Aprendizaje de técnicas avanzadas de Programación Orientada a Objetos mediante programación de juegos

Juan A. Recio García, Baltasar Fernández Manjón, Marco Antonio Gómez Martín
Dpto. Sistemas Informáticos y Programación
Universidad Complutense de Madrid
jareciog@fdi.ucm.es, balta@sip.ucm.es, marcoa@fdi.ucm.es

Resumen

Los juegos constituyen un área muy adecuada para la elaboración de prácticas de programación. En muchos casos, permiten la aplicación directa de los contenidos teóricos impartidos en otras asignaturas teóricas, como las relacionadas con estructuras de datos y algoritmia. Además, al ser prácticas que se alejan de los típicos "problemas de juguete", permiten introducir nuevas técnicas o tecnologías con demanda en el mundo laboral. En este artículo presentamos la experiencia realizada en un curso de Laboratorio de Programación III, en la que hemos propuesto la implementación de Sudokus en Java. Como veremos, el problema nos ha permitido poner en práctica conceptos de estructuras de datos y algoritmia, así como patrones de diseño y programación de dispositivos móviles.

1. Introducción

La asignatura descrita, Laboratorio de Programación III, tiene lugar en el tercer año del plan de estudios de la Ingeniería en Informática. En ese momento, los alumnos ya han cursado dos años de laboratorio de programación, así como dos asignaturas teóricas: una de introducción a la programación y otra de estructuras de datos. Durante ese año, además, muchos de ellos están cursando Metodología y Tecnología de la Programación, en la que se presentan diversas técnicas algorítmicas para resolver problemas. Por último, es muy posible que cursen también una asignatura en la que se dan a conocer los conceptos teóricos de Programación Orientada a Objetos (POO).

En este contexto, nuestra asignatura, eminentemente práctica, debe proponer la implementación de aplicaciones interactivas que impliquen el uso de esquemas algorítmicos. Después de un año con Pascal y otro con C/C++, este laboratorio utiliza Java [1].

La dinámica de la asignatura es la siguiente: se inicia con el desarrollo una serie de pequeñas aplicaciones de consola, que exponen los pilares fundamentales del lenguaje de programación, para después pasar a explicar los mecanismos de diseño de interfaces gráficas. Durante el año, pretendemos que los estudiantes capten la *filosofía* de Java, a saber, la independencia de la plataforma. Queremos que al finalizar el curso entiendan el lema o slogan inicial de sus creadores, "W*rite Once, Run Anywhere*", que ha provocado su popularidad.

Sin embargo, la asignatura no debe quedarse en la descripción y puesta en práctica del lenguaje de programación y sus librerías, sino que debe también aplicar los conceptos explicados en las asignaturas cursadas en los años anteriores y en las que están siendo cursadas en ese momento. En particular, es deseable que el laboratorio proponga prácticas en la que se utilicen distintas estructuras de datos, así como la implementación de algoritmos vistos en Metodología y Tecnología de la Programación. También es fundamental poner en práctica todos los conceptos que están siendo explicados en la asignatura de Programación Orientada a Objetos. Finalmente, hay que destacar que ésta constituye la última asignatura de laboratorio a la que se enfrentarán los alumnos durante sus estudios. Por lo tanto, es la última oportunidad para *explicar* los contenidos de carácter práctico. El año siguiente, nuestros alumnos cursarán Ingeniería del Software, en la que el esfuerzo del uso de cualquier tecnología en la implementación del proyecto (programación de dispositivos móviles, aplicaciones Web, etc.), debe ser asumida por el alumno [2], por lo que queríamos enseñar algunas de estas tecnologías para su uso en el proyecto del año siguiente.

