Diseño e implementación de Applets como material didáctico de apoyo para cursos de estructuras de datos

Betzabet García-Mendoza

Universidad Autónoma Metropolitana, Unidad Cuajimalpa,

Av. Vasco de Quiroga 4871, Col. Santa Fe Cuajimalpa, C.P. 05348, México, D.F., Teléfono: 5814-6500

209363465@alumnos.cua.uam.mx

Pablo A. Ruíz-Mendoza

Universidad Autónoma Metropolitana, Unidad Cuajimalpa,

Av. Vasco de Quiroga 4871, Col. Santa Fe Cuajimalpa, C.P. 05348, México, D.F., Teléfono: 5814-6500

209363000@alumnos.cua.uam.mx

Gerardo Real-Flores

Universidad Autónoma Metropolitana, Unidad Cuajimalpa,

Av. Vasco de Quiroga 4871, Col. Santa Fe Cuajimalpa, C.P. 05348, México, D.F., Teléfono: 5814-6500

209363111@alumnos.cua.uam.mx

Carlos R. Jaimez-González

Universidad Autónoma Metropolitana, Unidad Cuajimalpa,

Av. Vasco de Quiroga 4871, Col. Santa Fe Cuajimalpa, C.P. 05348, México, D.F., Teléfono:5814-6500

cjaimez@correo.cua.uam.mx

Esaú Villatoro-Tello

Universidad Autónoma Metropolitana, Unidad Cuajimalpa,

Av. Vasco de Quiroga 4871, Col. Santa Fe Cuajimalpa, C.P. 05348, México, D.F., Teléfono: 5814-6500

evillatoro@correo.cua.uam.mx

Resumen

En este artículo se presentan tres *applets* desarrollados en el lenguaje de programación Java, los cuales se utilizan como material didáctico para apoyar los cursos de estructuras de datos de licenciatura. Los *applets* demuestran de una manera dinámica, atractiva, y con ejemplos de la vida cotidiana, el funcionamiento de los tipos datos abstractos pila, cola y cola circular, en los cuales el estudiante interactúa mediante la creación de instancias de cada uno de los tipos de datos abstractos, y ejecutando el conjunto de operaciones definido para cada uno.

Palabras Clave: applets, estructuras de datos, lenguaje Java, material didáctico, tipo de dato abstracto.

1. Introducción

En los cursos de programación se emplea de manera regular el concepto de tipos de datos simples, mismos que se refieren a valores de tipo simple, como los números enteros, reales o caracteres. Sin embargo, en muchas situaciones se necesita procesar colecciones de valores que están relacionados entre sí, tales como listas de calificaciones, series de temperaturas medidas a lo largo de un periodo de tiempo, etc. El procesamiento de tales conjuntos de datos utilizando tipos de datos simples puede ser complicado, es por ello que la mayoría de los lenguajes de programación incluyen estructuras de datos, tales como los arreglos, los cuales pueden ser utilizados para representar vectores y matrices.

Este tipo de estructuras básicas se denominan también estructuras estáticas, debido a que las variables son direcciones simbólicas de posiciones de memoria, mismas que representan una relación estática que se establece por la declaración de las variables de una unidad de programa y que además se concibe durante la ejecución de esa unidad. Para muchos problemas de la vida real es difícil conocer cuánta memoria sería requerida para resolverlos o representarlos, por lo que es necesario contar con métodos que permitan adquirir posiciones adicionales de memoria cuando éstas sean requeridas

durante la ejecución del programa y, de igual forma liberarlas cuando éstas ya se hayan desocupado. A estas posiciones adicionales de memoria que se crean y están disponibles durante la ejecución de un programa se les conoce como variables dinámicas, las cuales son representadas con un tipo de dato conocido como apuntador (o referencia en el lenguaje de programación Java). Las variables dinámicas se utilizan para crear estructuras de datos dinámicas que se pueden ampliar y reducir a medida que se requiera durante la ejecución de un programa. Las estructuras de datos dinámicas se clasifican en lineales y no lineales.

En este artículo se describe la metodología de desarrollo que se siguió para el diseño e implementación de un conjunto de *applets* que son utilizados como material didáctico de apoyo para la enseñanza de algunos temas del curso de Estructuras de Datos de la Licenciatura en Tecnologías y Sistemas de Información de la Universidad Autónoma Metropolitana, Unidad Cuajimalpa. Los *applets* desarrollados específicamente apoyan la enseñanza de estructuras de datos no lineales, también denominadas como tipos de datos abstractos (TDA), los cuales son: pilas, colas y colas circulares.

Un TDA es un modelo compuesto por un conjunto de operaciones definidas sobre un conjunto de datos [1], [2]. Una pila (*stack*) es un TDA que almacena elementos que solamente pueden ser añadidos o extraídos por su parte superior; una pila es considerada una estructura de tipo *LIFO* (*Last In, First Out*), ya que el último elemento en entrar a la pila es el primero en salir, tal como ocurre en una pila de platos o en una pila de libros, en donde el último plato o libro colocado es el primero que se extrae. Una cola (*queue*) es también un TDA que almacena elementos y permite acceder a ellos por uno de sus extremos; un elemento se inserta en la cola por su parte final y se extrae por su parte frontal; una cola es considerada una estructura de tipo *FIFO* (*First In, First Out*), ya que el primer elemento en entrar a la cola es el primero en salir, tal como sucede en una ventanilla de atención a clientes en una tienda o supermercado, en donde el primer cliente que entra es el primero que se atiende. Una cola circular es también un TDA similar a una cola, pero en esta estructura se permite que la totalidad

de sus posiciones se utilicen para almacenar elementos sin necesidad de desplazar elementos, de modo que se una el extremo final con el extremo frontal de la cola.

El resto del artículo está organizado de la siguiente manera. La sección 2 presenta una revisión del estado del arte. En la sección 3 se describe el levantamiento de requerimientos que se realizó. La identificación y redacción de casos de uso se presentan en la sección 4. En la sección 5 se describen algunos otros diagramas y artefactos que fueron desarrollados en este trabajo, tales como el diagrama del modelo de dominio y los diagramas de secuencia. La implementación de los *applets* se presenta en la sección 6, donde se muestra la interfaz y funcionamiento de cada uno. Los controles de prueba se abordan en la sección 7. Finalmente, en la sección 8 se proporcionan las conclusiones y el trabajo futuro.

