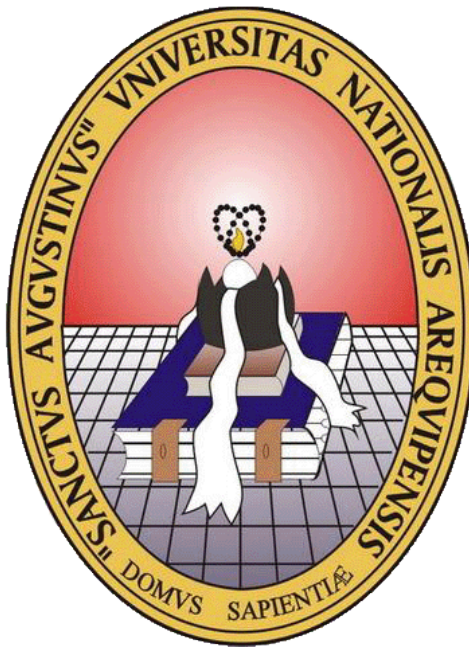


UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA
FACULTAD DE PRODUCCIÓN Y SERVICIOS
ESCUELA DE CIENCIAS DE LA COMPUTACIÓN



PRÁCTICA DE LABORATORIO 7
CURSO DE CIENCIAS DE LA COMPUTACIÓN II

ESTUDIANTE:
RUIZ MAMANI, EDUARDO GERMÁN

EMAIL: eruizm@unsa.edu.pe

CUI: 20193061

TURNO:

C

AREQUIPA- PERÚ

2021

LINK DEL REPOSITORIO: https://github.com/EGRM23/CCII_20193061.git

1. EJERCICIO

1. Crear una clase Persona del cual tendrá métodos asignar una edad y nombre. Una segunda clase, alumno, tendrá que heredar este contenido y a través de esta clase poder asignar los datos de edad y nombre de los estudiantes.

- **CÓDIGO**

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

//EDUARDO GERMAN RUIZ MAMANI
// 20193061

class Persona {
protected:
    int edad;
    string nombre;
public:
    Persona(int e, string n) {
        edad = e;
        nombre = n;
    }
    ~Persona() {};
    void setedad(int e) {
        edad = e;
    }
    void setnombre(string n) {
        nombre = n;
    }
    void imprimir() {
        cout << "Nombre: " << nombre << endl;
        cout << "Edad: " << edad << endl;
    }
};

class Alumno : public Persona {
public:
    Alumno(int e, string n) : Persona(e,n) {};
    ~Alumno() {};
};

int main(int argc, char *argv[]) {
    Alumno a(12,"Jose");
    a.imprimir();
    a.setedad(15);
    a.setnombre("Chayanne");
    a.imprimir();
    return 0;
}
```

- **CAPTURAS**

```
C:\Program Files (x86)\Zinjal\bin\runner.exe
Nombre: Jose
Edad: 12
Nombre: Chayanne
Edad: 15