Está claro que todos estos objetivos hacen que el diseño de las prácticas a realizar durante el año es complicada. Además, los alumnos pueden verse desbordados por la cantidad de materia a apren-

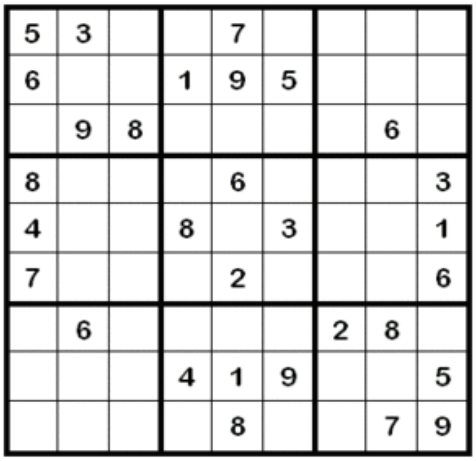

Figura 1. Ejemplo de rompecabezas Sudoku

der. Para tratar de contrarrestarlo, era primordial diseñar prácticas que consigan un alto grado de motivación.

Para esto último, parece que los juegos favorecen la aceptación por parte de los estudiantes de las prácticas [3]. La implementación de un juego ampliamente extendido hace que muchos alumnos vean en la aplicación final una meta atrayente, y no un mero fin para aprobar una asignatura.

En este artículo describimos una serie de prácticas propuestas durante el curso 2005-2006. En particular, se trata de la implementación de varias versiones de una aplicación para resolver Sudokus, que permite tanto la resolución manual por parte del usuario, como la automática por parte de la máquina.

Como veremos en los apartados siguientes, las prácticas nos ha permitido cubrir muchos de los objetivos citados previamente. En particular, el desarrollo fuerza el uso de metodologías avanzadas de diseño de clases, aplicando patrones de diseño [4]. Los patrones no solo los han tenido que utilizar como usuarios de las clases involucradas al programar con *frameworks* como Swing, sino también en el momento de diseñar la jerarquía de clases de las aplicaciones. También hemos impulsado la reutilización de código y la idea de lenguaje multiplataforma, pues la práctica debía ser implementada para PC, utilizando un interfaz de consola, para después utilizar la librería de componentes Swing de Java, y por último su adaptación para dispositivos móviles utilizando la arquitectura J2ME. Este último aspecto, además, nos ha permitido introducir una tecnología nueva que, muy posiblemente, aprovecharán al año siguiente en el desarrollo del proyecto de Ingeniería del Software.

En el siguiente apartado detallamos las normas del juego elegido, así como la justificación de su elección. La Sección 3 describe qué conceptos sobre algoritmia se ponen en práctica con el Sudoku. La Sección 4 introduce los patrones de diseño y su importancia en el desarrollo de software. Posteriormente, la Sección 5 explica la arquitectura planteada a los alumnos para la realización del Sudoku, mientras que la Sección 6 muestra el proceso de adaptación a dispositivos móviles. Para terminar, la Sección 7 presenta las conclusiones de la experiencia.

2. Sudoku

Sudoku [5] es un rompecabezas matemático de colocación que se popularizó en Japón en 1986 y se dio a conocer en el ámbito internacional en 2005. El objetivo es rellenar una cuadrícula de 9×9 celdas (81 casillas) dividida en subcuadrículas de 3×3 (también llamadas "cajas" o "regiones") con las cifras del 1 al 9 partiendo de algunos números ya dispuestos en algunas de las celdas. No se debe repetir ninguna cifra en una misma fila, columna o subcuadrícula. Un sudoku está bien planteado si la solución es única. La Figura 1 muestra un rompecabezas Sudoku. La resolución manual del problema requiere paciencia y ciertas dotes lógicas. La resolución de Sudokus por ordenador abordable en el nivel en el que se imparte la asignatura es el uso de Vuelta-Atrás.

Existen muchos juegos que podrían haberse elegido como práctica para este laboratorio. Las razones que llevaron a utilizar los Sudokus fueron:

- El juego se ha vuelto muy popular a partir del año 2005. La mayoría de los periódicos publican Sudokus en su sección de pasatiempos, por lo que es muy probable que sus reglas sean familiares para los alumnos.
- El modelo Orientado a Objetos de los problemas Sudokus es muy intuitivo y fácil de implementar ya que el juego se compone de un tablero, bloques, celdas, etc.

