Tarea Programada #3

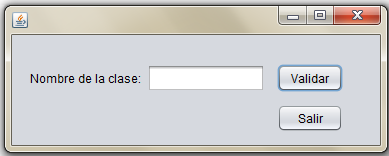
IC-2101 Programación Orientada a Objetos, I Semestre de 2013

Profesor: Luis Javier Chavarría Sánchez

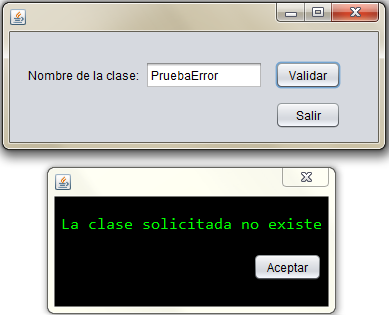
Estudiantes: Jasson González Torrez 201116046, Daniel Sánchez Castillo 201123523

a. Manual de Usuario y Pruebas de funcionalidad

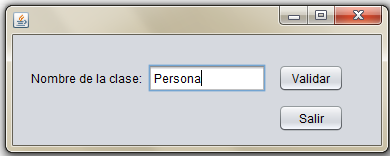
Al iniciar el programa, se le solicitará el nombre de la clase a la que desea realizar la introspección:

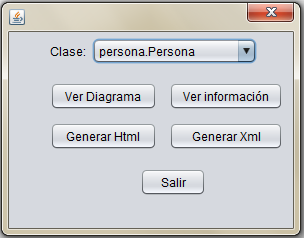


Luego de ingresar el nombre de la clase, se deberá presionar el botón Validar. En caso de que el nombre de la clase ingresada no se encuentre entre los paquetes del proyecto se mostrará el siguiente mensaje de error:



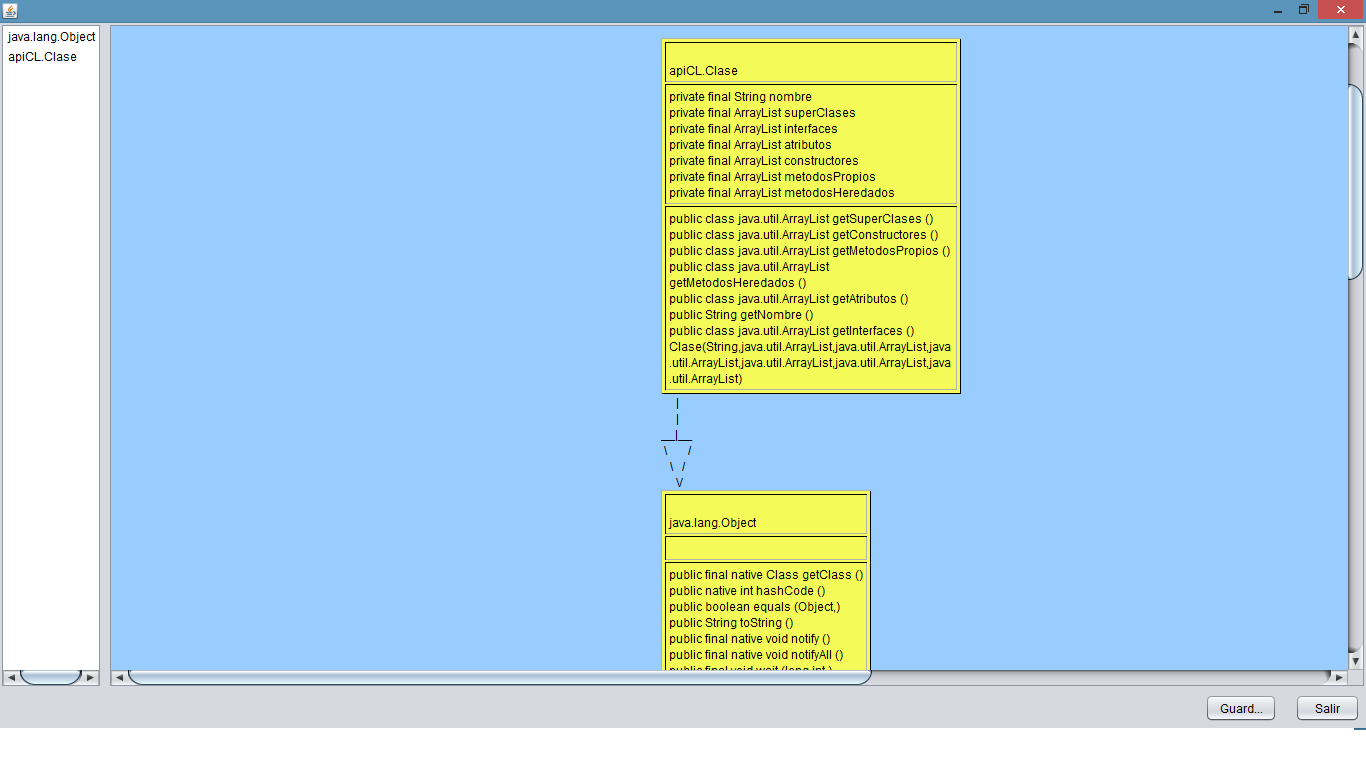
Para una clase que si exista el programa avanzará al siguiente menú:



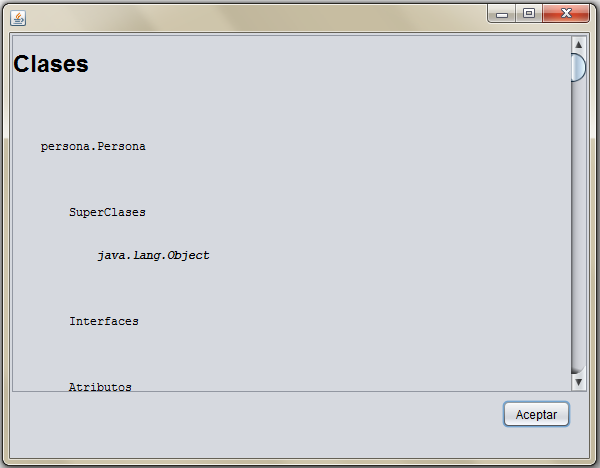


Descripción de las funciones:

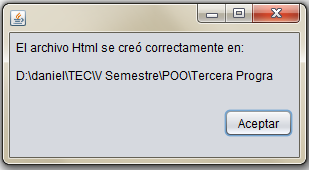
* **Ver Diagrama**: muestra el diagrama UML de la jerarquía de clases iniciando en la clase Object hasta llegar a la clase que se realiza la introspección.



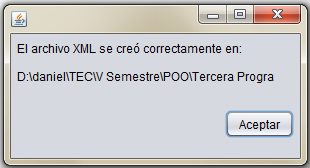
* **Ver Información**: muestra la información obtenida con la introspección de la clase indicada.



* **Generar HTML**: Genera un archivo html con el nombre de la clase e indica la carpeta en que es creado. Un mensaje similar a este es mostrado luego de la creación (cambiará la dirección según la dirección de la carpeta del proyecto) :

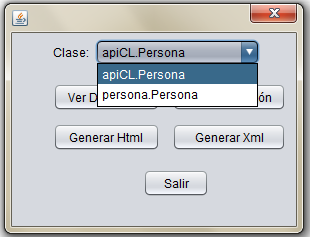


* **Generar XML**: Genera un archivo html con el nombre de la clase e indica la carpeta en que es creado. Un mensaje similar a este es mostrado luego de la creación (cambiará la dirección según la dirección de la carpeta del proyecto) :



* Salir: Regresa al menú inicial, desde el cual se podrá salir del programa o realizar la introspección nuevamente, ya sea de la misma clase o de otra.