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>
```

2. EJERCICIO

2. Crear una clase Color que mantenga 3 valores (RGB). Una segunda clase Material, tendrá como información una variable de texto que describa algún material (Ejemplo: madera, vidrio, plastico, etc.) Una tercera clase, Objetos, deberá de heredar contenido de ambas clases con la finalidad de describir diferentes objetos en cuanto a color y el material. (Ejemplo: mesa de color café y material de plástico)

- **CÓDIGO**

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

//EDUARDO GERMAN RUIZ MAMANI
// 20193061

class Color {
protected:
    int R;
    int G;
    int B;
public:
    Color() {
        R = 0;
        G = 0;
        B = 0;
    }
    Color(int r, int g, int b) {
        R = r;
        G = g;
        B = b;
    }
    ~Color() {}
    void mostrarColor() {
        cout << "RGB(" << R << "," << G << "," << B << ")";
    }
};

class Material {
protected:
    string mtrl;
public:
    Material(string m) {
        mtrl = m;
    }
}
```

```

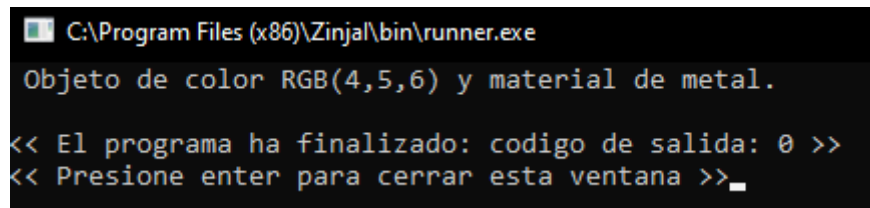
        ~Material() {}
};

class Objeto : public Color, public Material {
public:
    Objeto(int r, int g, int b, string m) : Color(r,g,b),
    Material(m) {}
    ~Objeto() {}
    void descripcion() {
        cout << " Objeto de color ";
        mostrarColor();
        cout << " y material de " << mtrl << ".";
    }
};

int main(int argc, char *argv[]) {
    Objeto o1(4,5,6,"metal");
    o1.descripcion();
    return 0;
}

```

- **CAPTURAS**



```

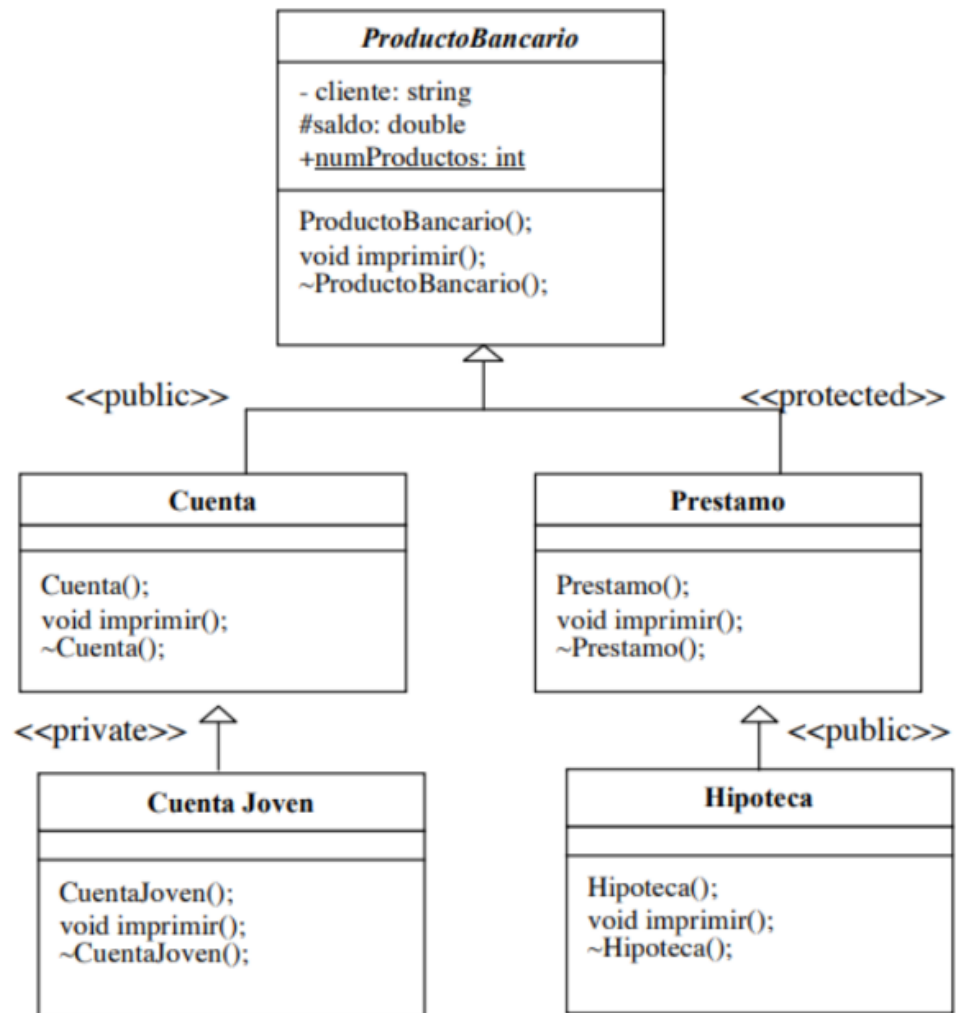
C:\Program Files (x86)\Zinjal\bin\runner.exe
Objeto de color RGB(4,5,6) y material de metal.

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>_