2. Estado del arte

En esta sección se presenta una descripción de algunas herramientas y aplicaciones que se analizaron como parte del estado del arte.

Tutorial interactivo de estructuras de datos

Este tutorial [3] fue realizado en la ciudad de La Paz, Bolivia, y tuvo como objetivo principal exponer los contenidos de avance de la materia de Estructura de Datos de la Universidad Católica Boliviana San Pablo en el año 2011, mostrando sus aplicaciones con los principales algoritmos, a través de *applets*. El tutorial cuenta con los siguientes temas: Estructuras estáticas, Listas Simples, Listas Circulares, Listas Dobles, Listas Dobles Circulares, Pilas y Colas, y Árboles. Cada tema se compone de una explicación textual, algunas imágenes y un *applet*.

Los temas de Pilas y Colas proporcionan una explicación de su funcionamiento y las operaciones que se pueden realizar en cada una de ellas. Los *applets* están basados en rectángulos que representan espacios de memoria, en los cuales se pueden almacenar únicamente números mediante la operación *Apilar/Encolar*, y borrar la información mediante la operación de *Desapilar/Desencolar*. También se tiene la opción

de *Buscar*, la cual permite buscar un número dentro de la pila o cola, según sea el caso; el *applet* muestra una ventana para solicitar el número a buscar y en la misma ventana muestra el resultado, encontrado o no encontrado.

Herramienta para estudio de estructuras de datos

Esta herramienta [4] para el estudio de estructuras de datos y algoritmos fue realizada por alumnos de la Universidad Complutense de Madrid; está implementada en Java y fue diseñada de manera modular. Al iniciar la aplicación se debe seleccionar un área de dos posibles: estructuras de datos o esquemas algorítmicos; posteriormente se debe elegir la estructura o esquema concreto que se desea utilizar. Para el caso de las estructuras de datos, se tienen pilas, colas, árboles binarios de búsqueda, colas de prioridad, tablas ordenadas y tablas dispersas.

La herramienta cuenta con una interfaz interactiva, en la cual se tienen dos opciones para cada una de las estructuras de datos: ver una simulación, o ejecutar operaciones sobre la estructura de datos. Existen dos vistas en la herramienta: la vista de usuario, en la cual se visualiza el comportamiento de la estructura de manera independiente de la implementación; y la vista de implementación, en la cual se muestra cómo se almacenan de manera concreta los datos de la estructura.

La representación gráfica de los elementos de las pilas y colas, son recuadros que se unen unos con otros; algunos con flecha para representar los apuntadores o referencias. Los nodos de los árboles son representados con círculos que se unen por medio de líneas, para representar las conexiones entre ellos.

Software para la enseñanza de algoritmos

Esta aplicación [5] está implementada en el lenguaje de programación Java, por medio de *applets*. En dicho trabajo se encuentran dos animaciones que presentan el algoritmo *Mergesort* y el TDA Pila. Para acceder a las animaciones se tiene un sitio web donde mediante un tutorial se presenta una descripción del funcionamiento del TDA Pila, algunas imágenes para su explicación, y finalmente la animación.

Para ilustrar la animación del TDA Pila se utilizó el ejemplo de un palíndromo, palabra o expresión que es igual si se lee de izquierda a derecha que de derecha a izquierda. La animación cuenta con cinco opciones: *Iniciar*, *Detener*, *Pausar*, *Paso a Paso* y *Continuar*. El usuario debe proporcionar la palabra a evaluar; la animación corta la palabra en letras y las coloca en una pila; en la interfaz se observan dos arreglos, uno en el que se van sacando letras y otro en el que se van agregando, para así mostrar si la palabra se lee igual de izquierda a derecha que de derecha a izquierda.

3. Levantamiento de requerimientos

Para la realización de este material didáctico se tuvieron los siguientes requisitos:

1. Por cada TDA deberá realizarse un *applet*.

2. Los *applets* deberán estar basados en situaciones de la vida real, con las que cualquier persona estaría familiarizado; y deberán ilustrar cómo operan las estructuras propuestas.

3. Los *applets* deberán permitir la interacción con los alumnos. Cada *applet* tendrá las siguientes operaciones: *Crear*, operación que permitirá al estudiante indicar el tamaño de la estructura; *Agregar*, operación que permitirá al estudiante agregar elementos al TDA; *Quitar*, operación que permitirá al estudiante quitar elementos del TDA; *Reiniciar*, operación que permitirá al estudiante indicar al *applet* que vuelva a su estado inicial, dando la oportunidad de volver a operar el TDA.

4. Se requiere además, de una vista que ejemplifique cómo se vería el TDA en memoria. Deberá existir un apuntador que indique el tope, el frente, el siguiente y último elemento según corresponda a la estructura o TDA que se esté probando.

4. Identificación y redacción de casos de uso

En esta sección se muestran los casos de uso identificados al momento de hacer el análisis de requerimientos correspondiente. Para cada uno de los casos de uso se identifica al actor principal, precondiciones, postcondiciones, el flujo básico del caso de

uso y el flujo alternativo. Por cuestiones de espacio, solamente se mostrarán algunos de los casos de uso que fueron identificados para el TDA Pila. En la Tabla 1 se presenta el caso de uso *Crear Pila;* en la Tabla 2 se muestra el caso de uso *Operar Pila;* y finalmente, en la Tabla 3 se tiene el caso de uso *Reiniciar Pila*.

Caso de uso: Crear Pila	
Actor Principal	Estudiante.
Precondición	Haber ingresado al documento HTML.
Postcondición	Poder operar la pila.
Flujo Básico	
Paso 1	El estudiante ingresa el tamaño de la pila.
Paso 2	El estudiante da clic en el botón Crear.
Paso 3	El sistema muestra gráficamente la pila del tamaño indicado por el estudiante.
Flujo Alternativo	
Paso 3a	El sistema muestra un mensaje de error, indicando el tamaño máximo que puede elegir.

