Trabajo Práctico 6:

Ingeniería Inversa, Herencia, Upcasting, Sobreescritura y Polimorfismo.

Dado el siguiente programa escrito en el lenguaje O.O. "C++", pasarlo a Java y luego 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?.

El código imprime:

Rafael
Es un operario Oficial
Miguel
Es un operario Técnico
Gabriel
Es un operario Directivo

2) Explique por escrito en una hoja cómo intervienen los conceptos de NoUpcasting, Herencia, Sobreescritura y Polimorfismo para que esa salida sea posible. Marque en el programa en qué lugares del mismo aparecen estos conceptos.

La herencia consiste de heredar las propiedades de una clase a otra, por ejemplo se ve en el caso de de las clases creadas que heredan de Empleado

Sobreescritura consiste en llamar una función con el mismo nombre pero que se pase parámetros distintos e internamente resuelve de una forma distinta. En el código se observa con la función imprime_cargo que se encuentran en todas las clases, la misma pero resuelven de una manera diferente que al padre.

Polimorfismo consiste en que una función entienda a qué clase tiene que llamar, la misma se resuelve pasando como parámetro además de la clase particular a la clase padre de todas, de esta forma se llamara a los datos correspondiente de la clase hijo si existe, en caso de que no se llamaría a la función padre.

El Upcasting consiste que al hacer un polimorfismo además de llamarse a los datos de las clases hijo siempre se llamaría a la del padre, aunque el hijo tenga la información para poder llamar. Este problema se observa más en el lenguaje de c++ ya que no maneja el Early Binding, para corregir eso hay que agregar el virtual en la función del padre como se observa en el código.

3) ¿Cómo se modificaría el programa si se contrataran en esta supuesta empresa a Cosme y Damián como médicos?

En este caso se tendría que agregar una clase nueva que sería médico que debe heredar de Empleado y crear los mismos en el Main.

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

El Early Binding se usa en la programación estructurada, la cual al crear el ejecutable busca donde se hace referencia a la llamada y se guarda de esa manera sin tener dinamismo.
En cambio el Late Binding funciona una vez creado el ejecutable, lo que consiste es en crear una tabla virtual que tiene todos los punteros a las funciones que es posible que se ejecuten y los distribuye a donde corresponde mientras es ejecutado

5) En su casa use el Eclipse 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?

Los mismos tienen las habilidades creadas pero no se le asignaron datos a las mismas

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.

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);
}
```

Nota: Lo que está en verde no se responde.