Práctico IV
19/09/2021
Programación Orientada a Objetos

PARTE I

Seguir el paso a paso y explicar cada concepto asociado a éstos.

- Crear una clase `Persona`.
- Agregar los atributos `nombre` y `apellido` con nivel de acceso `public`.
- Pedir por consola `nombre` y `apellido` de una persona.
- Instanciar una `Persona`, e inicializar sus atributos con los valores ingresados.
- Sobreescribir el método `ToString` para que el objeto imprima sus atributos.
- Agregar el atributo `documento` con nivel de acceso `private`.
- Agregar dos personas más de la misma forma, inicializando también el atributo `documento`.

  Justificar las decisiones de diseño.
- Aplicar encapsulamiento a la clase creada. Explique y justifique los cambios.
- Crear un método que permita a una `Persona` presentarse.
- Crear las subclases `Docente` y `Alumno` de `Persona`.

  `Docente` debe poseer el atributo `idDocente`.
  `Alumno` debe poseer el atributo `idAlumno`.

  Aplique encapsulamiento y justifique.
- Sobreescribir el método de presentación para cada subclase indicando la "ocupación" de cada uno al principio.
- **EXTRA:** crear una interface para `Persona`.

PARTE II

A partir de éste punto, los datos pueden ser cargados manualmente en el código:

- Crear la clase `WebAsignatura`, que contenga como atributo una lista de objetos del tipo `Tarea` llamada `ListaTareas`.
- Cada Tarea debe tener un identificador (`idTarea`), un `nombre`, un `contenido` y `respuestas`.
- Crear una clase llamada `Curso`, que tenga como atributo un `docente titular`, una `WebAsignatura` y una `lista de alumnos`.

PARTE III

A partir de lo ya creado:

- Debe existir un método que permita al `Docente` agregar una `Tarea` a la lista de `Tareas` de `WebAsignatura` con las respuestas correctas.
- Debe existir un método que permita al `Alumno` copiar una tarea, e ingresar sus propias respuestas, que sean guardadas en un atributo propio llamado `ListaTareasRealizadas`.
- Debe existir un método que permita al `Docente` listar las tareas realizadas de todos los alumnos del curso.
- Debe existir un método que permita listar todas las personas relacionadas a un curso.

PARTE IV

A partir del siguiente código, determinar cuáles casos fallarían o compilarían, por qué y cuál sería un caso sugerido para que funcionara correctamente.

```
class Persona {
	private string nombre;
	public string apellido;

	public Persona(string xNombre, string xApellido) {
		this.nombre = xNombre;
		this.apellido = xApellido;
	}

	private string getNombre() {
		return this.nombre;
	}

	public string getApellido() {
		return this.apellido;
	}

	public void setNombre (string xNombre) {
		this.apellido = xNombre;
	}
}

class Docente {
	private int idDocente;
	public Docente (
		string xNombre,
		string xApellido,
		int xIdDocente
	) : base(xNombre, xApellido) {
		this.xIdDocente = idDocente;
	}
```

```
      public string getIdDocente() {
            return this.idDocente;
      }
}

CASO 1 // Persona objPersona = new Persona(“Baruch”, “Spinoza”);
CASO 2 // objPersona.getNombre();
CASO 3 // objPersona.getIdDocente();
CASO 4 // Docente objDocente = new Persona(“Georg”, “Hegel”);
CASO 5 // Persona objDocente2 = new Docente(“Immanuel”, “Kant”,
1);
CASO 6 // objDocente2.getApellido();
CASO 7 // objDocente2.getIdDocente();
CASO 8 // objDocente2.nombre = “Emanuel”;
CASO 9 // objDocente2.apellido = “Kant”;
CASO 10 // objPersona.setNombre(“Baruj”);
CASO 11 // objPersona.setApellido(“Espinosa”);
```