Tabla 1. Caso de Uso Crear Pila.

Caso de uso: Operar Pila	
Actor Principal	Estudiante.
Precondición	Haber creado una pila.
Postcondición	Posibilidad de reiniciar el applet.
Flujo Básico	
Paso 1	El estudiante da clic en el botón *Push*.
Paso 2	El sistema inserta un nuevo elemento al final de la pila.
Paso 3	El estudiante da clic en el botón *Pop*.
Paso 4	El sistema extrae el último elemento de la pila.
Paso 5	El sistema actualiza el *TOP* después de cada operación.
Flujo Alternativo	
Paso 2a	El sistema bloquea el botón *Push* dado que la pila está llena.
Paso 4a	El sistema bloquea el botón *Pop* dado que la pila está vacía.

Tabla 2. Caso de Uso: Operar Pila.

Caso de uso: Reiniciar Pila	
Actor Principal	Estudiante.
Precondición	Haber creado u operado pila.
Postcondición	Volver a operar pila.
Flujo Básico	
Paso 1	El estudiante da clic en el botón *Reiniciar*.
Paso 2	El sistema reinicia el applet.

Tabla 3. Caso de Uso: Reiniciar Pila.

Diagrama de casos de uso

En la Figura 1 se muestra de manera resumida los casos de uso descritos previamente, mediante un diagrama, en el cual se observa que el único actor es el estudiante, quien puede interactuar con los diferentes TDA en el sistema.

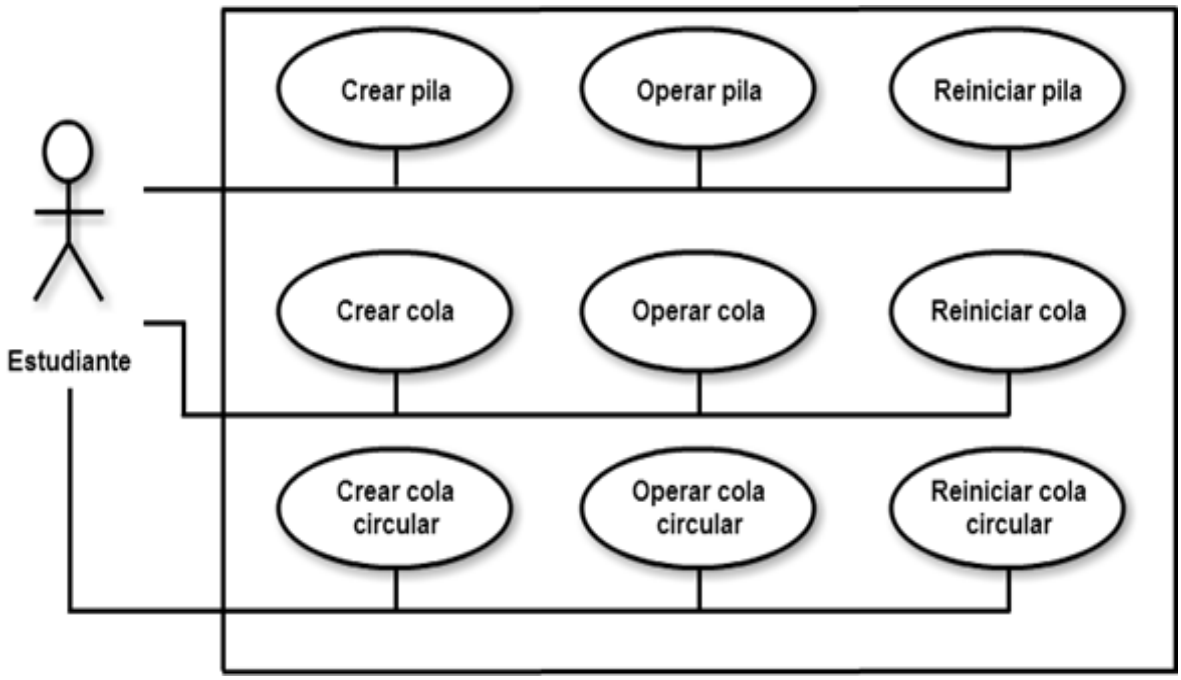

Fig. 1. Diagrama de Casos de Uso.

5. Diagramas y artefactos desarrollados

En esta sección se muestra de manera detallada algunos de los artefactos construidos durante el desarrollo de este material didáctico. Es conveniente recordar que la metodología del proceso unificado no es una metodología estricta en cuanto a la cantidad de documentación que se debe de generar durante el desarrollo de una aplicación [6]. Por el contrario, el proceso unificado sugiere utilizar sólo los documentos necesarios. Para ente proyecto se realizó la siguiente documentación y artefactos: casos de uso, modelo del dominio, diagramas de secuencia, diagrama de clases, reportes de pruebas.

En la Figura 2 se muestra el Modelo de Dominio, donde se observan los tres TDA (pila, cola y cola circular), para los cuales se desarrollaron los *applets*.

