Laboratorio 16

Patrones de Diseño: Abstract Factory

Nombre: Gabriel Fernando Rodriguez Cutimbo

CUI: 20212157

Grupo: B

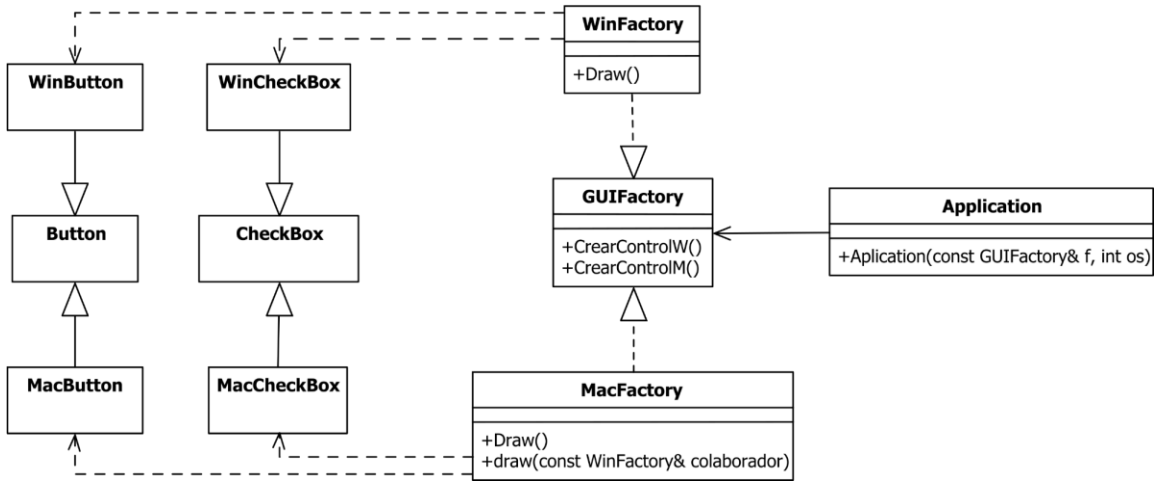
Repositorio GitHub:

https://github.com/gaco123/EPCC_CCII.git

1. Ejercicios

Resolver los siguientes ejercicios planteados:

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)



La función main debe ser:

```
int main() {
	std::cout << "Cliente: Windows ";
	Button* f1 = new Button();
	Aplication(*f1, 1); // 1 - Windows
	delete f1;
	std::cout << std::endl;

	std::cout << "Cliente: Mac ";
	Button* f2 = new Button();

	Aplication(*f2, 2); // 2 - Mac
	delete f2;
	return 0;
}
```

La salida espera debe ser:

Cliente : Windows
Dibujando Button Windows

Cliente: Mac
Dibujando Button Mac

Solución:

Código “GUIFactory.h”:

```
#ifndef GUIFACTORY_H
#define GUIFACTORY_H
#include "WinFactory.h"
#include "MacFactory.h"
#include "LinuxFactory.h"

class GUIFactory{
public:
	virtual WinFactory* CreateControlWin() const = 0;
	virtual MacFactory* CreateControlMac() const = 0;
	virtual LinuxFactory* CreateControlLinux() const = 0;
};
```

```
#endif
```

Código "WinFactory.h":

```
#ifndef WINFACTORY_H
#define WINFACTORY_H
#include <string>
using namespace std;

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

#endif
```

Código "MacFactory.h":

```
#ifndef MACFACTORY_H
#define MACFACTORY_H
#include "WinFactory.h"
#include <string>
using namespace std;

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

#endif
```

Código "LinuxFactory.h":

```
#ifndef LINUXFACTORY_H
#define LINUXFACTORY_H
#include <string>
using namespace std;

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

```
};

#endif
```

Código "CheckBox.h":

```
#ifndef CHECKBOX_H
#define CHECKBOX_H
#include "GUIFactory.h"
#include "WinCheckBox.h"
#include "MacCheckBox.h"
#include "LinuxCheckBox.h"

class CheckBox : public GUIFactory{
public:
	~CheckBox(){};
	WinFactory* CreateControlWin() const override{
		return new WinCheckBox();
	}
	MacFactory* CreateControlMac() const override{
		return new MacCheckBox();
	}
	LinuxFactory* CreateControlLinux() const override{
		return new LinuxCheckBox();
	}
};

#endif
```

Código "WinCheckBox.h":

```
#ifndef WINCHECKBOX_H
#define WINCHECKBOX_H
#include "WinFactory.h"

class WinCheckBox : public WinFactory{
public:
	string Draw() const override{
		return "Drawing Windows CheckBox";
	}
};

#endif
```

Código "MacCheckBox.h":

```
#ifndef MACCHECKBOX_H
#define MACCHECKBOX_H
#include "MacFactory.h"

class MacCheckBox : public MacFactory{
public:
	string Draw() const override{
		return "Drawing Mac CheckBox";
	}
	string draw(const WinFactory& colaborator) const override{
		const string result = colaborator.Draw();
		return "Mac CheckBox with help of " + result;
	}
};

#endif
```

Código "LinuxCheckBox.h":