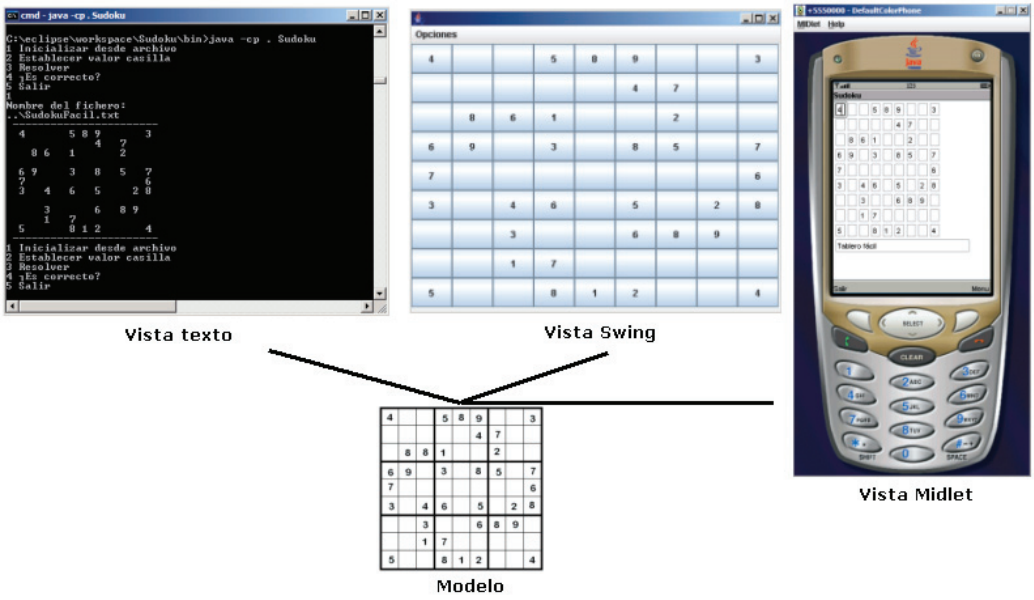

Figura 2. Diferentes vistas del Sudoku

- La resolución de los Sudokus permite utilizar distintos algoritmos de búsqueda estudiados desde el punto de vista teórico en otras asignaturas.
- La interfaz de usuario es relativamente simple, lo que nos permite implementarla tanto en modo texto/consola, con ventanas y botones (GUI Swing), utilizando tecnologías Web (Applets, JSP, Servlets) o tecnologías móviles (J2ME).

El amplio conjunto de tecnologías Java que pueden aplicarse para implementar un resolutor de Sudokus permite plantear prácticas de dificultad incremental que vayan introduciendo nuevos conceptos. De esta forma, se ahorra el tiempo necesario para crear una nueva práctica desde cero, se aprende a reutilizar código y se trabaja con prácticas de un tamaño mayor. Por ejemplo, si hemos desarrollado el Sudoku en modo texto podemos reutilizar toda la lógica a la hora de implementarlo utilizando una interfaz gráfica.

Las aplicaciones planteadas a los alumnos incluyen tanto la creación de problemas Sudoku como su resolución utilizando distintos algoritmos. Además, el sistema permitirá conocer si la solución introducida manualmente por el usuario es correcta.

3. Algoritmos

La elección del Sudoku permite que los alumnos apliquen los algoritmos estudiados en las asignaturas de *Estructura de Datos y de la Información* y *Metodología y Tecnología de la Programación*. En estas asignaturas se estudian diferentes algoritmos de búsqueda aplicables a la resolución de Sudokus.

Además de llevar a la práctica los conceptos teóricos, el sistema permite que los alumnos experimenten por ellos mismos la importancia de la complejidad computacional. Esta materia (explicada en las dos asignaturas citadas) estudia los distintos grados de complejidad de los algoritmos. Sin embargo, en muchos casos su estudio se realiza sólo en el campo teórico. Cuando los alumnos implementan los algoritmos de resolución de Sudokus ven de primera mano las consecuencias de las complejidades algorítmicas de las estrategias implementadas [6]. Por ejemplo, al implementar la estrategia de fuerza bruta (sin poda) observan que la búsqueda nunca termina por muy potente que sea la máquina en la que ejecutan la

estrategia. Obviamente, esto se debe a que una búsqueda en un espacio de estados con 9^{81} posibilidades plantea una complejidad inalcanzable actualmente. Esta observación se complementa cuando los alumnos implementan los algoritmos de búsqueda con podas. Al realizarlos, observan que estas estrategias sí que terminan en un tiempo apropiado.