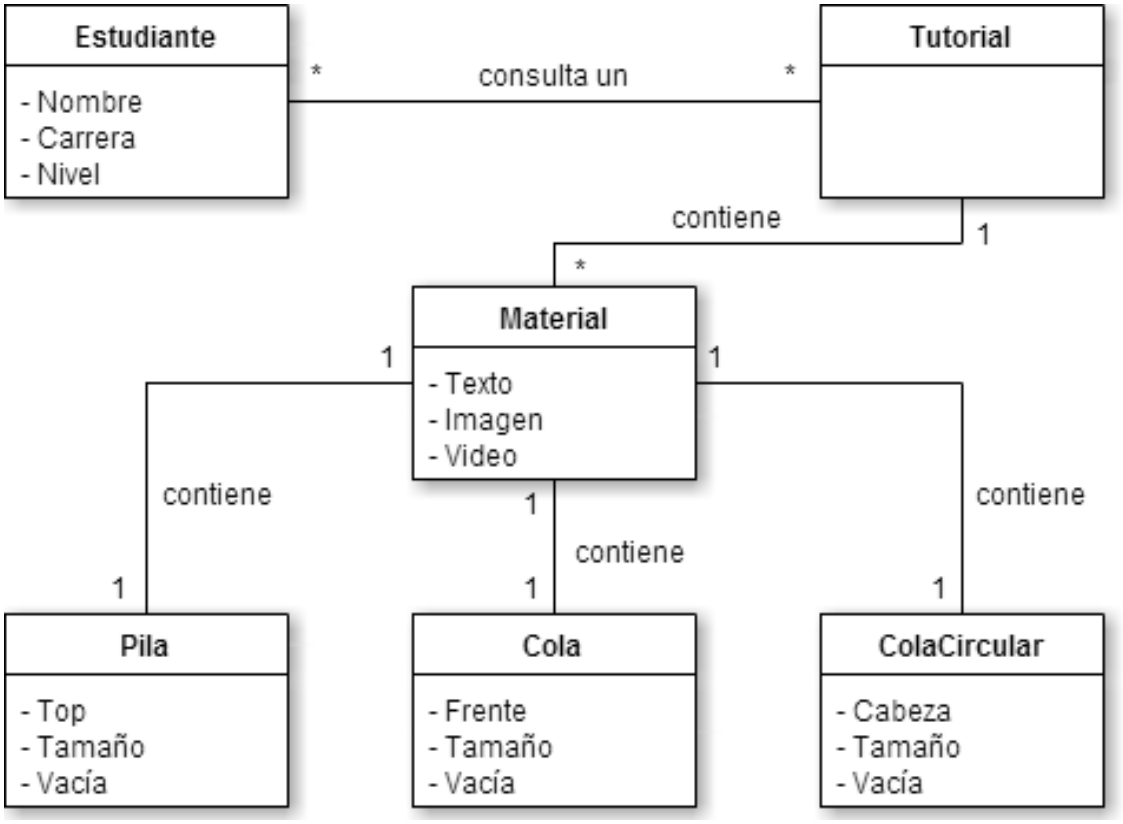

Fig. 2. Modelo de dominio.

Diagramas de secuencia

En esta sección se muestran los diagramas de secuencia, los cuales representan la interacción que existe entre los objetos o componentes del sistema; en este caso, muestran la ejecución de métodos entre los objetos que componen los *applets*. Cabe señalar, que solo se muestran algunos de los diagramas de secuencia del TDA Pila.

En la Figura 3 se muestra el diagrama de secuencia para crear una pila; en la Figura 4 se observa el diagrama de secuencia para insertar un elemento en una pila; en la Figura 5 se ilustra el diagrama de secuencia para extraer un elemento de una pila.

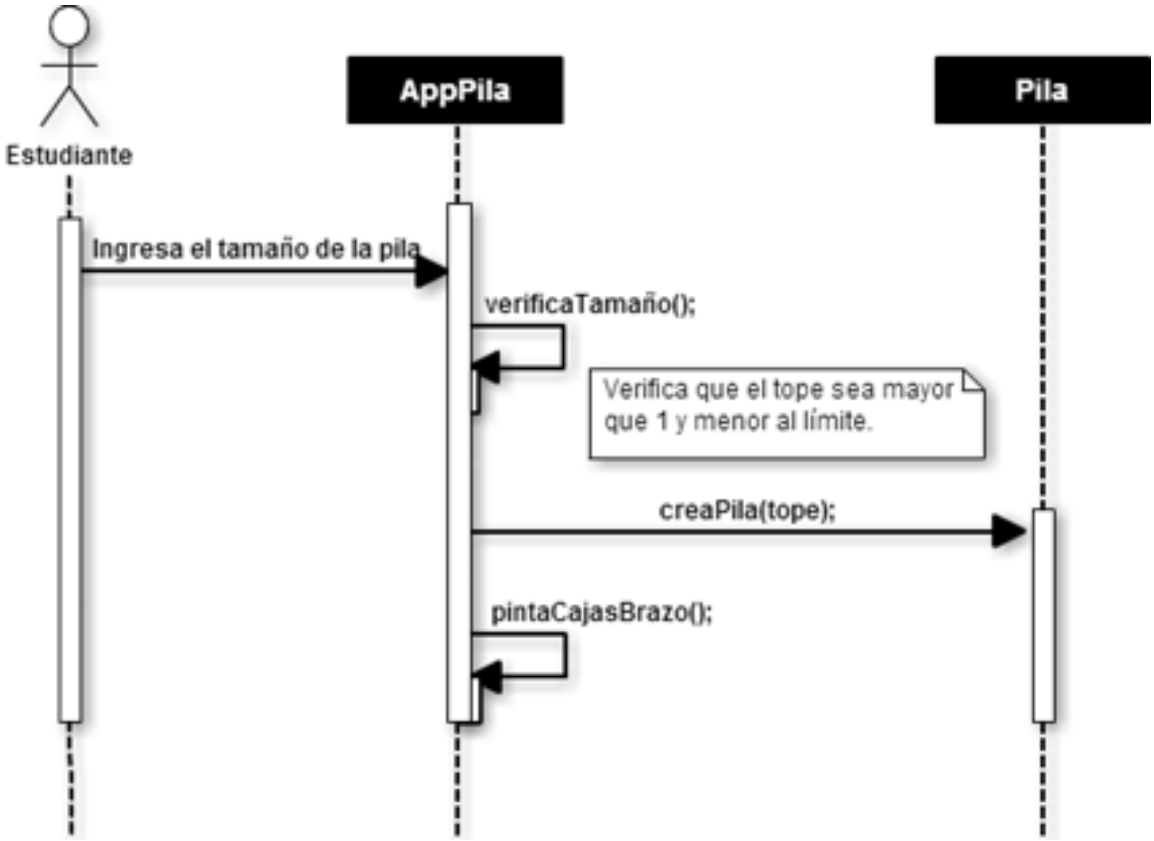

Fig. 3. Diagrama de secuencia para crear una pila.