```
#ifndef LINUXCHECKBOX_H
#define LINUXCHECKBOX_H
#include "LinuxFactory.h"

class LinuxCheckBox : public LinuxFactory{
public:
	string Draw() const override{
		return "Drawing Linux CheckBox";
	}
};

#endif
```

Código "Button.h":

```
#ifndef BUTTON_H
#define BUTTON_H
#include "GUIFactory.h"
#include "WinButton.h"
#include "MacButton.h"
#include "LinuxButton.h"

class Button : public GUIFactory{
public:
	~Button(){};
```

```
        WinFactory* CreateControlWin() const override{
                return new WinButton();
        }
        MacFactory* CreateControlMac() const override{
                return new MacButton();
        }
        LinuxFactory* CreateControlLinux() const override{
                return new LinuxButton();
        }
};

#endif
```

Código "WinButton.h":

```
#ifndef WINBUTTON_H
#define WINBUTTON_H
#include "WinFactory.h"

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

#endif
```

Código "MacButton.h":

```
#ifndef MACBUTTON_H
#define MACBUTTON_H
#include "MacFactory.h"

class MacButton : public MacFactory{
public:
        string Draw() const override{
                return "Drawing Mac Button";
        }
        string draw(const WinFactory& colaborator) const override{
                const string result = colaborator.Draw();
                return "Mac Button with help of " + result;
        }
};

#endif
```

Código “LinuxButton.h”:

```
#ifndef LINUXBUTTON_H
#define LINUXBUTTON_H
#include "LinuxFactory.h"

class LinuxButton : public LinuxFactory{
public:
	string Draw() const override{
		return "Drawing Linux Button";
	}
};

#endif
```

Código “Lab_16.cpp”:

```
//Laboratorio N°16
//Alumno: Gabriel Fernando Rodriguez Cutimbo
/*
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)
*/

#include <iostream>

#include "GUIFactory.h"
#include "WinFactory.h"
#include "MacFactory.h"
#include "LinuxFactory.h"

#include "Button.h"
#include "WinButton.h"
#include "MacButton.h"
#include "LinuxButton.h"

#include "CheckBox.h"
#include "WinCheckBox.h"
#include "MacCheckBox.h"
#include "LinuxCheckBox.h"

using namespace std;

void Application(const GUIFactory& f, int os) {
```

```
        if(os==1){
                const WinFactory* pa = f.CreateControlWin();
                cout << pa->Draw();
                delete pa;
        }
        else if(os==2){
                const MacFactory* pb = f.CreateControlMac();
                cout << pb->Draw();
                delete pb;
        }
        else if(os==3){
                const LinuxFactory* pc = f.CreateControlLinux();
                cout << pc->Draw();
                delete pc;
        }
        else{
                cout<<"Error SO no soportado";
        }
}

int main () {
        //Win
        cout << "Cliente => Windows\n"; // 1 - Windows
        //WinButton
        Button* f1a = new Button();
        Application(*f1a,1);
        delete f1a;
        cout << endl;
        //WinCheckBox
        CheckBox* f1b = new CheckBox();
        Application(*f1b,1);
        delete f1b;
        cout << endl << endl;

        //Mac
        cout << "Cliente => Mac\n"; // 2 - Mac
        //MacButton
        Button* f2a = new Button();
        Application(*f2a,2);
        delete f2a;
        cout << endl;
        //MacCheckBox
        CheckBox* f2b = new CheckBox();
        Application(*f2b,2);
        delete f2b;
        cout << endl << endl;
```

```
        //Linux
        cout << "Cliente => Linux\n"; // 3 - Linux
        //LinuxButton
        Button* f3a = new Button();
        Application(*f3a,3);
        delete f3a;
        cout << endl;
        //LinuxCheckBox
        CheckBox* f3b = new CheckBox();
        Application(*f3b,3);
        delete f3b;

        return 0;
}
```

Funcionamiento:

```
Cliente => Windows
Drawing Windows Button
Drawing Windows CheckBox

Cliente => Mac
Drawing Mac Button
Drawing Mac CheckBox

Cliente => Linux
Drawing Linux Button
Drawing Linux CheckBox
```

2. Entregables

Al final estudiante deberá:

1. Compactar el código elaborado y subirlo al aula virtual de trabajo. Agregue sus datos personales como comentario en cada archivo de código elaborado.
2. Elaborar un documento que incluya tanto el código como capturas de pantalla de la ejecución del programa. Este documento debe de estar en formato PDF.
3. El nombre del archivo (comprimido como el documento PDF), será su LAB16_GRUPO_A/B/C_CUI_1erNOMBRE_1erAPELLIDO.

   (Ejemplo: LAB16_GRUPO_A _2022123_PEDRO_VASQUEZ).
4. Debe remitir el documento ejecutable con el siguiente formato:

   LAB16_GRUPO_A/B/C_CUI_ EJECUTABLE_1erNOMBRE_1erAPELLIDO

   (Ejemplo: LAB16_GRUPO_A_EJECUTABLE_2022123_PEDRO_VASQUEZ).

En caso de encontrarse trabajos similares, los alumnos involucrados no tendrán evaluación y serán sujetos a sanción.