



Instituto Politécnico Nacional

Escuela Superior de Cómputo

Análisis y Diseño Orientado a Objetos

Profra. Reyna Elia Melara Abarca

Investigación sobre Métricas Orientadas a Objetos



Equipo 3:

Chávez Morones Ángel Uriel
Cuevas Solorza Eduardo
Pacheco Mijangos Mauricio Jesús
Romero Cisneros Victor Gustavo

Grupo: 2CV14

2021/2

Ejercicio 1.

Problema	Medir ayuda a:
Requisitos correctos	Describir los requisitos en términos expresados de manera que puedan ser verificables, medibles, no ambiguos.
Toma de decisiones	El camino a seguir al desarrollar el proyecto está definido, lo que hace posible planear, mejorar, optimizar el trabajo, con lo que se llega a un mejor resultado.
Falta de control	Llevar un monitoreo constante en el diseño, desarrollo y mantenimiento del proyecto es fundamental, debido a que así se conoce el estado del desempeño y progreso del proyecto, gestionando riesgos, innovando y resolviendo conflictos.
Exceso de gastos	Establecer un límite de gastos solo a lo que el proyecto requiera, para así determinar la calidad y cantidad de los recursos necesarios en términos de dinero, esfuerzo capacidad y tiempo a dedicar en el proyecto.
Costos de mantenimiento	Estimar cuántos programadores y recursos se necesitarán para dar mantenimiento al proyecto. Tener definido el costo de mantenimiento ayuda a repartirlo en: mejoramiento, adaptación y corrección.

Ejercicio 2.

Al ingresar el siguiente archivo (Brain.java) a nuestro programa de métricas orientada a objetos:

```
package sources;

public class Brain {
    long neuronas;

    public Brain(){
        neuronas=86_000_000_000L;
    }

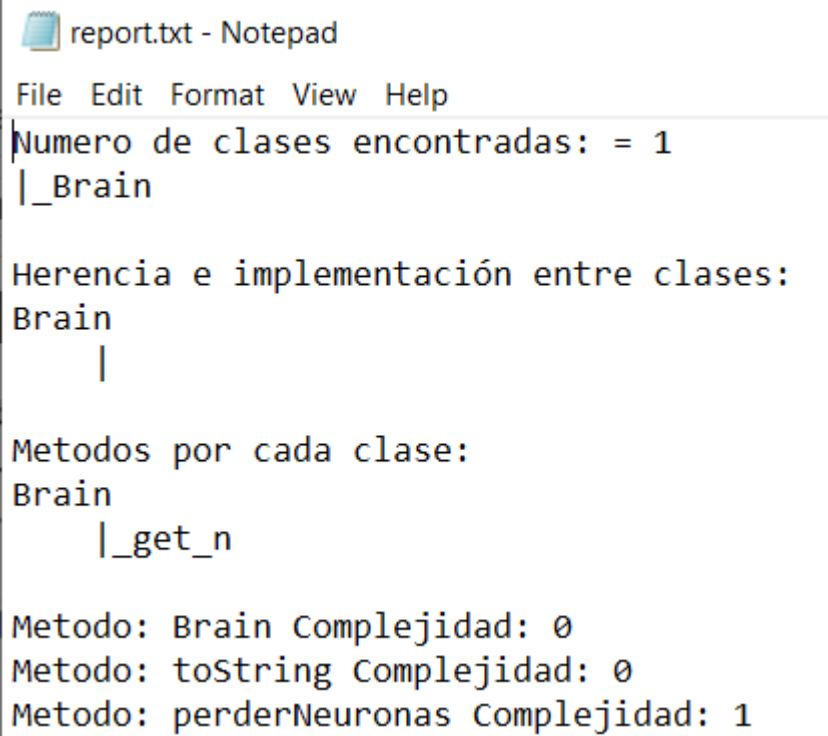
    @Override
    public String toString(){
        return "Numero de neuronas: "+neuronas;
    }
}
```

```

    public void perderNeuronas(long numeroNeuronas){
        if(numeroNeuronas > 0){
            neuronas -= numeroNeuronas;
        }
    }
}

```

Se tiene la siguiente salida en el archivo report.txt, que contiene el desglose de las mencionadas métricas para el archivo ingresado:



```

report.txt - Notepad
File Edit Format View Help
Numero de clases encontradas: = 1
|_Brain

Herencia e implementación entre clases:
Brain
|

Metodos por cada clase:
Brain
|_get_n

Metodo: Brain Complejidad: 0
Metodo: toString Complejidad: 0
Metodo: perderNeuronas Complejidad: 1

```

Ahora, con el archivo Heart.java:

```

package sources;

public class Heart {
    int bpm;

    public Heart(){
        bpm=60;
    }

    @Override
    public String toString(){
        return("BPM = "+bpm);
    }

    public void subirBpm(int aumento){
        bpm+=aumento;
    }
}

```

```

        if(bpm>=100){
            System.out.println("ATENCIÓN: Presión alta!!!!");
        }
    }

    public void bajarBpm(int descenso){
        bpm-=descenso;
        if(bpm<=60){
            System.out.println("ATENCIÓN: Presión baja!!!!");
        }
    }
}

```

Se tiene de salida:



report.txt - Notepad

File Edit Format View Help

Numero de clases encontradas: = 1
|_Heart

Herencia e implementación entre clases:

Heart
|

Metodos por cada clase:

Heart

Metodo: Heart Complejidad: 0

Metodo: toString Complejidad: 0

Metodo: subirBpm Complejidad: 1

Metodo: bajarBpm Complejidad: 1

Finalmente, con el archivo de entrada myFile.txt:

```

public class Tavo{
    public int n;
    public void set_n(int n){
        this.n = n;
    }
    public int get_n(){
        return n;
    }
}

public class Tavito extends Tavo{

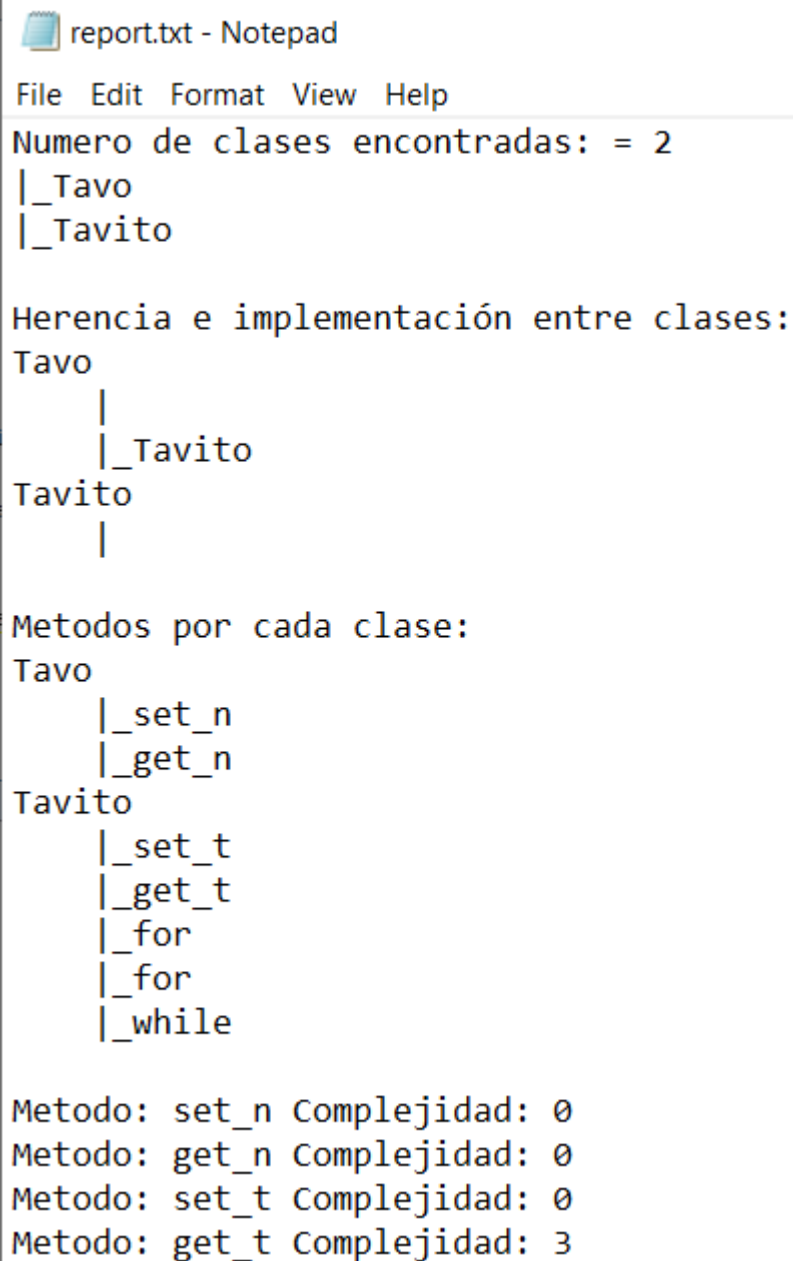
```

```

public int t;
public void set_t(int n){
this.t = n;
}
public int get_t(){
for(){}
for(){}
while(){}
return t;
}
}

```

Y en el archivo report.txt:



```

report.txt - Notepad
File Edit Format View Help
Numero de clases encontradas: = 2
|_Tavo
|_Tavito

Herencia e implementación entre clases:
Tavo
|
|_Tavito
Tavito
|

Metodos por cada clase:
Tavo
|_set_n
|_get_n
Tavito
|_set_t
|_get_t
|_for
|_for
|_while

Metodo: set_n Complejidad: 0
Metodo: get_n Complejidad: 0
Metodo: set_t Complejidad: 0
Metodo: get_t Complejidad: 3

```

Conclusiones

En conclusión, es útil medir y estimar el software porque en un determinado producto o proceso, etc., que estemos desarrollando es necesario poder conocerlo, saber cómo es y cuáles son sus características. Otro punto crucial dentro del desarrollo de Software es la realización de predicciones dentro del punto de vista de la planificación. A la hora de realizar una planificación estamos realizando una predicción de la duración, del esfuerzo, del tamaño, entre otras cosas, de lo que será el proceso de desarrollo. Para poder realizar estas predicciones de forma objetiva y precisa, necesitamos tener un conocimiento de cómo es nuestro proceso (definición), cómo se comporta (seguimiento y control) para poder así establecer con seguridad los patrones que seguirá. Por lo tanto, la medición nos permite predecir el comportamiento y evolución del desarrollo.

Además, la relación entre la calidad del producto con el proceso de desarrollo de un sistema es que, entre más tiempo pase uno desarrollando el producto se espera una mejor calidad de este mismo, por ejemplo; no es el mismo tiempo un sistema que imprime "Hola Mundo" a un sistema de mensajería, al igual que la calidad varía.

Una de las métricas más importantes es cumplir con los requisitos que se acordaron con el cliente, como son: la funcionalidad, el diseño, la optimización, entre otros acuerdos tomados. Pero la métrica más importante es la funcionalidad, ya que podemos tener el mejor diseño del mundo, pero si el software del cliente no funciona puede decir fácilmente "Sabes, no lo quiero porque no cumple con lo que acordamos", esto significa una pérdida de tiempo y recursos, también toda nuestra credibilidad como desarrolladores decae.