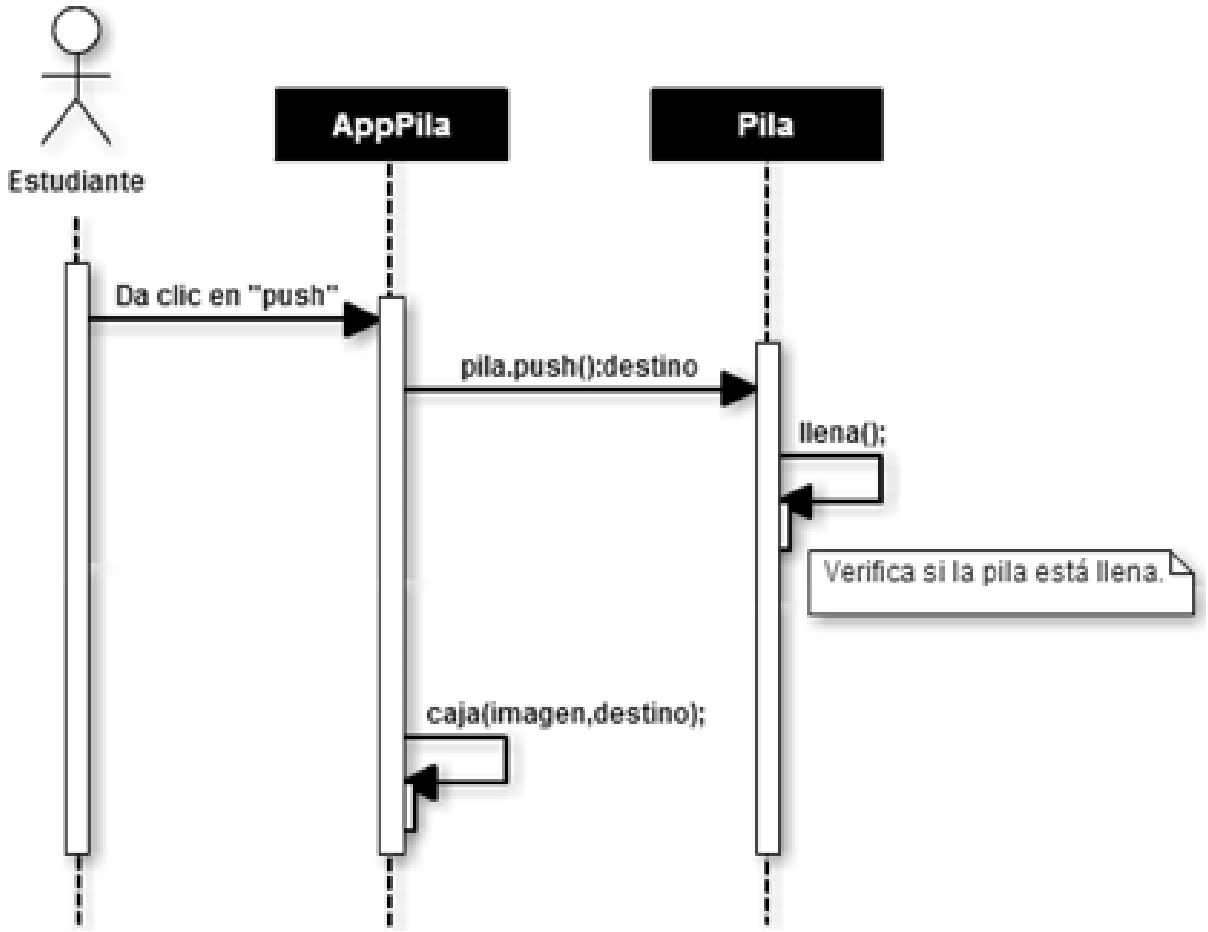

Fig. 4. Diagrama de secuencia para insertar un elemento en una pila.

