# UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

## FACULTAD DE INGENIERIA DE PRODUCCION Y SERVICIOS

### ESCUELA PROFESIONAL DE CIENCIAS DE LA COMPUTACIÓN


**LABORATORIO 15 – Patrones de Diseño - Builder.**

**DOCENTE:**

Enzo Edir Velásquez Lobatón

**ALUMNO:**

Owen Haziel Roque Sosa.

**FECHA:**

30/06/2022

## Arequipa – Perú

1. El alumno deberá de implementar un conjunto de clases que permita seleccionar las piezas de un automóvil, es decir, se podrán tener componentes a disposición del cliente (puertas, llantas, timón, asientos, motor, espejos, vidrios, etc.). Del cual el cliente puede indicar que características de color puede tener cada pieza. Al final mostrar opciones al Cliente o permitirle que él pueda escoger las piezas e indicar el color. Utilizar el patrón Builder.
   *Pista, en lugar de trabajar el producto con una lista de componentes, se puede alojar una estructura o clase.

```
class Auto {
public:
    vector<string> componentes;
    void ListaComp() const {
        cout << "----------------Componentes del automovil construido--------------\n\n";
        cout << "Color: " << componentes[0] << endl;
        cout << "Numero de asientos: " << componentes[1] << endl;
        cout << "\t -Material: " << componentes[5] << endl;
        cout << "Caracteristicas del Motor: \n";
        cout << "\t -Combustible: " << componentes[2] << endl;
        cout << "\t -Numero de Cilindros: " << componentes[3] << endl;
        cout << "Tipo de neumaticos: " << componentes[4] << endl;
        cout << "Tipo de direccion automotriz: " << componentes[6] << endl;
        cout << "Tipo de ventanas: " << componentes[7] << endl;
        cout << "Material de la Carrocería: " << componentes[8] << endl;
        cout << "\t -" << componentes[9] << endl;
        cout << "\n\n";
    }
};
```

```
class CarBuilder {
public:
    virtual ~CarBuilder() {}
    // color del auto
    virtual void setColor(string color) const = 0;
    // cantidad de asientos
    virtual void setAsientos(int asientos) const = 0;
    // Tipo del motor, deportivo, normal, etc.
    virtual void tipoMotor(Motor motor) const = 0;
    // Tipo / modelo de los neumaticos
    virtual void tipoNeumaticos(string neumaticos) const = 0;
    // material de los asientos
    virtual void tipoAsientos(string mat_asientos) const = 0;
    // tipo de direccion automotriz, (timon)
    virtual void tipoDireccion(string timon) const = 0;
    // tipos de ventanas: Vidrio reforzado, laminado, polarizado
    virtual void tipoVentanas(string ventanas) const = 0;
    // material de la carrorecia
    virtual void tipoCarroceria(string material) const = 0;
    // si el auto posee sistema de bolsas de aire
    virtual void hasAirBag(bool res) const = 0;
};
```

```
class BuilderEspecifico : public CarBuilder {
private:
    Auto* car;
public:
    BuilderEspecifico() {
        this->Reset();
    }
    ~BuilderEspecifico() {
        delete car;
    }
    void Reset() {
    void setColor(string color) const override {
        this->car->componentes.push_back(color);
    }
    void setAsientos(int asientos) const override {
    void tipoMotor(Motor motor) const override {
        this->car->componentes.push_back(motor.combustible);
        this->car->componentes.push_back(motor.ncil);
    }
    void tipoNeumaticos(string neumaticos) const override {
    void tipoAsientos(string mat asientos) const override {
    void tipoDireccion(string timon) const override {
    void tipoVentanas(string ventanas) const override {
    void tipoCarroceria(string material) const override {
    void hasAirBag(bool res) const override {
        if (res)
            this->car->componentes.push_back("Posee sistema de Bolsas de Aire");
        else
            this->car->componentes.push_back("NO Posee sistema de Bolsas de Aire");
    }
    Auto* GetAutomovil() {
        Auto* resultado = this->car;
        this->Reset();
        return resultado;
    }
};
```

```
class Director {
private:
    CarBuilder* builder;
public:
    void set_builder(CarBuilder* builder) {
        this->builder = builder;
    }
    void BuildNormalCar() {
        this->builder->setColor("Blanco");
        this->builder->setAsientos(4);
        Motor motorNormalCar = {
            Combustible[1],
            NCilindro[2]
        };
        this->builder->tipoMotor(motorNormalCar);
        this->builder->tipoNeumaticos("Neumaticos Todo Tiempo");
        this->builder->tipoAsientos("Cuero");
        this->builder->tipoDireccion(Direccion[0]);
        this->builder->tipoVentanas(Ventanas[1]);
        this->builder->tipoCarroceria(Carroceria[0]);
        this->builder->hasAirBag(false);
    }
    void BuildBulletProofCar() {
        this->builder->setColor("Negro");
        this->builder->setAsientos(6);
        Motor motorBpCar = {
            Combustible[1],
            NCilindro[2]
        };
        this->builder->tipoMotor(motorBpCar);
        this->builder->tipoNeumaticos("Neumatico Todo Terreno");
        this->builder->tipoAsientos("Aramida");
        this->builder->tipoDireccion(Direccion[1]);
        this->builder->tipoVentanas(Ventanas[0]);
        this->builder->tipoCarroceria(Carroceria[3]);
        this->builder->hasAirBag(true);
    }
};
```

```
void ClienteCode(Director& director)
{
    BuilderEspecifico* builder = new BuilderEspecifico();
    director.set_builder(builder);
    Auto* p;
    cout << "-------------FABRICACION DE AUTOMOVILES--------------\n";
    cout << "C.- Construir Automovil Comun" << endl;
    cout << "B.- Construir Automovil Blindado" << endl;
    cout << "P.- Construir Automovil Personalizado" << endl;
    cout << "X.- Salir" << endl;
    char opt;
    cin >> opt;
    switch (opt) {
    case 'C': // A. Comun
        {
    case 'B': // A. Blindado
        {
    case 'P': // A. Personalizado
        {
    case 'X':
        break;
    }
    delete builder;
}
int main() {
    Director* director = new Director();
    ClienteCode(*director);
    delete director;
    return 0;
}
```

