}

class Taxi:Auto{

	public override void acelerar()
	{
		Console.WriteLine("Velocidad = {0}", velocidad+=5);
	}

}
```

7) Utilizando las clases definidas en el ejercicio 6 codifique el siguiente programa:

```
class Ejercicio7{

	static void Main(){
		Auto a=new Auto();
		check(a);
		a=new Taxi();
		check(a);
		System.Console.ReadLine();
	}

	static void check(Auto a){
		if (a is Auto){
			Console.WriteLine("Es un auto no es un taxi");
		}else{
			if (a is Taxi) Console.WriteLine("Es un taxi");
		}
	}

}
```

¿Por qué en la segunda invocación del méto check no se imprimió “Es un taxi”? ¿Cómo puede solucionarlo?

8) ¿Qué puede decir acerca de la definición de la clase Taxi? ¿Es necesario definirle un constructor?

```
class Auto{

  private string marca;

  public Auto(string marca){
    this.marca=marca;
  }
}

class Taxi:Auto{
}
```

9) Pruebe el siguiente programa. Preste atención a los constuctores. ¿Porqué no es necesario agregar :base en el constructor de Taxi? Pruebe eliminar el segundo constructor de la clase Auto y observe lo que sucede.

```
using System;
class Program{
  static void Main(){
    Taxi t=new Taxi(3);
    Console.WriteLine(t.marca+ t.pasajeros);
  }
}
class Auto{
  public string marca="Ford";
  public Auto(string marca){
    this.marca=marca;
  }
  public Auto(){  }
}
class Taxi:Auto{
  public int pasajeros;
  public Taxi(int pasajeros){
    this.pasajeros=pasajeros;
  }
}
```