```

3. EJERCICIO

3. Dada la siguiente jerarquía de herencia, indica la visibilidad de los atributos de la clase ProductoBancario en las clases CuentaJoven e Hipoteca.



• **CÓDIGO**

```

• main.cpp
#include<iostream>
#include "ProductoBancario.h"
#include "Cuenta.h"
#include "Prestamo.h"
#include "CuentaJoven.h"
#include "Hipoteca.h"

using namespace std;

int main (int argc, char *argv[]) {
    ProductoBancario c1("Alejandro", 28.12, 20);
    c1.imprimir();

    Cuenta c2("Edson", 8.12, 20);
    c2.imprimir();

    Prestamo c3("Eduardo", 23.04, 20);
    c3.imprimir();
}
  
```

```

    CuentaJoven c4("Jose", 19.03, 20);
    c4.imprimir();

    Hipoteca c5("Silvia", 30.08, 20);
    c5.imprimir();

    return 0;
}

```

- **ProductoBancario.h**

```

#ifndef PRODUCTOBANCARIO_H
#define PRODUCTOBANCARIO_H

#include <iostream>
using namespace std;

class ProductoBancario {
public:
    ProductoBancario(string,double,int);
    ~ProductoBancario();
    void imprimir();
protected:
    string cliente;
    double saldo;
    int numProductos;
};

#endif

```

- **ProductoBancario.cpp**

```

#include "ProductoBancario.h"

ProductoBancario :: ProductoBancario(string n, double s, int
num) {
    cliente = n;
    saldo = s;
    numProductos = num;
}

ProductoBancario :: ~ProductoBancario() {}

void ProductoBancario :: imprimir() {
    cout << "CLIENTE\n";
    cout << "Nombre: " << cliente << endl;
    cout << "Saldo: " << saldo << endl;
    cout << "Cantidad de Productos: " << numProductos << endl
<< endl;
}

```

- **Cuenta.h**

```

#ifndef CUENTA_H
#define CUENTA_H
#include "ProductoBancario.h"

class Cuenta : public ProductoBancario {

```

```

public:
    Cuenta(string,double,int);
    ~Cuenta();
    void imprimir();
};

#endif

```

- **Cuenta.cpp**

```

#include "Cuenta.h"

Cuenta :: Cuenta(string n, double s, int num) :
ProductoBancario(n,s,num) {}

Cuenta :: ~Cuenta() {}

void Cuenta :: imprimir() {
    cout << "CUENTA DE ";
    ProductoBancario :: imprimir();
}

```

- **Prestamo.h**

```

#ifndef PRESTAMO_H
#define PRESTAMO_H
#include "ProductoBancario.h"

class Prestamo : protected ProductoBancario {
public:
    Prestamo(string,double,int);
    ~Prestamo();
    void imprimir();
};

#endif

```

- **Prestamo.cpp**

```

#include "Prestamo.h"

Prestamo :: Prestamo(string n, double s, int num) :
ProductoBancario(n,s,num) {}

Prestamo :: ~Prestamo () {}

void Prestamo :: imprimir() {
    cout << "PRESTAMO DEL ";
    ProductoBancario :: imprimir();
}

```

- **CuentaJoven.h**

```

#ifndef CUENTAJOVEN_H
#define CUENTAJOVEN_H
#include "Cuenta.h"

class CuentaJoven : private Cuenta {
public:

```

```

        CuentaJoven(string,double,int);
        ~CuentaJoven();
        void imprimir();
    };

#endif

