Actividad ejercicio integrador

**Sección teórica. 30 %**

1. La firma del método *setter* para el atributo tipo *string matriculaAlumno* de la clase ***Alumno*** es:
   1. string setMatricula();
      1. void setMatricula(string);
      2. void setMatricula();
      3. string setMatricula(string);
2. La firma del constructor por omisión de la clase ***Examen*** sería:
   * 1. void Examen();
     2. void Examen(){};
     3. Examen();
     4. Examen Examen();
3. En el siguiente segmento de código en C++, completa lo que se pide atendiendo la sintaxis correcta del lenguaje.

class **MiClase** {

int var1;

public:

MiClase();

**void setVar1 (int Var1);**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ //Coloca aquí la declaración o firma del método setVar1

**int getVar1 () ;**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ //Coloca aquí la declaración o firma del método getVar1

…….

};

Diseña el contenido del constructor por omisión y del método *setVar1*. Inicializa a *var1* en 101 en el constructor;

//Coloca aquí el código que implementa al constructor por omisión.

**MiClase::MiClase() {**

**Var1=101**

**}**

//Coloca aquí el código que implementa al método *setVar1*

***Void MiClase::setVar1 ( int Var1) {***

***this->Var1=Var1***

**}**

1. Siguiendo con el ejemplo anterior, completa la siguiente aplicación que utiliza MiClase.

#include “Miclase.h”

int main(){

// Declara el objeto ***miObjeto*** de tipo **MiClase**

**MiClase miObjeto;**

// Cambia por 1500 el valor de *var1* de ***miObjeto***

***miObjeto.setVar1(1500)***

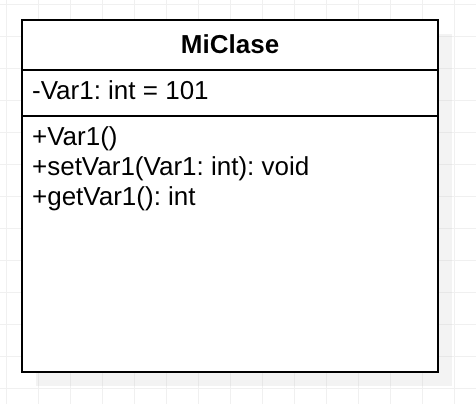
// Muestra en pantalla el contenido de *var1* de ***miObjeto*** return 0;

**Std::cout<<”El contenido de Var1 es: “ << miObjetio.getV1()<<std::endl;**

**Return 0;**

}

1. Usando el estándar revisado en clase, diseña el diagrama de clases de **MiClase**.



Miclase()



1

Actividad ejercicio integrador

**Sección práctica. 70 %**

1. **Diseña** e **implementa** en **C++** la **clase** modelada en el diagrama de la figura 1. Revisa en elapartado Especificaciones algunos requerimientos básicos que deberás considerar.
2. **Diseña** los **casos** de **prueba** que te permitan probar casos generales de tu clase.
3. **Coloca** en el **diagrama** (figura 1) el tipo de relación adecuada entre las clases si en la claseEmpleado se utiliza un atributo de clase Fecha.
4. En función de los casos de prueba, **agrega** una **aplicación** en consola que permita implementar los casos de prueba.

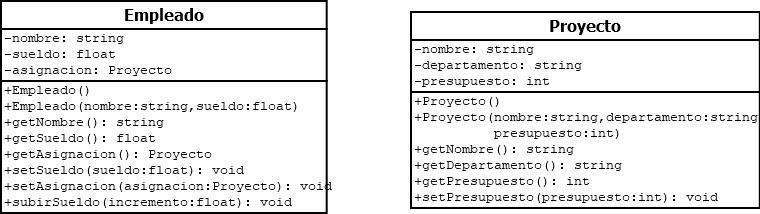




Figura 1. Diagrama de clases. Clase Empleado y clase Proyecto.

Especificaciones.

1. Incluye los **constructores** señalados para cada clase.
2. Incluye los métodos ***setters*** y ***getters*** requeridos para ambas clases.
3. ***subirSueldo***. El método recibe como parámetro un porcentaje de incremento entre el 1 y el 10%Si el valor recibido no está dentro de este rango, no se aplicará incremento alguno y se mostrará el mensaje en pantalla “*Porcentaje inválido, no hay incremento salarial*”. Si el porcentaje es correcto, al sueldo del empleado deberá aplicarse el incremento.
4. En tu aplicación,
   1. Declara tres empleados, *emp1, emp2 y emp3*. El empleado emp1 se llama “Pepe”, gana 10000. El empleado emp2 se llama “Rocio”, gana 12000. El empleado emp3 se llama

“Luis”, gana 13500.

* 1. Declara dos proyectos, proy1 y proy2. El proyecto proy1 se llama “ProjectX”, es controlado por el departamento de “Research” y tiene un presupuesto de 1000000. El proyecto proy2 se llama “PPG”, es controlado por el departamento “Marketing” y tiene un presupuesto de 50000.
  2. Asigna el proyecto proy1 a Pepe y Rocio y el proy2 a Luis.
  3. Muestra en pantalla los datos completos de los dos empleados. Acomoda los datos de cada empleado en un mismo renglón.
  4. Intenta subirle el salario a Pepe un 15%.
  5. Muestra en pantalla nombre y sueldo de Pepe.
  6. Muestra en pantalla el proyecto al que está asignada Rocío.
  7. Intenta subirle el salario a Luis un 5%.
  8. Muestra en pantalla el nombre y sueldo de Luis.