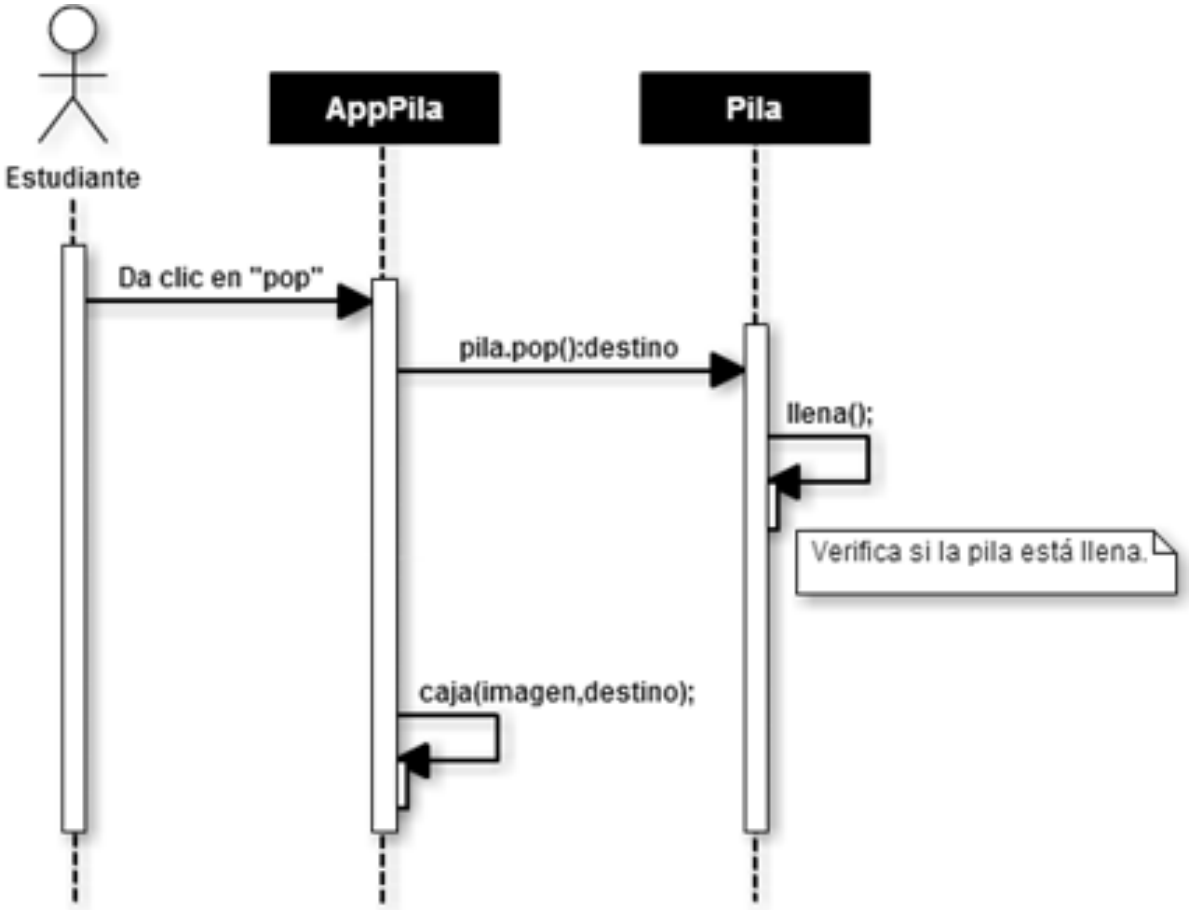

Fig. 5. Diagrama de secuencia para extraer un elemento de una pila.

6. Implementación de los applets

En esta sección se presenta la implementación de los tres *applets* descritos previamente: *applet* del TDA cola, *applet* del TDA cola circular, y *applet* del TDA pila. En las subsecciones siguientes se describen los escenarios utilizados para ilustrar cada uno de los TDA, así como su funcionamiento.

Applet del TDA cola

Para la implementación de este *applet* se diseñaron las imágenes mostradas en la Figura 6, las cuales representan lo siguiente: a) la caja de regalo vacía representa un espacio de memoria vacío en la cola; b) la caja de regalo llena representa un espacio de memoria lleno en la cola, es decir, un elemento que ha sido *encolado*; c) la caja de regalo abierta representa un espacio de memoria que ha sido desocupado en la cola, es decir, un elemento que ha sido *desencolado*; d) la persona caminando representa la ejecución de la operación que *encolará* o *desencolará* los elementos en la cola. Estas imágenes son utilizadas en conjunto con un escenario que también fue diseñado para ilustrar las operaciones de *encolamiento* y *desencolamiento* de elementos.

Fig. 6. Imágenes utilizadas en el applet del TDA cola.

La Figura 7 muestra el *applet* del TDA cola en ejecución, donde se observa una cola que ya ha sido creada con cuatro espacios en memoria, representados por las cajas de regalo, las cuales se encuentran inicialmente vacías cuando la cola ha sido creada. La mecánica de la animación de este *applet* se describe a continuación. El usuario del *applet* del TDA cola puede realizar las operaciones de *encolar* y *desencolar*, a través de los botones que se encuentran en la parte inferior.

La operación *encolar* pone en acción a la persona, la cual camina hasta el primer espacio de memoria disponible de la cola y *encola* el elemento, lo cual es representado

por el cambio de imagen de la caja de regalo vacía a la caja de regalo llena, simulando que la persona ha depositado un regalo en la caja. En este caso la operación *encolar* puede ser ejecutada únicamente cuatro veces, ya que en este ejemplo existen cuatro espacios de memoria disponibles. Cabe señalar que en una cola no es posible *encolar* elementos en espacios de memoria que ya han sido previamente ocupados (elementos que han sido *desencolados*).

La operación *desencolar* pone en acción a la persona, la cual camina hasta el primer espacio de memoria lleno de la cola y *desencola* ese elemento, lo cual es representado por el cambio de imagen de la caja de regalo llena a la caja de regalo abierta, simulando que la persona ha sacado el regalo de la caja.

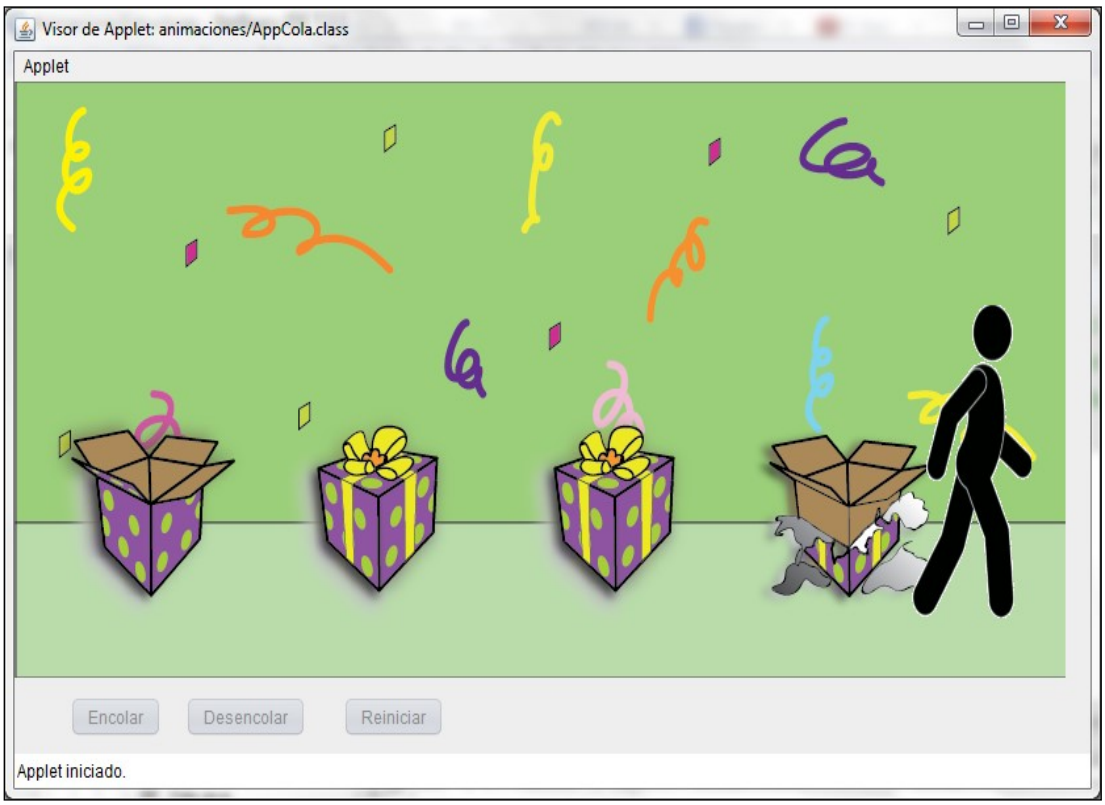

Fig. 7. Applet del TDA cola en ejecución.