```

- **CuentaJoven.cpp**

```
#include "CuentaJoven.h"
```

```

CuentaJoven :: CuentaJoven(string n, double s, int num) :
Cuenta(n,s,num) {}

```

```
CuentaJoven :: ~CuentaJoven() {}
```

```

void CuentaJoven :: imprimir() {
    cout << "CUENTA JOVEN DE ";
    ProductoBancario :: imprimir();
}

```

- **Hipoteca.h**

```

#ifndef HIPOTECA_H
#define HIPOTECA_H
#include "Prestamo.h"

```

```

class Hipoteca : public Prestamo{
public:
    Hipoteca(string,double,int);
    ~Hipoteca();
    void imprimir();
};

```

```
#endif
```

- **Hipoteca.cpp**

```
#include "Hipoteca.h"
```

```

Hipoteca :: Hipoteca(string n, double s, int num) :
Prestamo(n,s,num) {}

```

```
Hipoteca :: ~Hipoteca() {}
```

```

void Hipoteca :: imprimir() {
    cout << "HIPOTECA EN EL ";
    Prestamo :: imprimir();
}

```

- **CAPTURAS**


```
C:\Program Files (x86)\Zinjal\bin\runner.exe

CLIENTE
Nombre: Alejandro
Saldo: 28.12
Cantidad de Productos: 20

CUENTA DE CLIENTE
Nombre: Edson
Saldo: 8.12
Cantidad de Productos: 20

PRESTAMO DEL CLIENTE
Nombre: Eduardo
Saldo: 23.04
Cantidad de Productos: 20

CUENTA JOVEN DE CLIENTE
Nombre: Jose
Saldo: 19.03
Cantidad de Productos: 20

HIPOTECA EN EL PRESTAMO DEL CLIENTE
Nombre: Silvia
Saldo: 30.08
Cantidad de Productos: 20

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>
```

4. EJERCICIO

4. Escribe una clase de nombre ClaseDisco, que herede de la clase ClaseMultimedia los atributos y métodos definidos por usted. La clase ClaseDisco tiene, aparte de los elementos heredados, un atributo más también definido por usted. Al momento de imprimir la información debe mostrarse por pantalla toda la información.

- **CÓDIGO:**

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

//EDUARDO GERMAN RUIZ MAMANI
// 20193061

class Multimedia {
protected:
    string tipo;
    string codigo;
    int cantidad;
public:
    Multimedia(string t, string c, int cant) {
        tipo = t;
        codigo = c;
    }
};
```

```

        cantidad = cant;
    }

    ~Multimedia() {}

    void mostrarM() {
        cout << "MULTIMEDIA:\n";
        cout << "Tipo: " << tipo << endl;
        cout << "Codigo: " << codigo << endl;
        cout << "Cantidad: " << cantidad << endl;
    }
};

class Disco : public Multimedia{
protected:
    string modelo;
public:
    Disco(string t, string c, int cant, string m) :
    Multimedia(t,c,cant) {
        modelo = m;
    }

    ~Disco() {}

    void mostrarD() {
        cout << "DISCO:\n";
        cout << "Modelo: " << modelo << endl;
        cout << "CARACTERISTICAS ";
        mostrarM();
    }
};

int main(int argc, char *argv[]) {
    Disco d1("Portable","edu4ger56",23,"WW");
    d1.mostrarD();
    return 0;
}

```

- **CAPTURAS**

```

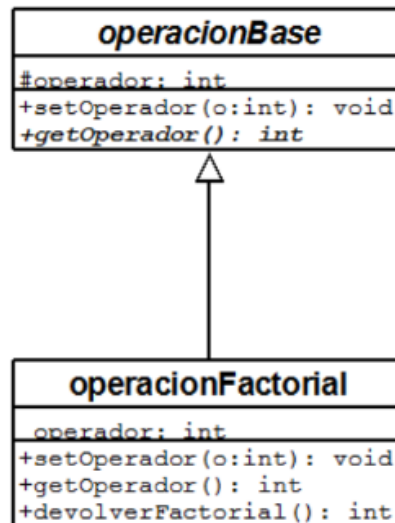
C:\Program Files (x86)\Zinjal\bin\runner.exe
DISCO:
Modelo: WW
CARACTERISTICAS MULTIMEDIA:
Tipo: Portable
Codigo: edu4ger56
Cantidad: 23

