UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA INDUSTRIAL Y DE SISTEMAS

CURSO: LENGUAJE DE PROGRAMACIÓN ORIENTADO A OBJETOS

PROFESOR: HANCCO CARPIO, RONY JORDAN

CICLO: 2015-I

ESTUDIANTE: CALACHUA RAMOS, GERSON ALONSO 20134549A

TRABAJO: ENSAYO Nº6

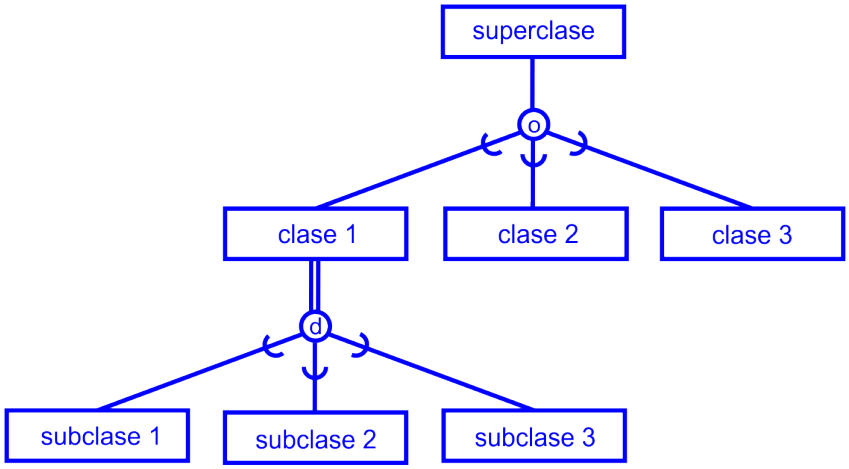
Calachua Ramos, Gerson Alonso 20134549A

SIXTH ESSAY

Siguiendo la dinámica del ensayo anterior este ensayo consta de dos temas, designados por el profesor, estos temas son Herencia y Polimorfismo ambos temas son de mucho interés y utilidad para nosotros quienes nos avocamos a una programación en un lenguaje orientado a objetos.

En primer lugar hablaremos acerca de lo que es la herencia, esta es una capacidad que presentan los lenguajes orientados a objetos para extender las clases. Esta nueva clase denominada “subclase” hereda el comportamiento y los atributos de la clase extendida ahora denominada “superclase”.

A continuación se observa un gráfico sencillo de lo que acabamos de explicar.



Del grafico se puede observar además un dato interesante y es que vemos que una superclase puede tener varias clases o subclases pero todas las subclases tienen una sola superclase, es más se podría decir que la superclase ni siquiera está “enterada” de que tiene o va a tener subclases.

Una subclase posee todos los atributos y métodos de una superclase pero esto no ocurre en sentido inverso; de lo mencionado se deduce que el mecanismo de la herencia es muy útil pues nos permite el ahorro de código, tiempo y eficiencia a la hora de declarar las clases para nuestros objetos ya que hace más sencilla su implementación y nos permite reutilizar el código empleado.

Ahora nos preguntamos ¿Cómo es que una subclase hereda de una superclase?

Pues para hacer esto posible se utiliza la palabra reservada ***extends.***

Ejm:

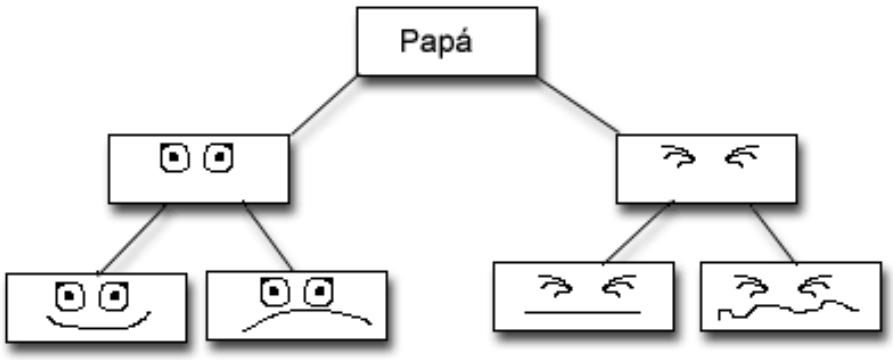
Class NombreSubclase ***extends*** NombreSuperclase {

    Declaraciones de variables y métodos

}

Una clase puede heredar de cualquier otra clase siempre y cuando esta no sea final.