En el caso específico de la Figura 7, el usuario *encoló* tres elementos en la cola, y *desencoló* un elemento, como se puede observar por las imágenes de las cajas de

regalo: el primer espacio de memoria ha sido *desencolado* (caja de regalo abierta), los dos siguientes espacios de memoria tienen elementos *encolados* (cajas de regalo llenas), y el último espacio de memoria está aún vacío (caja de regalo vacía). La animación termina cuando el usuario ha *encolado* y *desencolado* los cuatro elementos posibles en este ejemplo; o cuando se presiona el botón de reiniciar, con el cual se reinicia el *applet* y se puede crear una nueva cola con un número determinado de espacios de memoria (cajas de regalo vacías).

Applet del TDA cola circular

Para la implementación de este *applet* se diseñaron las imágenes mostradas en la Figura 8, las cuales representan lo siguiente: a) la silla vacía representa un espacio de memoria vacío en la cola circular; b) la persona sentada en la silla representa un espacio de memoria lleno en la cola circular, es decir, un elemento que ha sido *encolado*; c) la persona caminando representa la ejecución de la operación *encolar* si la persona se sentará en alguna silla, o *desencolar* si la persona se ha levantado de una silla en la que estaba sentada. Estas imágenes son utilizadas en conjunto con un escenario que también fue diseñado para ilustrar las operaciones de *encolamiento* y *desencolamiento* de elementos.

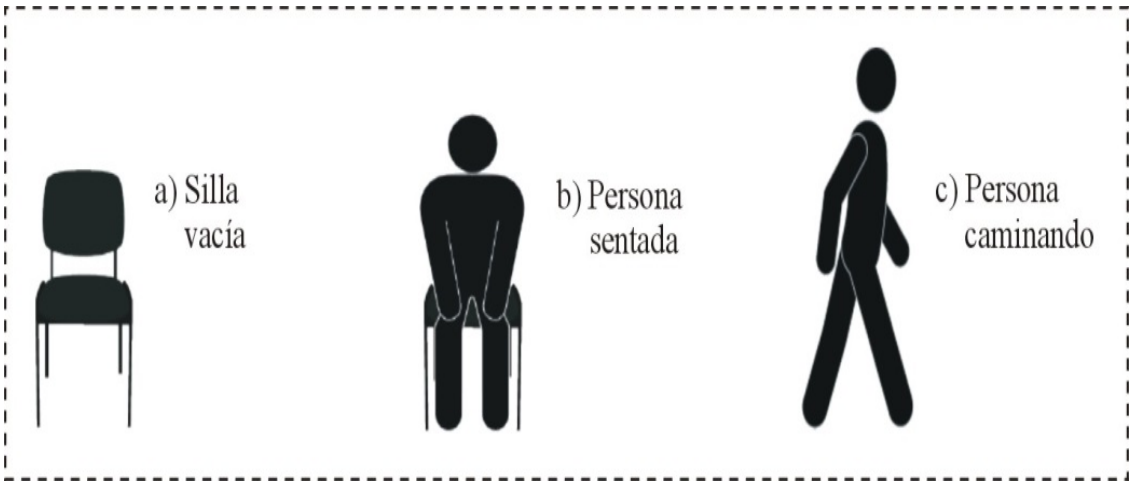

Fig. 8. Imágenes utilizadas en el applet del TDA cola circular.

La Figura 9 muestra el *applet* del TDA cola circula en ejecución, donde se observa una cola que ya ha sido creada con cinco espacios en memoria, representados por las sillas, las cuales se encuentran inicialmente vacías cuando la cola circular ha sido creada. La mecánica de la animación de este *applet* se describe a continuación. El usuario del *applet* del TDA cola circular puede realizar las operaciones de *encolar* y *desencolar*, a través de los botones que se encuentran en la parte inferior.

La operación encolar pone en acción a una persona, la cual camina hasta el primer espacio de memoria disponible de la cola y *encola* el elemento, lo cual es representado por el cambio de imagen de la silla vacía a la persona sentada en la silla, simulando que la persona ha ocupado la silla. En este caso la operación *encolar* puede ser ejecutada tantas veces como sillas vacías existan, ya que en una cola circular se pueden *encolar* elementos en espacios de memoria que ya han sido previamente ocupados (elementos que han sido *desencolados*).

Fig. 9. Applet del TDA cola circular en ejecución.

La operación *desencolar* pone en acción a la persona que se encuentra sentada en la silla al inicio de la cola circular, la cual se levanta de la silla y la deja libre, es decir, *desencola* el elemento de la cola circular, lo cual es representado por el cambio de imagen de la silla ocupada a la silla vacía, simulando que la silla queda libre al haberse levantado la persona de la silla.

En el caso específico de la Figura 9, el usuario *encoló* dos elementos en la cola, y *desencoló* un elemento, como se puede observar por las imágenes de las sillas: el primer espacio de memoria ha sido *desencolado*, ya que la persona se ha levantado de la silla; el siguiente espacio de memoria tiene un elemento *encolado* (silla ocupada por la persona sentada), y los últimos espacios de memoria están aún vacíos (sillas vacías). La animación termina cuando el usuario presiona el botón de reiniciar, con el cual se reinicia el *applet* y se puede crear una nueva cola con un número determinado de espacios de memoria (sillas vacías).

7. Pruebas

En esta sección se muestran algunas de las pruebas que se realizaron a cada uno de los *applets* desarrollados, las cuales consistieron en probar que los casos de uso correspondientes se llevaran a cabo de manera correcta. Solo se muestran dos pruebas realizadas sobre al *applet* del TDA pila. La Tabla 4 muestra el control de prueba para la operación de *Crear Pila*; y en la Tabla 5 para la operación *Push*.

