DEPARTAMENTO
INGENIERÍA INFORMÁTICA
Y CIENCIAS DE LA COMPUTACIÓN
FACULTAD DE INGENIERÍA UNIVERSIDAD DE CONCEPCIÓN

Estructuras de Datos (2022-1)

Laboratorio 3

Profesor: Alexander Irribarra

Ayudantes: Leonardo Aravena, Diego Gatica, Vicente Lermanda

Objetivos

Los objetivos del laboratorio son:

- Familiarizarse con los lenguajes de programación.
- Repasar y aplicar conceptos básicos de programación orientada a objeto.

Ejercicios

1. Implementar una clase *Persona* que cumpla los siguientes requisitos[1]:
   (a) Almacene su nombre completo en una variable del tipo *string*.
   (b) Almacene su día, més y año de nacimiento en variables de tipo entero.
   (c) Tener los siguientes métodos:
   - `string getName()`: Retorna el nombre.
   - `void setName(const string &newName)`: Modifica el nombre almacenado.
   - `void setBirthday(int day, int month, int year)`: Modifica la fecha de nacimiento.
   - `int getBirthDay()`: Retorna el día de nacimiento.
   - `int getBirthMonth()`: Retorna el mes de nacimiento.
   - `int getBirthYear()`: Retorna el año de nacimiento.
2. Implementar una clase *Estudiante* que cumpla los siguientes requisitos:
   - Debe heredar de la clase *Persona* del ejercicio 1.
   - Debe almacenar el nombre de su carrera en una variable del tipo *string*.

[1] La sintaxis para las clases y métodos variará en los lenguajes que son distintos a C++.

- Debe almacenar su año de ingreso a la carrera en una variable de tipo entero.
- Debe implementar los métodos *getters* y *setters* de sus variables, similarmente a como se hizo para la clase *Persona* en el ejercicio 1c.

3. Implementar una clase *Profesor* que cumpla los siguientes requisitos:
   - Debe heredar de la clase *Persona* del ejercicio 1.
   - Debe almacenar el nombre de su facultad en una variable del tipo *string*.
   - Debe almacenar su año de ingreso a la carrera en una variable de tipo entero.
   - Debe implementar los métodos *getters* y *setters* de sus variables, similarmente a como se hizo para la clase *Persona* en el ejercicio 1c.

4. Recordemos que el polimorfismo es una de las propiedades de la orientación a objeto que nos permite tratar a objetos de distintas subclases de una misma clase de una misma manera. Por ejemplo, podemos tratar a objetos de las clases *Estudiante* y *Profesor* a través de una interfaz común, que es la clase *Persona*. Para esto utilizamos punteros, como en el siguiente ejemplo:[2]

```
Estudiante *e = new Estudiante();
e->setName("Juanito Perez");
e->setBirthday(24,2,2002);
e->setMajor("Ingenieria Informatica");
Persona *p = (Persona*)e;
cout << p->getName() << endl;
// No podemos hacer p->getMajor(), pues la clase Persona no tiene
declarado este método
```

   Notar que similarmente, un objeto de la clase *Profesor* también puede ser apuntado por un puntero de tipo *Persona*, de manera que podemos acceder a todas las funcionalidades de la clase *Persona* por medio de este puntero.
   Responder brevemente a la siguiente pregunta: ¿Qué ventaja nos entrega el polimorfismo al momento de programar? Para responder esta pregunta, supongamos que queremos implementar una función que reciba a un *Estudiante* o a un *Profesor* y nos retorne su edad. ¿Como nos beneficia el hecho de que estas dos clases sean subclases de *Persona*?, ¿Qué sucedería si es que no utilizáramos la herencia?, ¿Tendríamos que implementar la misma función dos veces?.

[2]Este ejemplo está en C++, sin embargo en los otros lenguajes de programación también se puede aplicar de manera similar, aunque puede que sin necesidad de usar punteros.

5. Complementando a la pregunta anterior. Supongamos que completamos nuestro diseño con clases nuevas que definan al resto de personas que conforman a la universidad, como por ejemplo *Secretario*, *Decano*, *Auxiliar*, etc. Si utilizamos polimorfismo, ¿Tendremos que nuevamente implementar la función que calcula la edad, o la que ya está escrita nos basta?.

Observación

Los estudiantes pertenecientes al minor son libres de implementar las soluciones en el lenguaje de programación *C++*, *Java* o *Python*, en este caso pueden tomar el código proporcionado como una base para empezar a realizar los ejercicios.

Normas de entrega

Antes del siguiente laboratorio, se deben enviar todos los ejercicios resueltos a los ayudantes mediante la plataforma CANVAS.

Se debe entregar un archivo comprimido que contenga:

- Archivo PDF con el nombre completo, número de matrícula, las respuestas a las preguntas que correspondan y capturas de pantalla de la ejecución de sus códigos.
- Todos los ficheros del código fuente.
- **IMPORTANTE:** Los archivos debe llamarse *apellido1_nombre_03.formato*