1 Seminarios de C# *(Primera Parte)*

Los requerimientos de cada ejercicio del seminario serán expuestos desde el punto de vista práctico y teórico; es decir, para su exposición, cada equipo se basará en el caso práctico en cuestión para introducir y explicar el elemento teórico requerido. La exposición no es una mera enunciación de código. Preguntas como: *¿Por qué?*, *¿Basándose en qué?*, *¿Cómo se logra esto en el lenguaje* `X`*?* entre otras, deben hacerse.

Todos los miembros del equipo deben participar en la solución del ejercicio y estar preparados para exponer todo el trabajo. **La persona a exponer** se decide el día de la exposición. Quién no esté presente en la exposición de su equipo tiene `0` en la evaluación. (Note que estas notas se promedian y hay distinción entre 0 y 2).

1.1 Seminario 3

1.1.1 Varianza, Covarianza, Herencia y Polimorfismo

1. En la Universidad, una persona (que se identifica por su Nombre) puede representar diferentes roles:

- Estudiante (Acción: `RecibirClase()`)
- Profesor (Acción: `ImpartirClase()`)
- Alumno Ayudante (Estudiante que no es profesor pero actúa como tal en un momento dado, es decir, puede realizar `ImpartirClase()`)
- Trabajador (no todo trabajador es profesor, pero sí todos los profesores son trabajadores. Acción: `CobrarSalario()`)

   a. Diseñe una jerarquía en `C#` que represente/modele los roles anteriores y sus relaciones. Utilice alguna herramienta para ilustrar dicho modelo (Ejemplo: diseñador de clases de *Visual Studio*)
   b. ¿Es posible utilizar el siguiente código para imprimir una lista genérica de profesores? Haga los arreglos que crea necesario para que ejecute en caso de que su respuesta sea **NO**. Explique el funcionamiento de las características del lenguaje utilizadas.

```
void PrintPeople(IEnumerable<Person> people) {
   for(var p in people)
        Console.WriteLine(p.Name);
}
```

c. En la secretaría de la Facultad, usualmente se imprimen listados de estudiantes dado algún criterio (por nombre, por nota, etc). El algoritmo es el siguiente:

```
static void PrintStudents(IEnumerable<Student> students,
IComparer<Student> comparer) {
    foreach (var student in students.OrderBy(x => x, comparer))
         Console.WriteLine(student.Name);
}
```

Implemente un comparador que permita utilizar el código anterior para imprimir los estudiantes ordenados por nombre, pero que dicho comparador se pueda reutilizar luego para profesores, trabajadores y alumnos ayudantes. Explique las características del lenguaje utilizadas.

d. Explique e llustre el funcionamiento del siguiente código:

```
static void PrintByConsole(Action<Action<Person>> person) {
            person(x => Console.WriteLine(x.Name));
}
...
PrintByConsole(x=>new Student() {Name = "Pedro"});
```

e. El siguiente código recibe una colección de personas que pueden ejercer cualquier rol pero se quieren imprimir sólo los que son estudiantes. Complete el espacio para cumplir dicho objetivo.

```
static void PrintStudentsOnly(IEnumerable<object> people) {
    foreach (var student in people._____________________________________)
        Console.WriteLine(student.Name);
}
```

Objetivos: covarianza, contravarianza, herencia y polimorfismo, composición y encapsulamiento (wrapper)