Clases y Objetos en C++

Definición del concepto de Clase

En el Paradigma Orientado a Objetos (POO) una *clase* es una especie de plantilla que permite la descripción de cualquier objeto o persona mediante la definición de una serie de características llamadas *atributos*.

Tales atributos pueden ser definidos mediante el uso de los tipos de datos primitivos como *int, double, char o bool*, o a través de otros objetos, como string.

Los atributos son manipulados por medio de funciones o métodos (*comportamiento*) definidos por el programador o propias del lenguaje, que permiten alterar, cambiar, asignar o leer los valores asociados a los atributos declarados en la clase.

Ejemplo de Clase

Como se menciono anteriormente, en POO una *clase* es una plantilla que permite describir cualquier objeto (cosa, persona, lugar, …etc) de la vida real mediante características (*atributos*) que son manipulados con una serie de métodos (*comportamiento*).

En este sentido, podemos pensar en los siguientes objetos que pueden ser generalizados por medio de una *clase*.

Cliente de un banco
Número de id
Apellido
Nombre
Número cuenta

Estudiante
Código estudiantil
Apellido
Nombre
Código plan

Computador personal
Marca
Modelo
Número de referencia
Capacidad en RAM
Capacidad en DD
Tamaño de pantalla
Sistema Operativo
Arquitectura

Clase

Atributos

Definición del concepto de Objeto

Básicamente un objeto en POO es la representación mediante una variable especial o compuesta de alguna entidad de la vida real. Es decir, un *objeto* es un conjunto de datos y funciones relacionadas que permiten definir de manera especifica algo.

Se dice entonces que una *clase* es la plantilla que generaliza el concepto y un *objeto* es la representación especifica.

En los ejemplos anteriores las clases Cliente, Estudiante y Computador Personal son generalizaciones, pero no describen a un cliente, estudiante o computador de manera especifica.

Ejemplo de Objeto

Siguiendo con los ejemplos de definiciones de clase, tomaremos el de Cliente de una entidad bancaria, y definiremos 3 objetos para representar de forma especifica a 3 clientes.

Gráficamente lo anterior se vería así:

Cliente de un banco
Número de id
Apellido
Nombre
Número cuenta

Objeto1	
Atributos	**Valores**
Número de id	10
Apellido	LASSO
Nombre	LUIS
Número cuenta	100010

Objeto2	
Atributos	**Valores**
Número de id	20
Apellido	ROA
Nombre	ANA
Número cuenta	100020

Objeto3	
Atributos	**Valores**
Número de id	30
Apellido	RIOS
Nombre	EVA
Número cuenta	100030

En este caso el **Objeto1** representa al cliente LUIS.

El **Objeto2** representa al cliente ANA, y el **Objeto3** representa al cliente EVA.

Definición del concepto de Comportamiento (1)

El comportamiento son las funciones o métodos definidos por el programador o propias del lenguaje, que permiten cambiar el estado del objeto.

Dicho de otra forma, con el uso de tales métodos se pueden cambiar los valores de los atributos del objeto.

Como una buena practica de programación en un lenguaje O.O. se utilizan como estándar los métodos ***set*** y ***get*** para cambiar o retornar el valor de cada atributo definido en la clase. Asimismo, también es usada la sobrecarga de métodos, como estrategia para ampliar la funcionalidad (manera de utilizarse) la clase y asignar los valores a los objetos declarados.

Definición del concepto de Comportamiento (2)

En el ejemplo anterior, algunos de los métodos (comportamiento) que podrían estar definidos en la clase ***ClienteBanco*** de un banco serian:

```
void setNumID(int numero){
    numeroID = numero;
}

void setNumID(){
    cout<<"Ingrese el número de identificación: ";
    cin>>numeroID;
}

long getNumID(){
    return numeroID;
}
```

Estructura básica de una clase

```
//Directivas de definición de la clase
#ifndef NOMBRECLASE_H
#define NOMBRECLASE_H
//Inclusión de archivos de cabecera y librerías
#include<archivo1>
#include<archivo2>
#include<archivoN>

class NombreClase{ //uso de la notación CamelCase
   Sección pública:
      declaración de atributos, constructores, destructores y métodos //uso de la notación CamelCase
  Sección protegida:
     declaración de atributos, constructores, destructores y métodos //uso de la notación CamelCase
  Sección privada:
     declaración de atributos, constructores, destructores y métodos //uso de la notación CamelCase
};
#endif
```

Definición de la clase ClienteBanco

```
#ifndef CLIENTEBANCO_H
#define CLIENTEBANCO_H
#include<iostream>
using namespace::std;

class ClienteBanco{
    public:
        long numeroID, numeroCuenta;
        string apellido, nombre;

        ClienteBanco() {}
        virtual ~ClienteBanco() {}

    void setNumID(int numero){
        numeroID = numero;
    }

    void setNumID(){
        cout<<"Ingrese el número de identificación: ";
        cin>>numeroID;
    }

    long getNumID(){
        return numeroID;
    }

    protected:

    private:
};

#endif // CLIENTEBANCO_H
```

Definición de la clase

Archivos de cabecera y librerías

Atributos de la clase

Método(s) constructor y destructor

Sección pública

Métodos ***set*** y ***get*** para cambiar el estado del objeto declarado de

Sección protegida y privada

Fin de la definición de la clase

Definición de un objeto de la clase ClienteBanco

Un objeto se declara mediante la siguiente sintaxis:

NombreClase nombreObjeto;

Siguiendo con el ejemplo anterior, la definición de los objetos que representan a 3 clientes serian así:

ClienteBanco objeto1;//objeto para almacenar los datos de LUIS

ClienteBanco objeto2;//objeto para almacenar los datos de ANA

ClienteBanco objeto3;//objeto para almacenar los datos de EVA

Ejemplo de implementación

La carpeta “Entidad bancaria” dispuesta en el campus contiene la implementación de una aplicación en C++ que define la clase ClienteBanco (.h) y un archivo de prueba (.cpp) donde se declaran 3 objetos y se ingresan los datos para cada uno de ellos.