

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



PRÁCTICA DE LABORATORIO 16
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

1. Dado el siguiente modelo de la siguiente imagen, realizar la implementación del modelo. De ser posible, incluir una interfaz para Linux que también sea utilizado por los productos Button y CheckBox. (Las funciones Draw() solo imprimen el tipo de Producto y el sistema en que se encuentra)

- **CÓDIGO**

```
#include <iostream>
#include <string.h>
using namespace std;

class WinFactory;
class MacFactory;
class LinFactory;

class GUIFactory {
public:
    virtual WinFactory* CrearControlW() const = 0;
    virtual MacFactory* CrearControlM() const = 0;
    virtual LinFactory* CrearControlL() const = 0;
};

class WinFactory {
public:
    virtual ~WinFactory() {};
    virtual string Draw() const = 0;
};

class WinButton : public WinFactory {
public:
    string Draw() const override {
        return "Dibujando Button Windows";
    }
};

class WinCheckbox : public WinFactory {
    string Draw() const override {
        return "Dibujando Checkbox Windows";
    }
};

class LinFactory {
public:
    virtual ~LinFactory() {};
    virtual string Draw() const = 0;
};

class LinButton : public LinFactory {
public:
    string Draw() const override {
        return "Dibujando Button Linux";
    }
};
```

```

class LinCheckbox : public LinFactory {
    string Draw() const override {
        return "Dibujando Checkbox Linux";
    }
};

class MacFactory {
public:
    virtual ~MacFactory() {};
    virtual string Draw() const = 0;
    virtual string draw(const WinFactory& colaborador) const =
0;
    virtual string draw(const LinFactory& colaborador) const =
0;
};

class MacButton : public MacFactory {
public:
    string Draw() const override {
        return "Dibujando Button Mac";
    }
    string draw(const WinFactory& colaborador) const override
{
        const string result = colaborador.Draw();
        return this->Draw() + " con ayuda de un " +
result.substr(10,result.size()-1);
    }
    string draw(const LinFactory& colaborador) const override
{
        const string result = colaborador.Draw();
        return this->Draw() + " con ayuda de un " +
result.substr(10,result.size()-1);
    }
};

class MacCheckbox : public MacFactory {
public:
    string Draw() const override {
        return "Dibujando Checkbox Mac";
    }
    string draw(const WinFactory& colaborador) const override
{
        const string result = colaborador.Draw();
        return this->Draw() + " con ayuda de " +
result.substr(10,result.size()-1);
    }
    string draw(const LinFactory& colaborador) const override
{
        const string result = colaborador.Draw();
        return this->Draw() + " con ayuda de " +
result.substr(10,result.size()-1);
    }
};

```

```

class Button : public GUIFactory {
public:
    virtual ~Button() {
    }
    WinFactory* CrearControlW() const override {
        return new WinButton();
    }
    MacFactory* CrearControlM() const override {
        return new MacButton();
    }
    LinFactory* CrearControlL() const override {
        return new LinButton();
    }
};

class Checkbox : public GUIFactory {
public:
    virtual ~Checkbox() {
    }
    WinFactory* CrearControlW() const override {
        return new WinCheckbox();
    }
    MacFactory* CrearControlM() const override {
        return new MacCheckbox();
    }
    LinFactory* CrearControlL() const override {
        return new LinCheckbox();
    }
};

void Application(const GUIFactory& f, int os) {
    switch (os) {
        case 1: {
            const WinFactory* boton_win =
f.CrearControlW();
            cout << boton_win->Draw() << "\n";
            delete boton_win;
            cout << endl;
            break;
        };
        case 2: {
            const MacFactory* boton_mac =
f.CrearControlM();
            cout << boton_mac->Draw() << "\n";
            delete boton_mac;
            cout << endl;
            break;
        };
        case 3: {
            const LinFactory* boton_lin =
f.CrearControlL();
            cout << boton_lin->Draw() << "\n";
            delete boton_lin;
            cout << endl;
            break;
        };
    }
}

```

```

        };
    }

}

int main() {
    cout << "Cliente: Windows\n";
    Button* f1 = new Button();
    Application(*f1,1);
    delete f1;

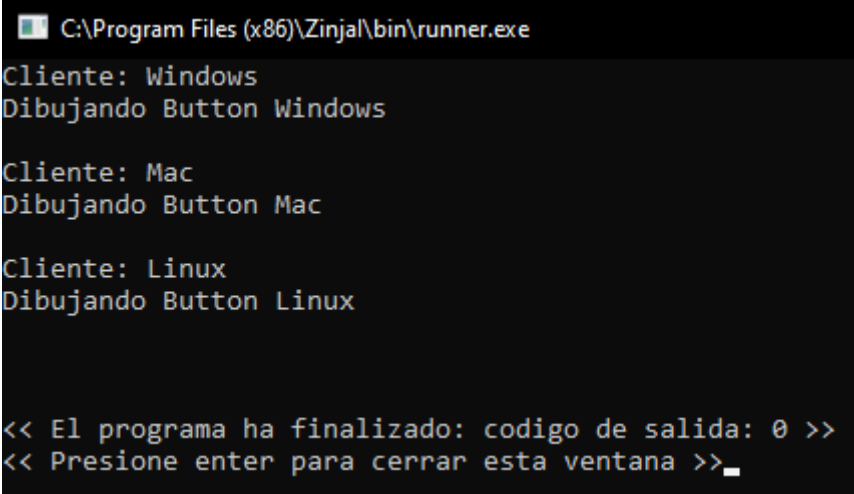
    cout << "Cliente: Mac\n";
    Button* f2 = new Button();
    Application(*f2,2);
    delete f2;

    cout << "Cliente: Linux\n";
    Button* f3 = new Button();
    Application(*f3,3);
    delete f3;

    return 0;
}

```

- **CAPTURAS**



```

C:\Program Files (x86)\Zinjal\bin\runner.exe
Cliente: Windows
Dibujando Button Windows

Cliente: Mac
Dibujando Button Mac

Cliente: Linux
Dibujando Button Linux

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

```