**NOTA importante:** En caso de que el nombre de clase indicado en la ventana inicial se encuentre en más de un paquete del proyecto, se mostrarán todas las clases encontradas en la parte superior del menú de funciones:



De esta forma el usuario puede seleccionar entre las clases encontradas a la que desea realizar la introspección.

b. Diagrama de clase

Para ver correctamente el diagrama, se adjunta este en con el nombre diagrama.jpg en la misma carpeta donde se encuentra este documento.

d. Lecciones aprendidas

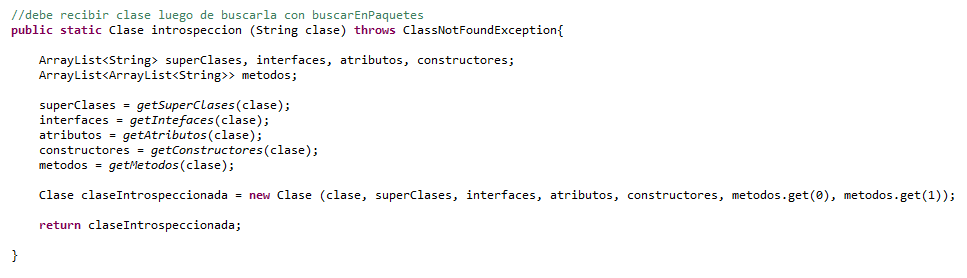
Gracias a este proyecto se obtuvo grandes conocimientos en la escritura de archivos .xml desde Java, además el manejo del Reflect puede llegar a ser muy útil para proyectos más avanzados en los próximos semestres de esta carrera y posiblemente para una futura vida laboral.

Los diagramas UML siguen siendo una parte importante del curso y luego de tres proyectos se puede asegurar que el conocimiento adquirido ha sido grande. Aun así más práctica será requerida para poder trabajar correctamente y dividir cada trabajo entre diagrama y código adecuadamente.

Debido a que el motor principal del programa se maneja dentro de dos clases (Introspeccion que obtiene la información y Clase que la almacena) los pantallazos utilizados podrían servir para más de un principio u olor.

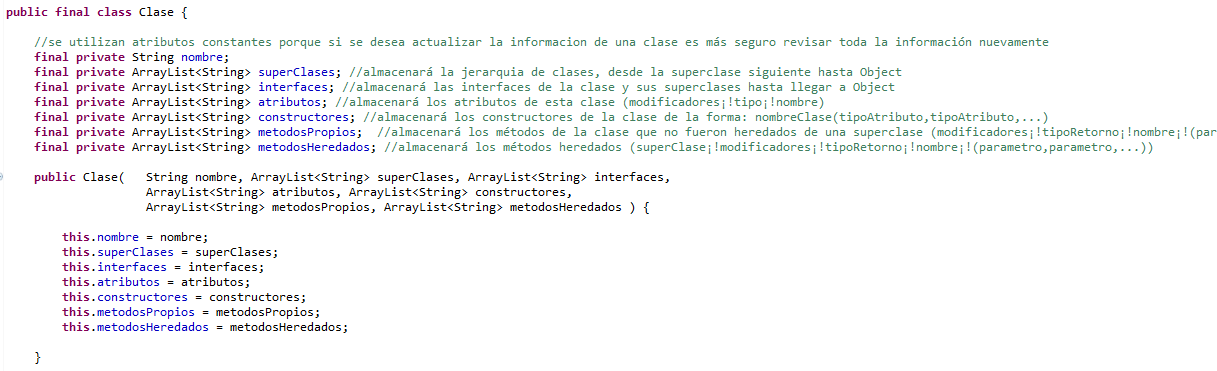
e. Principios

Carácter Personal y Estética



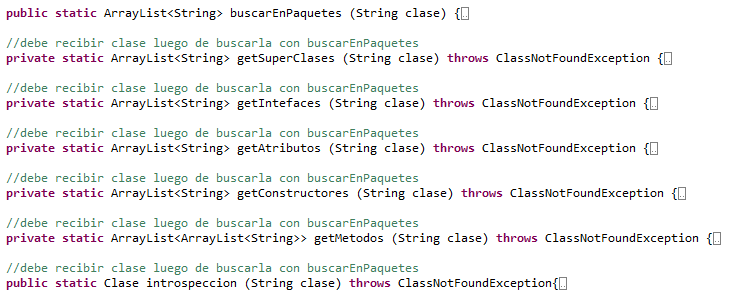
Aun cuando el código este lleno de métodos complejos que realizan trabajos más pesados, es importante dividir adecuadamente el trabajo y además tener un punto desde el cual se tenga control de lo que se necesita hacer y de las herramientas que se posee para hacerlo. El método introspección (incluido en la clase del mismo nombre) llama los otros métodos de esta clase y con ellos obtiene la información necesaria para crear un objeto de tipo Clase que tenga toda lo requerido.

