UNCUYO
UNIVERSIDAD NACIONAL DE CUYO

FACULTAD DE INGENIERÍA

Programación Orientada a Objetos

Clases y Objetos en C++ y Python

Esp. Ing. César Aranda

cesar.aranda@ingeniería.uncuyo.edu.ar

Ingeniería en Mecatrónica

Objetivos y contenidos

- Analizar los detalles de implementación de clases, miembros de datos y funciones miembro, instanciación, comunicación entre objetos, constructor y sobrecarga
- Discutir aspectos básicos de implementación en lenguajes C++ y Python

Clases en C++ y Miembros de la Clase

AutoEstereo
Marca : String Modelo : String Estado : Boolean Volumen : int FrecuenciaTipo : String Frecuencia : float EstadoUSB : int
encender() : void apagar() : void subirVolumen() : void bajarVolumen() : void cambiarAMFM() : void subirFrecuencia() : void bajarFrecuencia() : void

```
6  class AutoEstereo {
   public:
8      void encender();
9      void apagar();
10     void subirVolumen();
11     void bajarVolumen();
12     void cambiarAMFM();
13     void subirFrecuencia();
14     void bajarFrecuencia();
15
   private:
17     string Marca;
18     string Modelo;
19     bool Estado;
20     int Volumen;
21     string FrecuenciaTipo;
22     float Frecuencia;
23     int EstadoUSB;
24 };
```

Funciones Miembro

(métodos u operaciones)

Miembros de Datos

(atributos o propiedades)

Ing. César Aranda 3

Ajuste de Archivos Header y Definition

AutoEstereo.h + **AutoEstereo.cpp**

```
4  using namespace std;
5
6  class AutoEstereo {
7    public:
8      void encender();
9      void apagar();
10     void subirVolumen();
11     void bajarVolumen();
12     void cambiarAMFM();
13     void subirFrecuencia();
14     void bajarFrecuencia();
15
16   private:
17     string Marca;
18     string Modelo;
19     bool Estado;
20     int Volumen;
21     string FrecuenciaTipo;
22     float Frecuencia;
23     int EstadoUSB;
24 };
```

```
1  #include "AutoEstereo.h"
2
3  void AutoEstereo::encender() {
4      Estado = true;
5  }
6
7  void AutoEstereo::apagar() {
8      Estado = false;
9  }
10
11 void AutoEstereo::subirVolumen() {
12     if (Volumen < 100) Volumen++;
13 }
14
15 void AutoEstereo::bajarVolumen() {
16     if (Volumen > 0) Volumen--;
17 }
18
19 void AutoEstereo::cambiarAMFM() {
20     if (FrecuenciaTipo == "AM")
21         FrecuenciaTipo = "FM";
22     else
23         FrecuenciaTipo = "A";
24 }
```

Ing. César Aranda

Instanciación y Comunicación en C++

```
11  #include "AutoEstereo.h"
12
13  int main(int argc, char** argv) {
14      AutoEstereo obj_est;
15      obj_est.encender();
16      obj_est.apagar();
17      return 0;
18  }
```

Sólo se referencia al .h

Variable Objeto

```
15      e
16      est.
17      e
18      r
19  }
20
21
```

apagar() void
bajarFrecuencia() void
bajarVolumen() void
cambiarAMFM() void
encender() void
subirFrecuencia() void
subirVolumen() void

```
11  #include "AutoEstereo.h"
12
13  int main(int argc, char** argv) {
14      AutoEstereo *pobj_est;
15      pobj_est = new AutoEstereo;
16      pobj_est->encender();
17      pobj_est->apagar();
18      delete pobj_est;
19      return 0;
20  }
```

Puntero a Objeto

Mensaje
{de main a pobj_est}

5

Ocultamiento ➔ Visibilidad en C++

```
11  #include "AutoEstereo.h"
12
13  int main(int argc, char** argv) {
14      AutoEstereo obj_est;
15      obj_est.encender();
16      obj_est.apagar();
17      return 0;
18  }
```

```
8   #include <iostream>
9   using namespace std;
10
11  #include "AutoEstereo.h"
12
13  int main(int argc, char** argv) {
14      AutoEstereo est;
15      est.encender();
16      cout << est.Estado;
17      est.apagar();
18      return 0;
19  }
```

```
4   using namespace std;
5
6   class AutoEstereo {
7    public:
8       void encender();
9       void apagar();
10      void subirVolumen();
11      void bajarVolumen();
12      void cambiarAMFM();
13      void subirFrecuencia();
14      void bajarFrecuencia();
15
16   private:
17      string Marca;
18      string Modelo;
19      bool Estado;
20      int Volumen;
21      string FrecuenciaTipo;
22      float Frecuencia;
23      int EstadoUSB;
24  };
```

public | protected | private .....y friend

6

Funciones Miembro get/set (1)

```
13  int main(int argc, char** argv) {
14      AutoEstereo *pobj_est;
15      pobj_est = new AutoEstereo;
16      pobj_est->encender();
17      cout << "Estado: " << pobj_est->getEstado() << endl;
18      pobj_est->apagar();
19      cout << "Estado: " << (pobj_est->getEstado() ? "On" : "Off");
20      cout << endl << endl;
21      delete pobj_est;
22      return 0;
23  }
```

```
Estado: 1
Estado: Off

RUN SUCCESSFUL (t
```