Control de Prueba	
Nombre del módulo: AppPila	**Nombre de prueba:** Crear Pila
Versión de módulo: 03	**Versión de prueba:** 03
Detalle de Prueba	
Fecha de realización: 07/10/14	**Realizado por:** Betzabet García Mendoza
Objetivo:	Crear la estructura de una Pila, con el número de elementos deseados por el usuario.
Operación de prueba:	Se ingresa en el campo de texto el número 4. Se da clic en el botón Crear. El sistema muestra una grúa con cajas a su lado.
Resultado esperado:	El usuario indica el tamaño de la Pila y el sistema la crea del tamaño indicado.
Resultado obtenido:	¿Éxito? Sí (X) No ( ) ¿Por qué?

Tabla 4. Control de prueba: crear pila.

Control de Prueba	
Nombre del módulo: AppPila	**Nombre de prueba:** Push
Versión de módulo: 03	**Versión de prueba:** 03
Detalle de Prueba	
Fecha de realización: 07/10/14	**Realizado por:** Betzabet García Mendoza
Objetivo:	Insertar un elemento en la Pila.
Operación de prueba:	Se da clic en el botón Push. El sistema muestra gráficamente como se inserta un elemento en la Pila.
Resultado esperado:	El sistema inserta un nuevo elemento en la Pila.
Resultado obtenido:	¿Éxito? Sí (X) No ( ) ¿Por qué?

Tabla 5. Control de prueba: push.

8. Conclusiones

En este artículo se presentaron tres *applets* desarrollados en el lenguaje de programación Java, los cuales se utilizan como material didáctico para apoyar los cursos de estructuras de datos de licenciatura. Los *applets* demuestran de una manera dinámica, atractiva, y con ejemplos de la vida cotidiana, el funcionamiento de los tipos datos abstractos pila, cola y cola circular, en los cuales el estudiante interactúa mediante la creación de instancias de cada uno de los tipos de datos abstractos, y ejecutando el conjunto de operaciones definido para cada uno.

Dentro del trabajo futuro se planea desarrollar más *applets* para representar otros tipos de datos abstractos. Adicionalmente, se planea realizar pruebas de funcionalidad y usabilidad con estudiantes, así como incorporar estos *applets* a una plataforma de tutoriales interactivos [7], [8], y manejarlos de manera independiente como un conjunto de objetos de aprendizaje [9].

9. Referencias

[1] A. Aho, J. Hopcroft, J. Ullman, Estructuras de Datos y Algoritmos. Editorial Alhambra Mexicana. 1998. 438 pp.

[2] L. Joyanes, Fundamentos de Programación. Algoritmos, Estructuras de Datos y Objetos. Cuarta Edición. 2008. Editorial McGraw Hill.

[3] Tutorial Interactivo de Estructuras de Datos. Disponible en: http://osiris.ucb.edu.bo/~inf104/index_html/Colas%20y%20pilas.htm. Último acceso en septiembre de 2015.

[4] Una Herramienta para el Estudio de Estructuras de Datos y Algoritmos. Disponible en: http://eprints.ucm.es/6281/1/Cap._2_._p_13-23.pdf. Último acceso en Septiembre 2015.

[5] Software para la enseñanza de los Algoritmos, Departamento de Ingeniería Eléctrica. Disponible en: http://148.206.53.84/ tesiuami/UAMI10755.pdf. Último acceso en Septiembre 2015.

[6] C. Larman. UML y Patrones: Una Introducción al análisis y diseño orientado a Objetos y al proceso Unificado. 2a Edicion. 2008. Prentice Hall.

[7] C. Jaimez-González, C. Sánchez-Sánchez, S. Zepeda-Hernández, "A Web Platform for Creating and Administering Interactive Online Tutorials". Proceedings of the Canada International Conference on Education. Toronto, Canada. 4-7 Abril 2011. 88-92 pp.

[8] C. Jaimez-González, C. Sánchez-Sánchez, S. Zepeda-Hernández, "Creating and Administering Interactive Online Tutorials and Performance Evaluation Tests Through a Novel Web Platform". International Journal for Cross-Disciplinary Subjects in Education (IJCDSE). Infonomics Society. Vol. 2. No. 3. Septiembre 2011. 447-455 pp.

[9] C. Jaimez-González, W. Luna-Ramirez, "Supporting Structured Programming Courses Through a Set of Learning Objects". Proceedings of the IEEE International Conference on Information Society (i-Society). London, UK. 10-12 Noviembre 2014. 124-128 pp.

10. Autores

Betzabet García Mendoza es estudiante de la Licenciatura en Tecnologías y Sistemas de Información (LTSI), en la Universidad Autónoma Metropolitana, Unidad Cuajimalpa (UAM-C). Sus intereses se centran principalmente en el desarrollo de aplicaciones web dinámicas adaptables a diferentes dispositivos.

Pablo Ruíz Mendoza es estudiante de la LTSI en la UAM-C. Sus intereses se centran principalmente en el desarrollo de aplicaciones web dinámicas.

Gerardo Real Flores es estudiante de la LTSI en la UAM-C. Sus intereses se centran principalmente en el desarrollo de aplicaciones web dinámicas.

Carlos Roberto Jaimez González obtuvo el doctorado en Ciencias de la Computación y la maestría en Tecnologías para Comercio Electrónico, ambos por la Universidad de Essex, en el Reino Unido; es Licenciado en Computación por la Universidad Autónoma Metropolitana, Unidad Iztapalapa. Actualmente es profesor investigador del

Departamento de Tecnologías de la Información, de la Universidad Autónoma Metropolitana, Unidad Cuajimalpa; es miembro del Sistema Nacional de Investigadores. Dentro de sus intereses de investigación están las tecnologías y sistemas para apoyar la educación superior, interoperabilidad en sistemas distribuidos, objetos distribuidos y servicios web, XML y tecnologías relacionadas, aplicaciones web y móviles para comercio electrónico, así como sistemas multiagente.

Esaú Villatoro Tello es profesor investigador del Departamento de Tecnologías de la Información, de la Universidad Autónoma Metropolitana, Unidad Cuajimalpa. Dentro de sus intereses de investigación está el procesamiento de lenguaje natural.