Claridad, Código Auto Documentado y Comentarios, Explicito sobre Implícito



Debido a que esta clase sería la que guardaría toda la información obtenida por Introspección se utilizan listas con nombres claros, que indiquen su contenido, y además comentarios para facilitar el entendimiento de la información y de su acomodo dentro de estas listas. Además se utilizan nombres específicos como metodosPropios y metodosHeredados para favorecer lo explícito, se da a entender, no solamente que incluyen métodos de la clase, sino también de donde obtiene la clase estos métodos.

Distribución



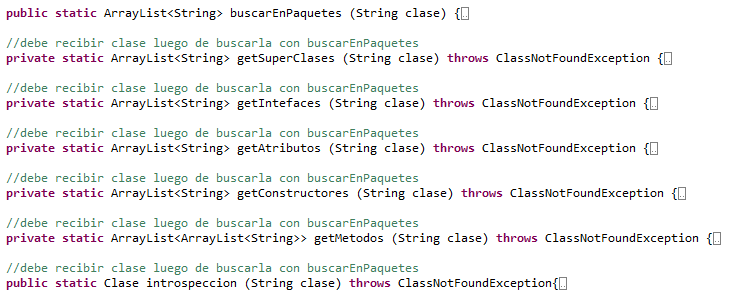
Ante la necesidad de obtener mucha información de la clase indicada se distribuye el trabajo en varios métodos, uno para cada división importante de datos. Con esto se entiende más fácilmente en que método se obtiene cada parte y en caso de ser necesario cambiar el formato o incluir más información se puede cambiar solamente el método que obtiene la misma.

El TIEMPO y EL DINERO de otros

Para las últimas dos semanas del semestre el manejo tiempo de un estudiante de Ing. en Computación es complicado, con proyectos finales, exámenes finales, etc. se debe encontrar una forma de realizar todo adecuadamente. Gracias a trabajar con el mismo equipo todo el semestre, la división no quita tiempo a la realización de diagramas y código. Siendo este el tercer proyecto en un corto lapso de tiempo, el lenguaje es conocido y se puede dedicar la mayoría del tiempo a las funcionalidades más específicas (como la creación de archivos .xml)

e. Olores del software

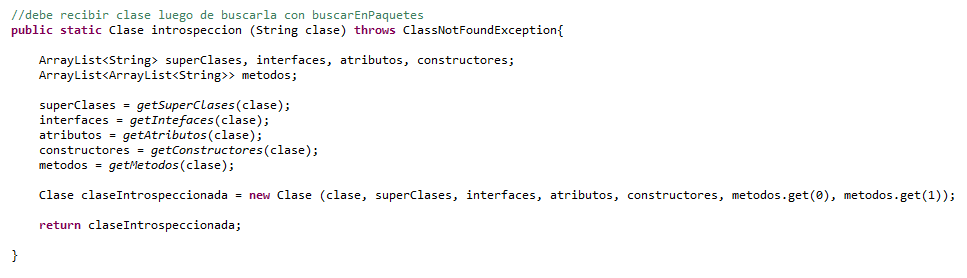
Rigidez, Viscosidad e Inmovilidad



Con la distribución utilizada para la introspección se disminuye la rigidez, ya que si hay un error en la información obtenida se puede llegar fácilmente al método que obtiene la misma y a partir de ahí trabajar para reparar el problema. Además si se desea en otro momento obtener solamente las súper clases de una clase, se puede fácilmente copiar este método y adaptarlo al nuevo programa, disminuyendo la inmovilidad.

