Instituto Tecnológico de Costa Rica

Ingeniería en Computación

Sede Central Cartago

Programación Orientada a Objetos – Grupo 01

Prof. Ericka Marín Shuman

II Semestre, 2019

Estudiantes:

Joseph Tenorio Pereira (2019064588)

Carlos Varela Ramírez (2019077177)

Fabrizio Ferreto Saborío (2019177147)

Tarea programada #2: Introspección

Entrega: 30 de octubre de 2019

***Índice***

Portada………………………………………………………………………..……………1

Índice……………………………………………………………………………………..…2

Manual del usuario……………………………………………..…………………………3

Bitácora…………………………………………………………………………..…………6

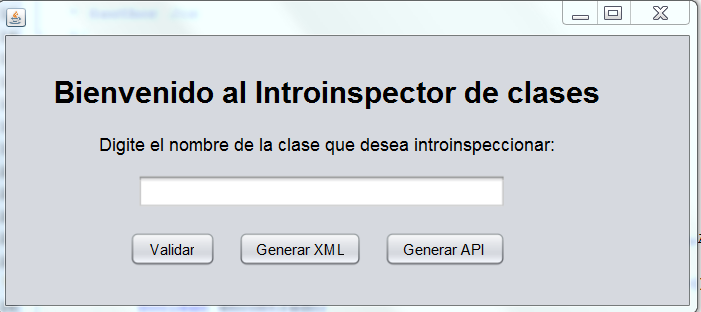
Diagramado de clases……………………………………….……………………………7

Análisis de resultados…………………………………………….………………………8

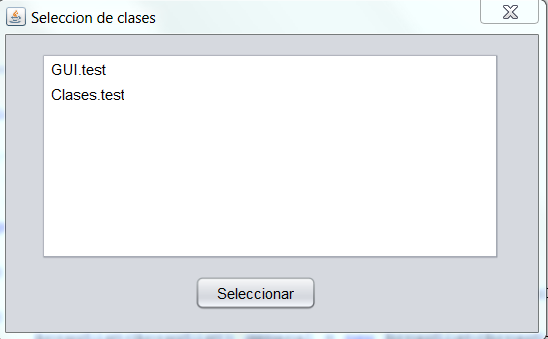
Lecciones aprendidas ……………………………………………………………………9

***Manual del usuario***

El programa adjuntado a este documento tiene como principal objetivo el análisis introspectivo de clases incluidas en su propio código fuente. Como parte de dicho análisis, el programa mostrara cierta información sobre la clase dada, tal como listado de superclases, interfaces implementadas, métodos, constructores, atributos, entre otros. Adicionalmente, el programa brinda la posibilidad de generar un documento en formato XML con la información anteriormente mencionada.

Primeramente, al momento de ejecutar el programa se muestra el menú principal, el cual tiene el siguiente aspecto:

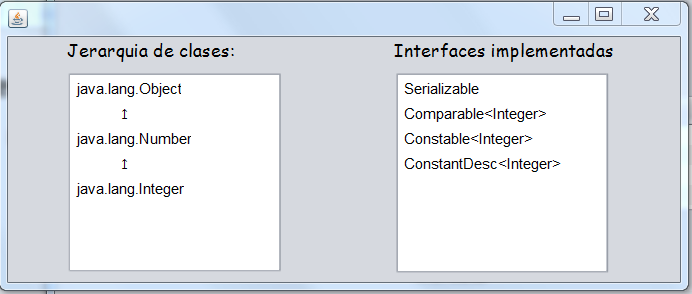
-***Validación de clase:*** Antes de proceder con las funciones de generar API y generar XML, es necesario validar una clase sobre la cual realizar la introspección. Para ello se debe ingresar el nombre de la clase en el espacio indicado y presionar el botón accionar. Si el programa encuentra una única clase con el nombre especificado, se mostrara un mensaje de confirmación. Si no se encontrara ninguna clase cuyo nombre coincidiera con lo ingresado, aparecerá un mensaje que así lo indica. En caso de encontrar múltiples clases con el nombre especificado, deberá seleccionar una de ellas por medio de una ventana de selección como la siguiente:



