Laboratorio de Computación II

Programación orientada a objetos
Composición

Características de la POO

- Construcción y destrucción: La capacidad de realizar acciones al crearse y al destruirse el objeto de manera automática.
- Encapsulamiento: Restricción del acceso a elementos (propiedades y métodos) que no deban accederse desde fuera de clase. Pueden accederse si se les desarrolla un método público para tal fin.
- Composición y agregación: Diferentes maneras de relacionar clases.
- Herencia: Poder crear una clase derivada a partir de una clase base de la cual obtendrá todos los rasgos.
- Polimorfismo: La posibilidad de que varios objetos de diferentes clases, pero con una base común, se comporten diferente ante una misma acción.

Composición

Tipo de relación entre clases que permite que una clase A componga a una clase B.

```
class Fecha{
  private:
    int dia, mes, anio;
  public:
     Fecha(int, int, int);
};
class Persona{
  private:
    char apellido[50];
    char nombre[50];
    Fecha nacimiento;
};
```


Como se puede observar, una Persona está compuesta por una Fecha. Es decir, *<<tiene un>>* atributo de tipo Fecha como elemento privado.
La composición se caracteriza por destruir junto con el objeto compuesto todos los objetos que lo componen. No puede existir la fecha de nacimiento si ya no existe el objeto persona.

Encapsulamiento y composición

```
class Fecha{
  private:
    int dia, mes, anio;
  public:
     Fecha(int, int, int);
};
class Persona{
  private:
    char apellido[50];
    char nombre[50];
    Fecha nacimiento;
  public:
     cargar();
};
```

Como es de esperarse, no se puede acceder directamente a los miembros privados de los objetos que componen a nuestra clase ya que los mismos se encuentran encapsulados. Necesitamos hacer uso de métodos públicos para accederlos.

```
void Persona::cargar(){
  cin >> apellido;
  cin >> nacimiento.dia;
}
```

Encapsulamiento y composición

```
class Fecha{
  private:
    int dia, mes, anio;
  public:
     Fecha(int, int, int);
};
class Persona{
  private:
    char apellido[50];
    char nombre[50];
    Fecha nacimiento;
  public:
     cargar();
};
```

Siguiendo la misma lógica, tampoco podremos acceder desde afuera de la clase compuesta a ninguno de los elementos privados. Necesitaremos un método público que nos permita hacerlo.

```
int main(){
  Persona p;
  cout << p.nacimiento.getDia();
}
```

Composición

```
class Auto{
  private:
    Puerta *puertas;
    Rueda ruedas[4];
    Motor motor;
    int velocidad;
  public:
    Auto(int p, Motor m){
      puertas = new Puerta[p];
      motor = m;
      velocidad = 0;
    }
};
int main(){
  Motor oMotor("Diesel");
  Auto coche(5, oMotor);
  coche.mostrar();
}
```

Una simplificación de la clase Auto está compuesta por un vector de cuatro elementos de Rueda, un objeto Motor y un puntero a Puerta.
El constructor reciba la cantidad de puertas (y pide memoria dinámica para crear el vector) y también recibe un objeto motor.
Asigna otra de las propiedades que es la velocidad a 0.

Ejemplos en C/C++