Aunque queda fuera del alcance de los alumnos que cursan esta asignatura, la práctica de Sudoku puede ser utilizada para probar otros algoritmos estudiados en asignaturas de cursos posteriores. Es el caso de técnicas de búsquedas informadas o heurísticas explicadas en asignaturas de Inteligencia Artificial del ciclo superior de la Ingeniería. Siguiendo las ideas descritas en [7], se podría implementar el resolutor de Sudokus utilizando un sistema de reglas.

En relación con las posibles implementaciones de los algoritmos tenemos que adelantar aquí una de las características del entorno de desarrollo J2ME. Este entorno cuenta con un sistema emulador de dispositivos móviles que tiene la característica de poder simular la velocidad real de los dispositivos emulados. De esta forma podemos establecer una velocidad de ejecución fija independiente de la potencia del PC donde se ejecuta el emulador. Esta característica permite medir la eficiencia de los algoritmos independientemente de la máquina donde se ejecuten. Para motivar a los alumnos, les ofrecemos en la última parte que intenten mejorar los tiempos de respuesta de la implementación de ejemplo realizada por nosotros.

Esta característica también tiene otra ventaja de cara a la experiencia docente: permite ejemplificar a los alumnos que en determinados escenarios no es posible programar suponiendo una gran disponibilidad de recursos. Los alumnos tienden a adoptar la opinión de que no es necesario optimizar los algoritmos ni intentar minimizar los recursos, al dar por hecho que la máquina será siempre tan potente como para ejecutar su código sin ningún problema de eficiencia. Hemos notado esta suposición al revisar las distintas implementaciones de los alumnos en las estrategias de búsqueda. Cuando después se les pide que ejecuten su código con limitaciones de velocidad y memoria (en el emulador de dispositivos móviles) se dan cuenta de que la disponibilidad ilimitada de recursos no siempre es cierta y que en determinados entornos la eficiencia es determinante. Al optimizar las estrategias implementadas observan las mejoras de rendimiento producidas por la modificación de los distintos elementos estudiados en teoría: estructuras de datos inapropiadas, estructuras de control mal definidas, etc.

4. Patrones de diseño

Los *Patrones de diseño* [4] son un conjunto de técnicas o diseños software que permiten escribir aplicaciones extensibles y reutilizables. La popularidad de los patrones se ha incrementado notoriamente en los últimos años al utilizarse para la creación de grandes aplicaciones, armazones o incluso el propio Java. Según ha aumentado su importancia, ha crecido la demanda de profesionales con conocimientos sobre patrones en el mundo laboral. Sin embargo, los patrones no tienen un lugar específico dentro de ninguna asignatura de la carrera de Informática, ofreciéndose en algunos casos como asignatura optativa, seminario o dentro del temario de Ingeniería del Software.

Debido a la importancia actual de estas técnicas hemos querido diseñar nuestro Sudoku aplicando algunas de ellas (la más importantes) y de esta forma brindar a los alumnos la posibilidad de utilizarlas en un ejemplo claro, práctico e intuitivo [8].

Entre los patrones de diseño destaca la arquitectura Modelo-Vista-Controlador (MVC). Esta arquitectura permite intercambiar fácilmente las interfaces de usuario de los programas. Esta característica es muy importante en los desarrollos software actuales. Por ejemplo, cualquier aplicación que se realice con ventanas tiene que estar preparada para su integración en la Web. Aunque la lógica no varía, el modo de presentar los datos al usuario cambiará sustancialmente de una tecnología a otra.

Los diseños MVC consisten en tres tipos de objetos: El *Modelo* implementa la lógica del programa, la *Vista* implementa la interfaz que muestra los datos al usuario, mientras que el *Controlador* define cómo se reacciona ante la entrada del usuario (se encarga de orquestar el flujo de datos entre el Modelo y la Vista). Si no se utiliza MVC las aplicaciones suelen mezclar la lógica y presentación de la aplicación, acoplando el código y dificultando su reutilización. MVC desacopla la presentación y la lógica estableciendo-

do un protocolo de suscripción y notificación entre ellos. Una vista deberá reflejar el estado del modelo y cuando éste cambie será notificada para actualizarse.