AutoEstereo.h

```
4   #include <string>
5   using namespace std;
6
7   class AutoEstereo {
8   public:
16      void setFrecuencia(float Frecuencia);
17      void setFrecuenciaTipo(string FrecuenciaTipo);
18      bool getEstado();
19      float getFrecuencia();
20      string getFrecuenciaTipo();
21      int getVolumen();
```

AutoEstereo.cpp

```
43
44  bool AutoEstereo::getEstado() {
45      return this->Estado;
46  }
```

Funciones Miembro get/set (2)

```
17  int main(int argc, char** argv) {
18      AutoEstereo *pobj_est;
19      pobj_est = new AutoEstereo;
20      pobj_est->setFrecuencia(100.3);
21      cout << "Frecuencia Actual: " << pobj_est->getFrecuencia() << endl;
22
23      pobj_est->setFrecuencia(99.9);
24      cout << "Frecuencia Actual: " << pobj_est->getFrecuencia() << endl;
25
26      for (int i = 0; i < 6; i++)
27          pobj_est->bajarFrecuencia();
28      cout << "Frecuencia Actual: " << pobj_est->getFrecuencia() << endl;
```

```
Frecuencia Actual: 100.3
Frecuencia Actual: 99.9
Frecuencia Actual: 99.3
Estado: 1
Estado: Off
```

AutoEstereo.cpp

```
43  void AutoEstereo::setFrecuencia(float Frecuencia) {
44      this->Frecuencia = Frecuencia;
45  }
46
47  void AutoEstereo::setFrecuenciaTipo(string FrecuenciaTipo) {
48      this->FrecuenciaTipo = FrecuenciaTipo;
49  }
```

Ing. César Aranda 8

Constructor

```
12  int main(int argc, char** argv) {
13      AutoEstereo *pobj_est;
14
15      pobj_est = new AutoEstereo("Alpine ", "iXA-W404");
16      cout << "Marca: " << pobj_est->getMarca() << endl;
17      cout << "Modelo: " << pobj_est->getModelo() << endl;
18
```

```
7   class AutoEstereo {
8   public:
    ..........................................
10      AutoEstereo(string Marca, string Modelo);
    ..........................................
24      string getMarca();
25      string getModelo();
```

```
Marca: Alpine
Modelo: iXA-W404
Frecuencia Actual: 100.3
Frecuencia Actual: 99.9
Frecuencia Actual: 99.3
Estado: 1
Estado: 0ff
```

```
7   AutoEstereo::AutoEstereo(string Marca, string Modelo) {
8       this->Marca = Marca;
9       this->Modelo = Modelo;
10  }
```

AutoEstereo.cpp

Ing. César Aranda 9

Sobrecarga de Metodos

```
22      pobj_est->setFrecuencia(99.9);
23      cout << "Frecuencia Actual: " << pobj_est->getFrecuencia() << endl;
    ..........................................
29      pobj_est->setFrecuencia(103.1, "FM");
30      cout << "Frecuencia Actual: " << pobj_est->getFrecuencia()
31           << "[ " << pobj_est->getFrecuenciaTipo() << " ]" << endl;
```

```
Modelo: iXA-W404
Frecuencia Actual: 100.3
Frecuencia Actual: 99.9
Frecuencia Actual: 99.3
Frecuencia Actual: 103.1[ FM ]
Estado: 1
Estado: 0ff
```

AutoEstereo.h

```
18      void setFrecuencia(float Frecuencia);
19      void setFrecuencia(float Frecuencia, string FrecuenciaTipo);
20      void setFrecuenciaTipo(string FrecuenciaTipo);
```

```
48  void AutoEstereo::setFrecuencia(float Frecuencia) {
49      this->Frecuencia = Frecuencia;
50  }
51
52  void AutoEstereo::setFrecuencia(float Frecuencia, string FrecuenciaTipo) {
53      this->Frecuencia = Frecuencia;
54      this->FrecuenciaTipo = FrecuenciaTipo;
55  }
```

AutoEstereo.cpp

10

Constructor Sobrecargado

```
13      AutoEstereo *pobj_est;
14
15      pobj_est = new AutoEstereo("Alpine ", "iXA-W404");
16      cout << "Marca: " << pobj_est->getMarca() << endl;
37      cout << endl << endl;
38
39      pobj_est = new AutoEstereo();
40      cout << "Marca: " << pobj_est->getMarca() << endl;
41      cout << "Modelo: " << pobj_est->getModelo() << endl;
```

```
Marca: Alpine
Modelo: iXA-W404
Frecuencia Actual: 100.3
Frecuencia Actual: 99.9
Frecuencia Actual: 99.3
Frecuencia Actual: 103.1[ FM ]
Estado: 1
Estado: Off

Marca: Pichicho
Modelo: Generico
```