La viscosidad se evitó dividiendo el trabajo, en caso de que un cambio fuera necesario para obtener una parte de la información específica no será necesario cambiar el diseño de la clase, solamente se deberá cambiar el código dentro del método que se desea modificar.

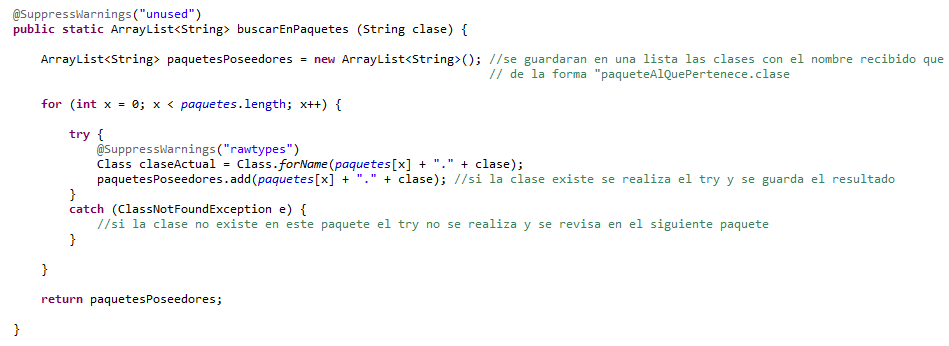
Fragilidad, Complejidad innecesaria



Se maneja un método controlador/creador para evitar tener conexiones que aumenten la fragilidad. En lugar de usar cada método de Introspeccion en la clase Clase para obtener la información, se genera un solo método que obtenga toda la información y cree el objeto.

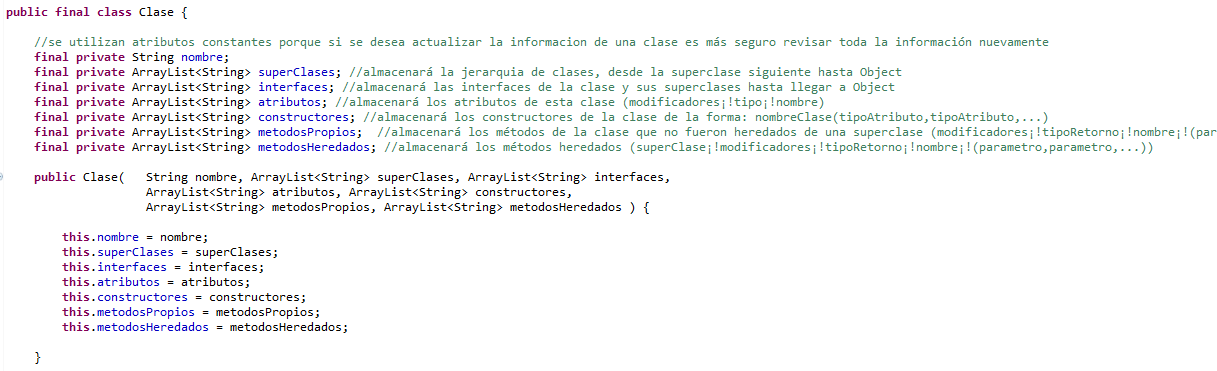
La complejidad se disminuye con este método ya que desde otras partes del programa no es necesario saber cómo se obtiene la información, ni cuales métodos obtienen cada parte; si la interfaz desea acceder a la información de una clase solamente debe llamar a este método y almacenar el objeto que retorna.

Repetición innecesaria



Para evitar que cada método de introspección tuviera que revisar si la clase que recibe existe se genera un método que revisa las clases en los paquetes del proyecto y retorna las clases existentes con el nombre indicado. De esta forma se hace solamente 1 vez antes de crear los objetos y no cada vez que se desea obtener una pieza de información

Opacidad



Se trata de disminuir la opacidad y aumentar la claridad con comentarios en partes donde se puede complicar la lectura o el entendimiento de información. Además se trabaja con varias líneas de creación de listas (en lugar de solo una) para poder explicar el uso que se dará a cada atributo.