UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

FACULTAD DE INGENIERIA DE PRODUCCION Y SERVICIOS

ESCUELA PROFESIONAL DE CIENCIAS DE LA COMPUTACIÓN

LABORATORIO 16 – Patrones de Diseño –

Abstract Factory.

DOCENTE:

Enzo Edir Velásquez Lobatón

ALUMNO:

Owen Haziel Roque Sosa.

FECHA:

30/07/2022

Arequipa – Perú

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)

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

class AbstractButton;
class AbstractCheckBox;

class AbstractGUIFactory {
public:
    virtual AbstractButton* CrearButton() const = 0;
    virtual AbstractCheckBox* CrearCheckBox() const = 0;
};

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

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

class MacButton : public AbstractButton {
public:
    std::string Draw() const override {
        return "Dibujando Button Mac";
    }
};

class LinuxButton : public AbstractButton {
public:
    std::string Draw() const override {
        return "Dibujando Button Linux";
    }
};
```

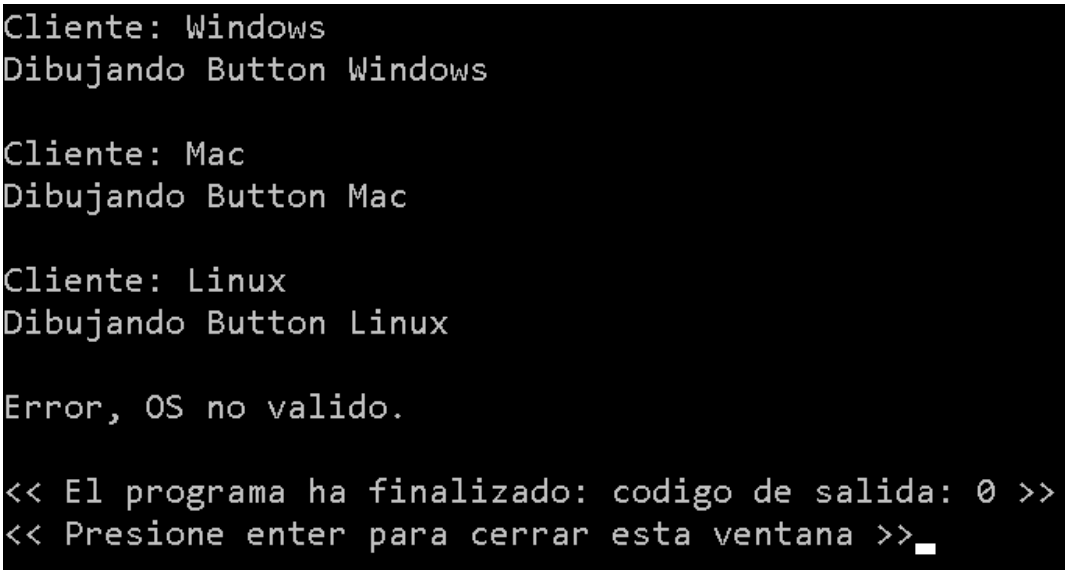
```cpp
class AbstractCheckBox {
public:
    virtual ~AbstractCheckBox() {};
    virtual std::string Draw() const = 0;
};

class WinCheckBox : public AbstractCheckBox {
public:
    std::string Draw() const override {
        return "Dibujando CheckBox Windows.";
    }
};

class MacCheckBox : public AbstractCheckBox {
public:
    std::string Draw() const override {
        return "Dibujando CheckBox Mac.";
    }
};

class LinuxCheckBox : public AbstractCheckBox {
public:
    std::string Draw() const override {
        return "Dibujando CheckBox Linux.";
    }
};
```

```cpp
class WinFactory : public AbstractGUIFactory {
public:
    AbstractButton* CrearButton() const override {
        return new WinButton();
    }
    AbstractCheckBox* CrearCheckBox() const override {
        return new WinCheckBox();
    }
};

class MacFactory : public AbstractGUIFactory {
public:
    AbstractButton* CrearButton() const override {
        return new MacButton();
    }
    AbstractCheckBox* CrearCheckBox() const override {
        return new MacCheckBox();
    }
};

class LinuxFactory : public AbstractGUIFactory {
public:
    AbstractButton* CrearButton() const override {
        return new LinuxButton();
    }
    AbstractCheckBox* CrearCheckBox() const override {
        return new LinuxCheckBox();
    }
};
```

```cpp
void Aplication(int os) {
    switch (os){
    case 1: // Windows
        {
            WinFactory* Win = new WinFactory();
            const AbstractButton* btn = Win->CrearButton();
            delete Win;
            std::cout << btn->Draw() << "\n";
            break;
        }
    case 2: // Mac
        {
            MacFactory* Mac = new MacFactory();
            const AbstractButton* btn = Mac->CrearButton();
            delete Mac;
            std::cout << btn->Draw() << "\n";
            break;
        }
    case 3: // Mac
        {
            LinuxFactory* Lnx = new LinuxFactory();
            const AbstractButton* btn = Lnx->CrearButton();
            delete Lnx;
            std::cout << btn->Draw() << "\n";
            break;
        }
    default:
        std::cerr << "Error, OS no valido.";
        return;
    }
}
```

```cpp
int main() {
    std::cout << "Cliente: Windows " << endl;
    // IMPOSIBLE instanciar clases Abstractas
    // AbstractButton* f1 = new AbstractButton();
    Aplication(1); // 1 - Windows
    // delete f1;
    std::cout << std::endl;
    std::cout << "Cliente: Mac " << endl;
    // IMPOSIBLE instanciar clases Abstractas
    // AbstractButton* f2 = new AbstractButton();
    Aplication(2); // 2 - Mac
    //delete f2;
    std::cout << std::endl;
    std::cout << "Cliente: Linux " << endl;
    Aplication(3);
    // En caso no sea un OS implementado
    std::cout << std::endl;
    Aplication(4);
    return 0;
}
```

Resultado:


Cliente: Windows
Dibujando Button Windows
Cliente: Mac
Dibujando Button Mac
Cliente: Linux
Dibujando Button Linux
Error, OS no valido.
<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>