**Práctica de OO: II,H,U,S y P**

Dado el siguiente programa escrito en el lenguaje O.O. "C++", obtener el

diagrama de clases UML correspondiente usando Ingeniería Inversa.

Cuando termine con el Diagrama de Clases guarde todos sus apuntes. Responda

escribiendo en una hoja con su nombre lo siguiente:

1) ¿Qué imprime este código en la pantalla?.

2) Explique por escrito en una hoja cómo intervienen los conceptos de Upcasting,

Herencia, Sobreescritura y Polimorfismo para que esa salida sea posible.

3) ¿Cómo se modificaría el programa si se contrataran en esta supuesta empresa a

Cosme y Damián como médicos?

4) ¿Qué entiende por Late Binding y Early Binding?

5) En su casa use el CodeBlock para compilar y ejecutar este programa

y agregarle el nuevo requerimiento solicitado.(El código final envíelo por mail)

6) ¿Qué habilidades tienen Oficial Rafael , Tecnico Miguel y Directivo Gabriel?

7) Les pido algo más. Agreguen a la clase Medico atributos nombre y matrícula. En

main agreguen al final la generación de una lista y un set para almacenar a los

objetos Cosme y Damian con sus atributos nombre, matrícula y habilidad cinco.

Luego muestren el contenido de la lista y el set. Luego eliminen a Damian de la lista

y a Cosme del set. Finalmente vacíen ambos Containers y verifíquenlo enviando

el mensaje correspondiente antes de terminar el programa.

Lo que hagan con la lista y el set en el main, verifíquenlo también con TDD.

**NOTA: Responda todo aquello que pueda responder, lo que no, estúdielo y envíelo**

**por mail a: robertogug@aol.com.**

#include <iostream.h>

class Habilidades

{

char habilidad1[15];

char habilidad2[15];

};

class Empleado

{

public:

virtual void imprime\_cargo()

{

cout << "Cargo no definido" << endl;

}

};

class Directivo:public Empleado

{

Habilidades uno;

void imprime\_cargo()

{

cout << "Es un directivo" << endl;

}

};

class Operario:public Empleado

{

Habilidades dos;

void imprime\_cargo()

{

cout << "Es un operario" << endl;

}

};

class Oficial:public Operario

{

Habilidades tres;

void imprime\_cargo()

{

cout << "Es un operario oficial" << endl;

}

};

class Tecnico:public Operario

{

Habilidades cuatro;

void imprime\_cargo()

{

cout << "Es un operario tecnico" << endl;

}

};

void imprimir(Empleado& i)

{

i.imprime\_cargo();

}

main()

{

Oficial Rafael ;

Tecnico Miguel;

Directivo Gabriel;

cout << "Rafael" << endl;

imprimir(Rafael);

cout << "Gabriel" << endl;

imprimir(Gabriel);

cout << "Miguel" << endl;

imprimir(Miguel);

}

1. Este código imprimirá en la pantalla lo siguiente:

Rafael

Es un operario oficial

Gabriel

Es un directivo

Miguel

Es un operario tecnico



class Habilidades

{

char habilidad1[15];

char habilidad2[15];

};

class Empleado

{

public:

virtual void imprime\_cargo() // polimorfismo

{

cout << "Cargo no definido" << endl;

}

};

class Directivo:public Empleado // Hereda de Empleado

{

Habilidades uno;

void imprime\_cargo() // sobrecarga

{

cout << "Es un directivo" << endl;

}

};

class Operario:public Empleado // Hereda de Empleado

{

Habilidades dos;

void imprime\_cargo() // sobrecarga

{

cout << "Es un operario" << endl;

}

};

class Oficial:public Operario // Hereda de Operario

{

Habilidades tres;

void imprime\_cargo() // sobrecarga

{

cout << "Es un operario oficial" << endl;

}

};

class Tecnico:public Operario // Hereda de Operario

{

Habilidades cuatro;

void imprime\_cargo() // sobrecarga

{

cout << "Es un operario tecnico" << endl;