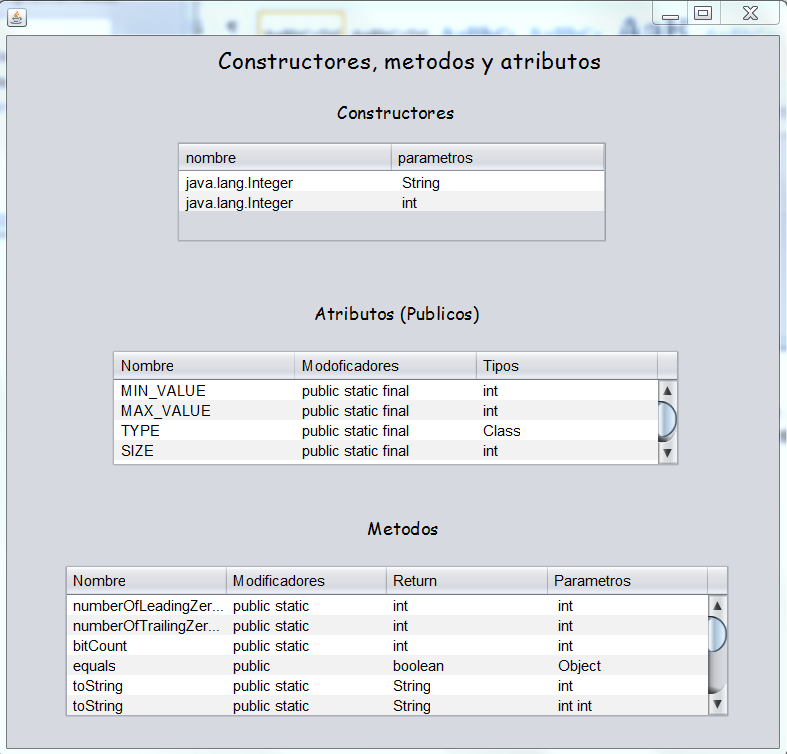
Una vez seleccionada una opción, la clase indicada que lista para el análisis introspectivo.

- ***Generación de API:*** Como ya mencionó, es necesario validar una clase antes de acceder a esta opción, ya que se trata del análisis de la clase propiamente dicho. La información obtenida se muestra resumidamente en dos ventanas.

En la primera de ellas, se muestra la sucesión de superclases heredadas, partiendo de la clase indicada y acabando en clase “Object”. Adicionalmente, esta ventana muestra el listado de interfaces implementadas por la clase en sí y su respectivo linaje de superclases. Por ejemplo, el análisis de la clase “Integer” arroja los siguientes resultados.



En la otra ventana generada se muestra la información correspondiente a los constructores, atributos y métodos de la clase. Siguiendo con el ejemplo anterior, el análisis de la clase “Integer” muestra la siguiente información.



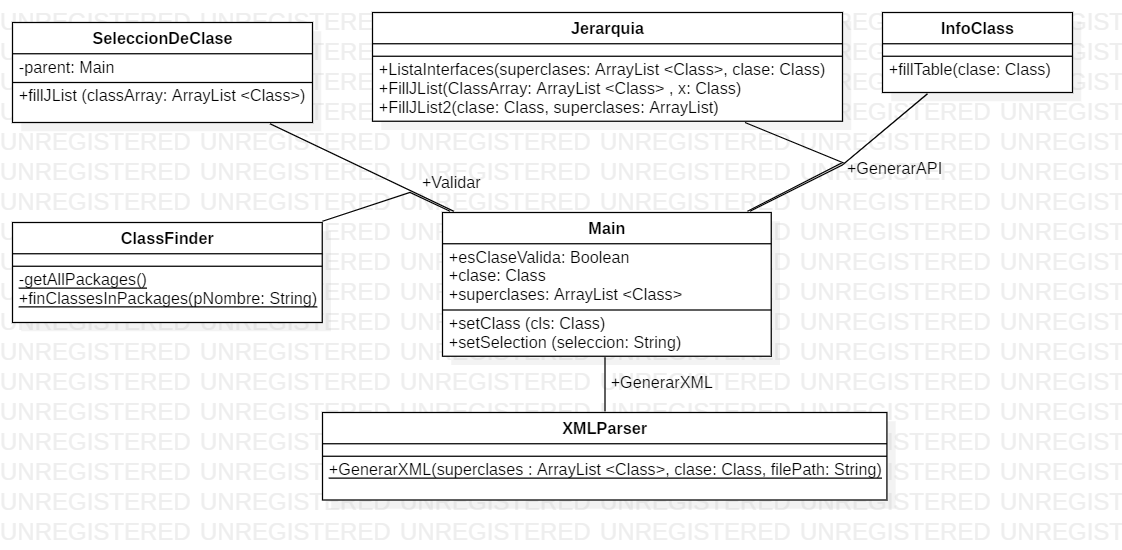
- ***Generar XML:*** Esta opción permite organizar la información de la API en el formato de un archivo XML, el cual es creado en la carpeta del mismo nombre ubicada junto a los archivos del programa. Al igual que con el punto anterior, es necesario validar una clase antes de generar un nuevo archivo XML.