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>

```

5. EJERCICIO

5. Escribe un programa que implemente la siguiente jerarquía de clases



- **CÓDIGO**

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

//EDUARDO GERMAN RUIZ MAMANI
// 20193061

class operacionBase {
private:
    int operador;
public:
    void setOperador(int op) {
        operador = op;
    }

    int getOperador() {return operador;}
};

class operacionFactorial : public operacionBase {
private:
    int operador;
public:
    void setOperador(int op) {operacionBase ::
setOperador(op);}

    int getOperador() {return operacionBase :: getOperador();}

    int devolverFactorial() {
        int n = getOperador();

        for (int i = n-1; i > 0; i--)
            n *= i;

        return n;
    }
};
```

```

int main(int argc, char *argv[]) {
    operacionFactorial of1;
    of1.setOperador(5);
    cout << "Operador: " << of1.getOperador() << endl;
    cout << "Factorial: " << of1.devolverFactorial();
    return 0;
}

```

- **CAPTURAS**

```

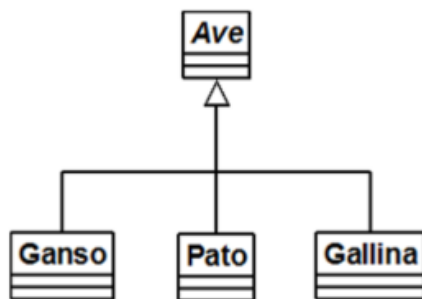
C:\Program Files (x86)\Zinjal\bin\runner.exe
Operador: 5
Factorial: 120

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>_

```

6. EJERCICIO

6. Escribe un programa que implemente la siguiente jerarquía de clases, Debe implementar aquellos atributos y métodos necesarios para que se pueda ejecutar el siguiente programa:



- **CÓDIGO**

- **main.cpp**

```

#include<iostream>
#include "Ave.h"
#include "Ganso.h"
#include "Pato.h"
#include "Gallina.h"
using namespace std;

int main (int argc, char *argv[]) {
    Ganso a1("Armando");
    a1.mostrardat();

    Pato a2("Donald");
    a2.mostrardat();

    Gallina a3("Bessy");
    a3.mostrardat();
    return 0;
}

```

- **Ave.h**

```

#ifndef AVE_H
#define AVE_H
#include <iostream>
using namespace std;

class Ave {
public:
    Ave(string);
    ~Ave();
    bool getvolar() {return volar;}
    string getNombre() {return nombre;}
    string getTipo() {return tipo;}
    void mostrardat();
protected:
    string nombre;
    string tipo;
    int patas = 2;
    bool volar;
};

#endif

```

- **Ave.cpp**

```

#include "Ave.h"

Ave :: Ave(string n) {
    nombre = n;
}

Ave :: ~Ave() {}

void Ave :: mostrardat() {
    cout << "AVE\n";
    cout << "Nombre: " << nombre << endl;
    cout << "Especie: " << tipo << endl;
    cout << "Cantidad de Patas: " << patas << endl;
    cout << "Puede Volar? ";
    if (volar == true)
        cout << "SI\n";
    else
        cout << "NO\n";

    cout << endl;
}

```

- **Ganso.h**

```

#ifndef GANSO_H
#define GANSO_H
#include "Ave.h"

class Ganso : public Ave {
public:
    Ganso(string);
    ~Ganso();
};

```

```
#endif
```

- **Ganso.cpp**

```
#include "Ganso.h"
```

```
Ganso :: Ganso(string n) : Ave(n) {  
    tipo = "Ganso";  
    volar = true;  
}
```

```
Ganso :: ~Ganso() {}
```

