Actividad 3 - Ejecución.

Lenguajes de Programación II.

Ingeniería en Desarrollo de Software

Tutor: Miguel Ángel Rodríguez Vega

Alumno: Ramón Ernesto Valdez Felix

Fecha:24/11/2023

Índice

Introducción

En la actividad 3 de la materia de lenguajes de programación II, se necesita realizar un programa en la herramienta de programación de visual estudio con la compatibilidad de c++ donde requerimos la creación de un script en donde utilizaremos clases, getter, setter y destructores para la empresa de nombre UNI donde pueda controlar algunos datos de sus empleados y alumnos con sus datos personales y se correlaciones con las subclases a utilizar para obtener la información deseada por la empresa UNI usando como ejemplo las siguientes: (Persona, alumno y maestro) estas deben de utilizar atributos que es correlaciones entre si mostrando la información completa al ser solicitada en el código script de C++, adiciona se la evidencia de la creación del programa y sus resultado para ser entregados en la actividad final de la materia en curso.

Descripción

En esta actividad final entregaremos el documento realizado de nombre “Ejecución” esto nos dará el derecho a ser calificada para así obtenerte la puntuación de la calificación final de la materia impartida por el docente o maestro asignado a la materia de leguajes de la programación II, ya que es necesario realizar la documentación para la actividad 3 donde se nos pide que trabajemos con la creación de un script de c++ en la herramienta de programación de visual estudio utilizando una clase padre, subclases utilizando para obtener el requerimiento solicitado por la empresa UNI para así tener el control de sus empleados, alumnos donde esto proporcione la información correcta al realizar ejecución del script y así poder tomar las capturas de evidencia que se anexar en el documento a entregar.

Justificación

En esta actividad se trabajará con la herramienta de visual estudio con la creación de un script programado C++ donde utilice una clase maestras, tres o más subclases utilizando getter, setter y destructores en la programación con la finalidad que se hereden los atributos principales en las subclases a utilizar en la documentación a presentar al docente de la materia de lenguajes de la programación:

- Usar una clase padres con herencia en los atributos a utilizar en las subclases.
- Subirlo al GitHub el documento realizado compartiendo el link para que pueda consultar el docente o maestro.
- Descargar e instala visual estudio para la creación del script de C++ a utilizar.
- Script de C++ como evidencia de la programación realizada para la actividad.
- Pantallas de evidencia del código y resultados de ejecución del programa de C++.
- Utilizar los métodos getter, setter y destructor para la creación del script de C++.

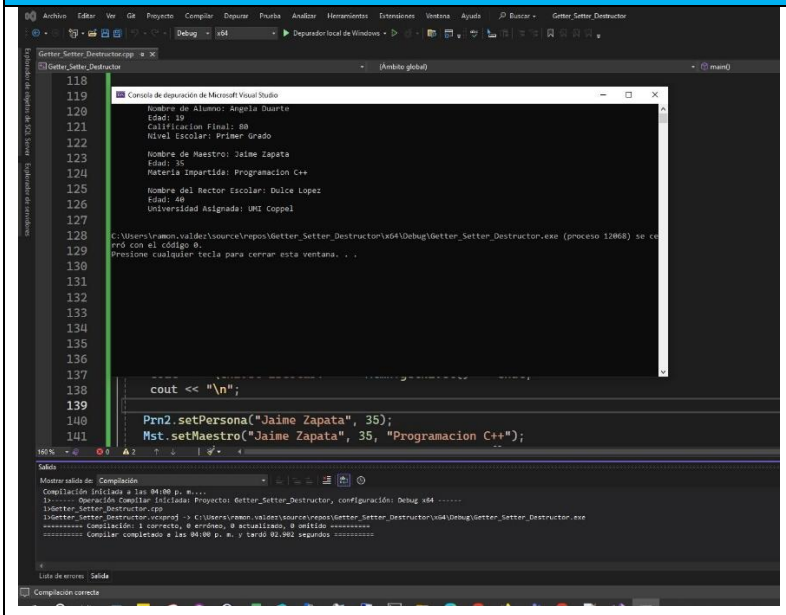
Desarrollo:

En esta actividad realizaremos las actividades siguientes: un programa en el lenguaje de programación de C++ utilizando la herramienta de visual estudio donde plasmaremos el código o script solicitado por la actividad en curso de la clase de lenguaje de programación II donde utilizaremos los métodos de getter, setter y destructores.

Link: GitHub

Ejecución.

En esta actividad realizaremos la ejecución del script de C++ creado en la herramienta de visual estudio donde en el cual se utilizaron los métodos de getter, setter y destructores para la creación.

Ejecución de código de C++ utilizando en el programa realizado en la herramienta de visual estudio anexando la imagen de evidencia a continuación.	
	Se realiza la ejecución mostrando las 3 clases distintas Alumno, Maestro y Rector, agregando los atributos que corresponde a cada una de las clases mostradas en pantalla al ejecutar el script de C++ con los Métodos Getter, Setter y Destructores.

Código.

En esta actividad tomaremos todas las evidencias que sean requeridas para la explicación del funcionamiento correcto del programa realizado en C++, el código del programa servirá como evidencia principal, las imágenes de ejecución evidencian de funcionalidad el código generado para la entrega de la actividad solicitada:

Código C++ utilizando como base una clase padre y 3 subclases con la herencia de os atributos a través del uso del polimorfismo en el script realzado.

```cpp
#include <iostream>
#include <string>
using namespace std;

class Persona {
private:
    string Nombre;
    int Edad;

public:
    Persona();
    virtual void setPersona(string, int);
    string getNombre();
    int getEdad();
    virtual ~Persona();
};

class Alumno : public Persona {
private:
    float CalFinal;
    string Nivel;

public:
    Alumno();
    void setAlumno(string, int, float, string);
    float getCalFinal();
    string getNivel();
    ~Alumno();
};

class Maestro : public Persona {
private:
    string Materia;

public:
    Maestro();
    void setMaestro(string, int, string);
    string getMateria();
    ~Maestro();
};
```

En el script del programa realizado en el lenguaje de programación C++:

- Clase Padre Persona: (Nombre y Edad)
    - Subclase Alumnos.
        - Calificación Final.
        - Nivel de estudio.
    - Subclase Maestro.
        - Materia Impartida.
    - Subclase Rector.
        - Atributo: Universidad asignada.