}

};

void imprimir(Empleado &i) //upcasting

{

i.imprime\_cargo();

}

main()

{

Oficial Rafael ;

Tecnico Miguel;

Directivo Gabriel;

cout << "Rafael" << endl;

imprimir(Rafael);

cout << "Gabriel" << endl;

imprimir(Gabriel);

cout << "Miguel" << endl;

imprimir(Miguel);

}

**Explicacion:**

**Sobrecarga**: Existe más de una función con el mismo nombre, esto significa que si se llama a la función imprime\_cargo(); accionara a la clase a la que se le esté apuntando.

**Upcasting**: en la función imprimir(Empleado &i) podemos observar que ocurre un Upcasting, esto significa que no importe la clase que sea &i, esta función fuerza a &i para que sea clase Empleado.

**Polimorfismo**: En la función virtual void imprime\_cargo() , al declarar una función virtual, significa que se cancelara el upcasting generado, y tomando el ejemplo anterior de upcasting, se dejara de forzar a &i a ser Empleado, y volverá a ser su clase original.

1. Para esto se deberá realizar la siguiente modificación:

Se coloca la siguiente clase, donde hereda de Empleado.

class Medico :public Empleado

{

Habilidades cinco;

void imprime\_cargo()

{

cout << "Es un Medico" << endl;

}

};

Definimos a Cosme y a Damian como Medicos y llamamos a la función imprimir() :

Medico Cosme, Damian;

cout << "Cosme" << endl;

imprimir(Cosme);

cout << "Damian" << endl;

imprimir(Damian);

1. Late Binding: es cuando el compilador identifica al objeto en el tiempo de ejecución, y luego acciona a la función correcta, en el polimorfismo, cuando ocurre una función virtual, el compilador lo reconoce al momento de ejecución y acciona a la función correcta.

Early binding: asocia directamente una dirección a la llamada de función. La sobrecarga de funciones, es un ejemplo de Early Binding ya que si un objeto invoca a una función sobrecargada, este mismo apuntara a la función que pertenece dicho objeto.

5)

#include <iostream>

using namespace std;

class Habilidades

{

char habilidad1[15];

char habilidad2[15];

};

class Empleado

{

public:

virtual void imprime\_cargo()

{

cout << "Cargo no definido" << endl;

}

};

class Directivo:public Empleado

{

Habilidades uno;

void imprime\_cargo()

{

cout << "Es un directivo" << endl;

}

};

class Operario:public Empleado

{

Habilidades dos;

void imprime\_cargo()

{

cout << "Es un operario" << endl;

}

};

class Oficial:public Operario

{

Habilidades tres;

void imprime\_cargo()

{

cout << "Es un operario oficial" << endl;

}

};

class Tecnico:public Operario

{

Habilidades cuatro;

void imprime\_cargo()

{

cout << "Es un operario tecnico" << endl;

}

};

class Medico :public Empleado

{

Habilidades cinco;

void imprime\_cargo()

{

cout << "Es un Medico" << endl;

}

};

void imprimir(Empleado &i)

{

i.imprime\_cargo();

}

main()

{

Oficial Rafael ;

Tecnico Miguel;

Directivo Gabriel;

Medico Cosme, Damian;

cout << "Rafael" << endl;

imprimir(Rafael);

cout << "Gabriel" << endl;

imprimir(Gabriel);

cout << "Miguel" << endl;

imprimir(Miguel);

cout << "Cosme" << endl;

imprimir(Cosme);

cout << "Damian" << endl;

imprimir(Damian);

}

6) Todas estas clases poseen un objeto de clase Habilidad, esto significa que tienen los arreglos char habilidad1 y habilidad2.

class Habilidades {

char habilidad1[15];

char habilidad2[15];

//al iniciar Habilidades se le aplican los arreglos habilidad1 y habilidad2.

};