***Bitácora de trabajo***

Dado que el presente proyecto se desarrolló en su mayoría por medio de un repositorio de la plataforma GitHub, se presentan a continuación todos los commits realizados sobre el mismo, los cuales reflejan el progreso en la creación del programa.

* 22/10/2019: Joseph Tenorio: Se crea la ventana principal del proyecto y se adjunta el modelado UML previamente realizado en grupo.
* 23/10/19: Joseph Tenorio: Se añade la funcionalidad de validar clases cuyo nombre no se repita en los paquetes del programa.
* 23/10/19: Joseph Tenorio: Se añade la funcionalidad de validar clases cuyo nombre no se repita en los paquetes del programa.
* 24/10/19: Joseph Tenorio: Se finaliza la validación de clase, incluyendo clases con el mismo nombre.
* 25/10/19: Joseph Tenorio: Se cargan los objetos correspondientes a la jerarquía de superclases a partir de la clase dada, y se crea la ventana de muestra de dicha información.
* 27/10/19: Carlos Varela: Se implementa un primer prototipo del mecanismo para la creación de archivos XML
* 28/10/19: Fabrizio Ferreto: Se añade la segunda ventana del API, con su respectiva muestra de los resultados (constructores, métodos y atributos) del último análisis de clase.
* 28/10/19: Carlos Varela: Se termina de implementar la creación de los archivos XML y se añade la muestra de las interfaces implementadas a la API
* 29/10/19: Carlos Varela y Joseph Tenorio: Se realizan las pruebas finales del programa, se corrigen errores varios y se valida el excluir a las interfaces del análisis de clases.

***Diagramado de clases***

******

***Análisis de resultados***

A continuación se listan las funcionalidades que en un principio se esperaban del programa, y su respectivo porcentaje de completitud alcanzado al final de proyecto.

1) Reconocer cuando una clase se ubica dentro del ámbito de ejecución: **100%**

2) Reconocer cuando hay más de una clase con el mismo nombre y permitir la selección de una de estas: **100%**

3) Determinar y mostrar la jerarquía de clases presente hasta la clase “Object”: **100%**

4) Determinar y mostrar las interfaces implementadas por la clase analizada y su línea jerárquica de superclases: **100%**

5) Determinar y mostrar los atributos de la clase: **100%**

6) Determinar y mostrar los constructores de la clase: **100%**

7) Determinar y mostrar los métodos de la clase: **100%**

8) Creación del archivo en formato XML con toda la información de la API: **100%**

***Aspectos relevantes y lecciones aprendidas***

Por último, en esta sección se presenta un listado de conocimientos adquiridos durante el desarrollo del programa:

- Se debió adquirir conocimiento técnico sobre la implementación de las clases Class, Object y Package propias de las librerías estándares de Java. Lo anterior se debe a que estas clases debían ser empleadas en la funcionalidad de validación de clases.

- Fue necesario adquirir conocimientos sobre diversos componentes asociados a la compilación de un programa escrito en Java, tales como los paquetes, el classloader, archivos de clase de Java, etc. Los anteriores conceptos fueron aclarados y profundizados en su definición y función.

- En cuanto a la creación del archivo en formato XML, para ello fue necesario implementar las librerías File, Transformer, DocumentBuilder, DOM y StreamResult, por lo que se adquirió un conocimiento básico sobre su uso.

- Se debieron implementar diversas herramientas de introspección para desarrollar las funcionalidades del programa, por lo que se profundizó en el uso de varios métodos introspectivos encontrados en la librería reflections, por ejemplo.

- Respecto al trabajo en equipo, se puso en práctica el uso de un repositorio de GitHub como forma de compartir el trabajo realizado entre distintos miembros. Dado que se realizó una distribución de labores en la cual las funcionalidades asignadas a cada miembro dependían entre sí, se desarrolló la práctica de mantener una buena comunicación sobre los cambios de implementación realizados por cada integrante.