Donde la información principal es heredad desde la clase padre a las subclases utilizadas los métodos getter, setter en el script programado.

```
class Rector : public Persona {
private:
    string Uni;

public:
    Rector();
    void setRector(string, int, string);
    string getUni();
    ~Rector();
};

Persona::Persona() {
}//Constructor Persona

void Persona::setPersona(string _Nom, int _Edad) {
    Nombre = _Nom;
    Edad = _Edad;
}

string Persona::getNombre() {
    return Nombre;
}

int Persona::getEdad() {
    return Edad;
}

Persona::~Persona() {}//Destructor

Alumno::Alumno() {
}//Constructor Alumno
```

```
void Alumno::setAlumno(string _Nom, int _Edad, float _CalFin, string _Nvl) {
    CalFinal = _CalFin;
    Nivel = _Nvl;
}

float Alumno::getCalFinal() {
    return CalFinal;
}

string Alumno::getNivel() {
    return Nivel;
}

Alumno::~Alumno() {}//Destructor

Maestro::Maestro() {
}//Constructor Maestro

void Maestro::setMaestro(string _Nom, int _Edad, string _Materia) {
    Materia = _Materia;
}

string Maestro::getMateria() {
    return Materia;
}

Maestro::~Maestro() {}//Destructor

Rector::Rector() {
}//Constructor Maestro

void Rector::setRector(string _Nom, int _Edad, string _Uni) {
    Uni = _Uni;
}
```

```
Rector::~Rector() {}//Destructor

int main() {

    Persona Prn0, Prn1, Prn2, Prn3;
    Alumno Almn;
    Maestro Mst;
    Rector Rtr;

   //Prn0.setPersona("Prueba Nombre", 1);
    //cout << "\tNombre: " << Prn0.getNombre() << endl;
    //cout << "\tEdad: " << Prn0.getEdad() << endl;
    //cout << "\n";


   Prn1.setPersona("Angela Duarte", 19);
   Almn.setAlumno("Angela Duarte", 19, 80, "Primer Grado");
    cout << "\tNombre de Alumno: " << Prn1.getNombre() << endl;
    cout << "\tEdad: " << Prn1.getEdad() << endl;
    cout << "\tCalificacion Final: " << Almn.getCalFinal() << endl;
    cout << "\tNivel Escolar: " << Almn.getNivel() << endl;
    cout << "\n";

   Prn2.setPersona("Jaime Zapata", 35);
   Mst.setMaestro("Jaime Zapata", 35, "Programacion C++");
    cout << "\tNombre de Maestro: " << Prn2.getNombre() << endl;
    cout << "\tEdad: " << Prn2.getEdad() << endl;
    cout << "\tMateria Impartida: " << Mst.getMateria() << endl;
    cout << "\n";

   Prn3.setPersona("Dulce Lopez", 40);
   Rtr.setRector("Dulce Lopez", 40, "UMI Coppel");
    cout << "\tNombre del Rector Escolar: " << Prn3.getNombre() << endl;
    cout << "\tEdad: " << Prn3.getEdad() << endl;
    cout << "\tUniversidad Asignada: " << Rtr.getUni() << endl;
    cout << "\n";


   return 0;
}
```

Conclusión

En conclusión: Los setters y getters son métodos que permiten acceder y modificar los atributos de una clase de manera controlada y segura. En C++, son fundamentales para garantizar la encapsulación y el abstraccionismo de los objetos, ya que impiden que se acceda directamente a los datos privados de una clase desde fuera de ella. Además, a través de los setters y getters, es posible realizar validaciones y operaciones adicionales antes de permitir el acceso o la modificación de un atributo. Aunque su uso puede resultar un poco más tedioso que el acceso directo a los atributos, los setters y getters son una práctica recomendada en la programación orientada a objetos, ya que mejoran el modularidad, la seguridad y la mantenibilidad del código. En resumen, los setters y getters en C++ son métodos que se utilizan para acceder y modificar los valores de los atributos de una clase de manera controlada. Son una buena práctica de programación, ya que permiten mantener la encapsulación de la información y evitar que los atributos sean modificados de manera inapropiada. Además, los setters y getters son una forma efectiva de proteger los datos privados de una clase, lo que aumenta la seguridad y la calidad del

código. En definitiva, los setters y getters son una herramienta esencial en la programación orientada a objetos en C++

Cuando se define un destructor para una clase, éste es llamado automáticamente cuando se abandona el ámbito en el que fue definido. Esto es así salvo cuando el objeto fue creado dinámicamente con el operador new, ya que, en ese caso, cuando es necesario eliminarlo, hay que hacerlo explícitamente usando el operador delete. En general, será necesario definir un destructor cuando nuestra clase tenga datos miembros de tipo puntero, aunque esto no es una regla estricta.

Referencias

GitHub: Let’s build from here. (n.d.)

¿Qué son los setters y getters en C++? (2023, May 18). *Beagle Spain - Tu Web y Tienda de Beagle*. https://beaglespain.com/que-son-los-setters-y-getters-en-c/

Pozo, S. (2001). *C++ con clase*. https://conclase.net/c/curso/cap30