- **Pato.h**

```
#ifndef PATO_H  
#define PATO_H  
#include "Ave.h"
```

```
class Pato : public Ave {  
public:  
    Pato(string);  
    ~Pato();  
};
```

```
#endif
```

- **Pato.cpp**

```
#include "Pato.h"
```

```
Pato :: Pato(string n) : Ave(n) {  
    tipo = "Pato";  
    volar = true;  
}
```

```
Pato :: ~Pato() {}
```

- **Gallina.h**

```
#ifndef GALLINA_H  
#define GALLINA_H  
#include "Ave.h"
```

```
class Gallina : public Ave {  
public:  
    Gallina(string);  
    ~Gallina();  
};
```

```
#endif
```

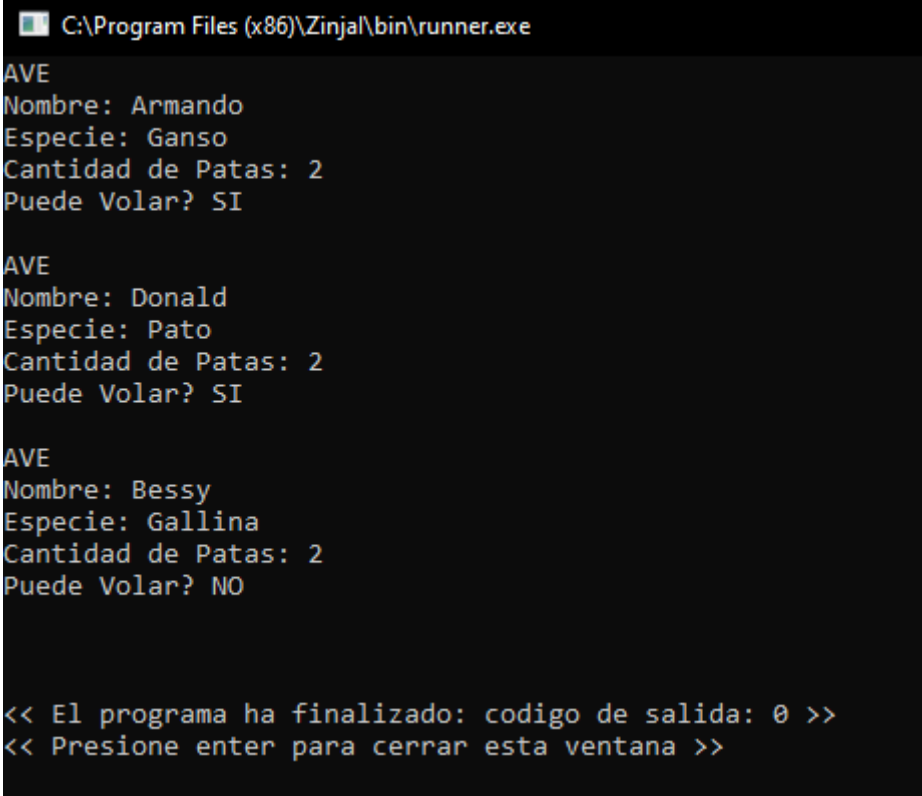
- **Gallina.cpp**

```
#include "Gallina.h"
```

```
Gallina :: Gallina(string n) : Ave(n) {  
    tipo = "Gallina";  
    volar = false;
```

```
}  
  
Gallina :: ~Gallina() {}
```

- **CAPTURAS**



```
C:\Program Files (x86)\Zinjal\bin\runner.exe  
AVE  
Nombre: Armando  
Especie: Ganso  
Cantidad de Patas: 2  
Puede Volar? SI  
  
AVE  
Nombre: Donald  
Especie: Pato  
Cantidad de Patas: 2  
Puede Volar? SI  
  
AVE  
Nombre: Bessy  
Especie: Gallina  
Cantidad de Patas: 2  
Puede Volar? NO  
  
<< El programa ha finalizado: codigo de salida: 0 >>  
<< Presione enter para cerrar esta ventana >>
```