Vectores definidos usados para parametrizar las opciones de ciertos componentes del automóvil:

```
const vector<string> Direccion {"Direccion Mecanica","Direccion Hidraulica",
    "Direccion Electrohidraulica","Direccion  Electromecanica"};
const vector<string> Ventanas {"Vidrio Blindado","Vidrio Laminado",
    "Vidrio Polarizado"};
const vector<string> Carroceria {"Aleacion de Hierro","Aleacion de Aluminio",
    "Aleacion de Magnesio","Hierro con Acero Balistico"};
const vector<string> Combustible {"Gasolina","Diesel","Gas","Electrico"};
const vector<string> NCilindro {"Mono","Doble","Multiple"};

struct Motor {
    string combustible;
    string ncil;
};
```

Case ‘P’: Construir Auto Personalizado

```
int nasientos, comb, cil, dir, vent, carr;
string col, neum, matseats;
char airb;
cout << "Color: ";
cin.ignore();
getline(cin, col);
cout << "N° Asientos: ";
cin >> nasientos;
cout << "Material de los asientos: ";
cin.ignore();
getline(cin, matseats);
cout << "Aspectos del motor: \n";
cout << "Tipo de combustible: " << endl;
for (size_t i = 0; i < Combustible.size(); i++)
    cout << "\t [" << i << "]->" << Combustible[i] << endl;
cin >> comb;
cout << "N° de cilindros: " << endl;
for (size_t i = 0; i < NCilindro.size(); i++)
    cout << "\t [" << i << "]->" << NCilindro[i] << endl;
cin >> cil;

Motor motorCustomCar = {
    Combustible[comb],
    NCilindro[cil]
};
cout << "Tipo neumaticos: ";
cin.ignore();
getline(cin, neum);
cout << "Direccion automotriz: " << endl;
for (size_t i = 0; i < Direccion.size(); i++)
    cout << "\t [" << i << "]->" << Direccion[i] << endl;
cin >> dir;
cout << "Tipo de Ventanas: " << endl;
for (size_t i = 0; i < Ventanas.size(); i++)
    cout << "\t [" << i << "]->" << Ventanas[i] << endl;
cin >> vent;
cout << "Material de la Carroceria: " << endl;
for (size_t i = 0; i < Carroceria.size(); i++)
    cout << "\t [" << i << "]->" << Carroceria[i] << endl;
cin >> carr;
cout << "Bolsas de Aire?\ty/n" << endl;
cin >> airb;
bool AirB = (airb == 'y') ? true : false;
system("pause");
system("CLS");
builder->setColor(col);
builder->setAsientos(nasientos);
builder->tipoMotor(motorCustomCar);
builder->tipoNeumaticos(neum);
builder->tipoAsientos(matseats);
builder->tipoDireccion(Direccion[0]);
builder->tipoVentanas(Ventanas[1]);
builder->tipoCarroceria(Carroceria[0]);
builder->hasAirBag(AirB);
p = builder->GetAutomovil();
p->ListaComp();
delete p;
break;
```

```
-------------FABRICACION DE AUTOMOVILES--------------
C.- Construir Automovil Comun
B.- Construir Automovil Blindado
P.- Construir Automovil Personalizado
X.- Salir
P
Color: azul
N° Asientos: 6
Material de los asientos: Cuerina
Aspectos del motor:
Tipo de combustible:
         [0]->Gasolina
         [1]->Diesel
         [2]->Gas
         [3]->Electrico
1
N° de cilindros:
         [0]->Mono
         [1]->Doble
         [2]->Multiple
2
Tipo neumaticos: Neumaticos Todo Tiempo
Direccion automotriz:
         [0]->Direccion Mecanica
         [1]->Direccion Hidraulica
         [2]->Direccion Electrohidraulica
         [3]->Direccion  Electromecanica
3
Tipo de Ventanas:
         [0]->Vidrio Blindado
         [1]->Vidrio Laminado
         [2]->Vidrio Polarizado
2
Material de la Carroceria:
         [0]->Aleacion de Hierro
         [1]->Aleacion de Aluminio
         [2]->Aleacion de Magnesio
         [3]->Hierro con Acero Balistico
2
Bolsas de Aire? y/n
y
```

```
----------------Componentes del automovil construido--------------

Color: azul
Numero de asientos: 6
         -Material: Cuerina
Caracteristicas del Motor:
         -Combustible: Diesel
         -Numero de Cilindros: Multiple
Tipo de neumaticos: Neumaticos Todo Tiempo
Tipo de direccion automotriz: Direccion Mecanica
Tipo de ventanas: Vidrio Laminado
Material de la Carrocería: Aleacion de Hierro
         -Posee sistema de Bolsas de Aire



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