Como ya lo mencionamos, una clase solo puede tener una superclase y esta clase a la ves puede tener tantas subclases como sea necesario (se prefiere, para evitar problemas en los programas, que el máximo de subclases que pueda tener sea 5). Debido a esta regla de “una sola superclase” este mecanismo suele presentarse en la forma de “árbol”.



En java sabemos que todas las clases existen como parte de una jerarquía de herencia. Cuando creamos un objeto su constructor debe ser invocado por cada super y sub clase de la jerarquía.

La palabra ***super*** es la empleada para llamar explícitamente al constructor de una superclase y con ella inicializar los atributos de la clase de la que se hereda. La cual presenta la siguiente sintaxis:

Super(lista de argumentos);

Lo que estamos haciendo al ejecutar esta línea de código es simplemente llamar a los constructores de la superclase y en el campo entre paréntesis donde dice “lista de argumentos” se utiliza para llamar a un constructor en particular o para designar algún valor.

Cabe resaltar que este comando va inmediatamente después de la función creada para los constructores de la subclase para que de esta manera se llamen a los constructores de la superclase y se inicialicen sus atributos.

Dicho todo esto veremos un ejemplo sencillo de cómo se aplica este mecanismo ya estudiado.

```
public class DepositoEstructurado extends Deposito {

   …
   public DepositoEstructurado(Persona titular,
             double capitalFijo, double capitalVariable,
             int plazoDias, double tipoInteresFijo) {

      super(titular, capitalFijo, plazoDias,
            tipoInteresFijo);

      this.capitalVariable = capitalVariable;
   }
   …
}
```

En este ejemplo se observa que la superclase es Deposito mientras que la subclase es DepositoEstructurado. Dentro de la subclase en el constructor se están enviando los valores de “titular, capitalFijo, capitalVariable, plazoDias, tipoInteresFijo”y se observa (como lo mencionamos) que inmediatamente abajo del constructor se escribe el comando “super” esto para llamar a los constructores de la superclase Deposito e inicializar los valores “titular, capitalFijo, plazoDias, tipoInteresFijo” y como capitalvariable no pertenece a la superclase pues sencillamente se creará su constructor en la subclase como observamos.

Finalmente hablaremos de lo que es la “sobrecarga de métodos”, esta es una rama del polimorfismo y de una manera sencilla se podría decir que consta de la creación de varios métodos con el mismo nombre pero que presentan parámetros o argumentos diferentes, sea en tipo o cantidad. Lo que hace java es verificar el tipo y número de valores para seleccionar cual método emplear.

Para explicarlo mejor nos valdremos de la siguiente imagen.

```
class Escritor
{
    public void Escribe(){
        Console.WriteLine("No tengo parámetros");
    }
    public void Escribe(int numero) {
        Console.WriteLine("Número: " + numero.ToString());
    }
    public void Escribe(int numero, string texto) {
        Console.WriteLine("Número: " + numero.ToString() +
            ", Texto: " + texto);
    }
    public void Escribe(int numero, string texto, DateTime fecha){
        Console.WriteLine("Número: " + numero.ToString() +
            ", Texto: " + texto + ", Fecha: " + fecha.ToString());
    }
}
```

Observamos que dentro de la clase Escritor existen 4 métodos con el mismo nombre, en el primero no se tiene ningún parámetro, en el segundo se le envía in entero, el tercero recibe un entero y un texto tipo String, y por último el cuarto recibe un entero, una cadena y un tipo dato fecha.

Ahora si creamos en otra clase un:

Numero numero = new Numero();

Y hacemos:

Numero.set(10);

Pues java verá que le estamos enviando un entero por lo que lo primero que hará es fijarse la cantidad de valores que le estamos mandando, luego de eso verá el tipo de valor que le mandamos y de acuerdo a ello elegirá el método que corresponde; del mismo modo si esta vez le enviamos un entero y una cadena java elegirá el tercer método ya que este le corresponde; cabe resaltar que el orden es muy importante por lo cual si le enviamos una cadena y un entero java nos botará error ya que no existe método alguno que reciba estos parámetros en ese orden.

Terminamos este ensayo esperando que haya sido de ayuda para el lector y le invitamos a ver ejemplos de los temas aquí tratados, viendo los archivos compartidos en el github en la misma ubicación que este ensayo.

BIBLIOGRAFÍA:

- ✓ Fundamentos-de-programacion-en-Java
- ✓ Introduccion-al-lenguaje-de-programacion-Java-UNAM