AutoEstereo.h

```
7   class AutoEstereo {
8   public:
9       AutoEstereo();
10      AutoEstereo(string Marca, string Modelo);
11      AutoEstereo(string FrecuenciaTipo, float Frecuencia);
12      void Encender();
```

```
3   AutoEstereo::AutoEstereo() {
4       this->Marca = string("Pichicho");
5       this->Modelo = string("Generico");
6   }
```

AutoEstereo.cpp

Clases y Objetos en Python

```
class Star:
    """Clase para estrellas (ejemplo)
       Archivo: star.py
    """

    # Numero total de estrellas
    num_stars = 0

    def __init__(self, name):
        self.name = name
        Star.num_stars += 1

    def __str__(self):
        return "Estrella {}".format(self.name)
```

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import star

if __name__ == "__main__":

    estrella1 = star.Star('Altair')

    print(estrella1.name)
```

```
class Star(object):
    ''' Clase para estrellas Equivalente '''
```

Clases + Objetos de Clase y de Instancia

```
class Auto:
    """Clase para autos (ejemplo) // Archivo

    num_autos = 0        # Numero total de au

    def __init__(self, marca):
        self.marca = marca
        Auto.num_autos += 1

    def __str__(self):
        return "Auto: {}".format(self.marca)

    def set_veloc(self, veloc):
        self.veloc = veloc

    def set_tiempo(self, tiempo):
        self.tiempo = tiempo

    def get_dist(self):
        print "La distancia del {} es de {:.2f}".format(self.marca, self.veloc*self.tiempo)

    def get_num_autos(self):
        return Auto.num_autos
```

```
#!/usr/bin/python
# -*- coding: utf-8 -

from auto import Auto

if __name__ == "__mai

    auto1 = Auto('Ford')
    auto2 = Auto('Audi')

    print "Auto 1: ", auto1.marca
    print "Auto 2: ", auto2.marca
    print(auto1)
    auto1.set_veloc(100)
    auto1.set_tiempo(2)
    auto1.get_dist()
    print "Total de autos: ", auto1.get_num_autos()
    print "o bien        : ", auto1.num_autos
    print "o bien        : ", Auto.num_autos
    #print help(auto1)
```

```
Auto 1:  Ford
Auto 2:  Audi
Auto: Ford
La distancia del Ford es de 200
Total de autos:  2
o bien        :  2
o bien        :  2
```

Ing. César Aranda 13

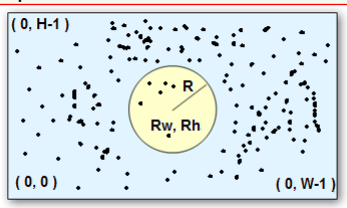
Práctica usando C++ y Python

- Objetivo:
  - Implementar una aplicación de consola, usando OO
- Consigna:

  Dada una circunferencia de radio R ubicada en el centro de una pantalla de W x H, se desea calcular el porcentaje de 1000 puntos generados aleatoriamente que caen dentro de ella.

  R, W y H son valores dados por el operador, cumpliéndose las siguientes restricciones:
  - 50<R<600 pixels
  - Máximo W es 1024 pixels
  - Máximo H es 768 pixels

Ejemplo genérico



Observar la relación de aspecto considerando Rw y Rh y los límites de coordenadas de pantalla

Ing. César Aranda 14

Referencias C++

- DEITEL, H. M. y DEITEL, P. J. (2009): Cómo Programar en C/C++, 6ª edicion, Prentice-Hall
- ACERA GARCIA, M.A. (2010): C/C++, Anaya Multimedia
- BOOCH, G., RUMBAUGH, J., JACKOBSON, I. (2006): Lenguaje Unificado de Modelado, Manual de Referencia Uml 2.0, Addison-Wesley.
- LARMAN, Craig (2003): UML y patrones. 2ª edición. Prentice Hall.
- SCHMULLER, Joseph (1998): Aprendiendo UML en 24hs. Prentice Hall
- http://www.cplusplus.com/doc/tutorial/classes/
- http://es.wikipedia.org/wiki/C%2B%2B
- http://www.uml.org/

Ing. César Aranda 15

Referencias Python

- Von Rossum, G. (2017): El tutorial de Python 3. Editorial Python Software Foundation. Disponible en URL http://tutorial.python.org.ar/, accedido en 2017.
- http://pyspanishdoc.sourceforge.net/lib/lib.html
- Chazalet, S. (2017): Python3, los fundamentos del lenguaje. 2da edición. Ediciones Eni.
- Pérez castaño, A. (2016): Python fácil. Editorial Marcombo
- Beazley, D. y Jones, B. (2013). Python Cookbook. 3ra edición. O'Reilly Media. California.
- Hinojoza Gutierrez, A.P. (2016): Python, paso a paso. Editorial Ra-Ma. Madrid
- González Duque, R. (2008). Python para todos. UPR http://mundogeek.net/tutorial-python/, accedido en 2015.
- Hinojoza Gutierrez, A.P. (2016): Taller de Python. En URL https://www.psicobyte.com/descargas/taller_python.pdf , accedido en 2016

Ing. César Aranda 16