Este mecanismo permite enlazar varias vistas con el mismo modelo o intercambiarlas fácilmente. Por ejemplo, la primera práctica sobre el Sudoku propone el desarrollo del juego con una interfaz textual (consola). En las sucesivas prácticas se añadirá la interfaz gráfica con Swing y la interfaz para dispositivos móviles con MIDP. Aunque nosotros, por limitación temporal, no proponemos el desarrollo de una interfaz Web, el diseño planteado posibilita una fácil adaptación en caso de incluir estas materias en el temario del curso.

La Figura 2 muestra el diseño MVC utilizado por nuestra aplicación Sudoku. El modelo se comparte por las tres vistas permitiendo su intercambio de forma sencilla y organizada.

Para implementar el mecanismo de suscripción/notificación de eventos del MVC utilizaremos el patrón de diseño *Observer*. Además la aplicación debe permitir la utilización de distintas estrategias de resolución de Sudokus. Para implementar dicha funcionalidad existe una solución muy elegante basada en los patrones de diseño *Factoría Abstracta* y *Estrategi*a.

Aunque nos centremos en la utilización de patrones para la creación de una aplicación debemos resaltar que este tipo de diseños se utilizan extensivamente en muchas librerías y armazones. Entre ellas encontramos los paquetes Swing para el desarrollo de interfaces gráficas en PCs y el paquete MIDP para la creación de interfaces gráficas en dispositivos móviles. El dominio de estos paquetes, especialmente Swing, es uno de los objetivos de nuestro Laboratorio de Programación así como de la mayoría de cursos avanzados de Java. Los componentes de Swing junto con su control de eventos están claramente diseñados mediante el patrón de diseño *Model-Delegate:* una versión simplificada del MVC donde el controlador y la vista se implementan en la misma clase. Mediante el estudio de Patrones de Diseño conseguimos que los alumnos comprendan sin problemas el funcionamiento de Swing o MIDP además de aprender a crear nuevas aplicaciones bien estructuradas.

Los detalles de todos estos patrones se explican en la siguiente sección junto con la arquitectura de nuestro sistema Sudoku.

5. Arquitectura

El diseño de nuestro sistema implementa fielmente la arquitectura MVC. Para ello se divide la aplicación en dos paquetes: modelo y vista. La Figura 3 muestra la organización de código planteada a los alumnos. Esta figura ilustra la estructura de paquetes del código, donde se aprecia la existencia de un paquete para el modelo y dos para las vistas (consola y swing). En el paquete de clases del modelo aparecen las clases que almacenan la estructura de datos del tablero. Además, existe un subpaquete donde se implementan las distintas estrategias de resolución de Sudokus.

En la primera aproximación, no incluimos el objeto controlador dentro del diseño para que los alumnos comprendan progresivamente el funcionamiento del MVC. De esta forma empezamos implementando un sistema más cercano al patrón Observer o Model-Delegate. Una vez que los alumnos asimilan su funcionamiento ampliamos la aplicación para implementar el patrón MVC completo.

Para desarrollar toda esta arquitectura de forma progresiva y coherente hemos divido las prácticas en tres etapas:

1. La primera etapa explica cómo desarrollar aplicaciones donde la lógica está separada de la interfaz al usuario, cómo estructurar el código para poder aplicar diversas estrategias de resolución en un mismo problema, y por último cómo elegir de forma elegante el tipo de estrategia.
2. La segunda parte de la práctica de Sudoku pide implementar la interfaz gráfica. Las características solicitadas son bastante simples, por lo que puede servir como una primera introducción a Swing. Aparte de este objetivo, se pretende que el alumno se dé cuenta de la facilidad de intercambiar la interfaz de usuario (vista) cuando se ha realizado el diseño MVC.
3. Por último, la parte final solicita crear una vista para dispositivos móviles utilizando la tecnología J2ME. Desde el punto de vista de la arquitectura, este desarrollo es similar a la segunda parte de la práctica ya únicamente hay que implementar la interfaz gráfica y añadirla al diseño MVC.

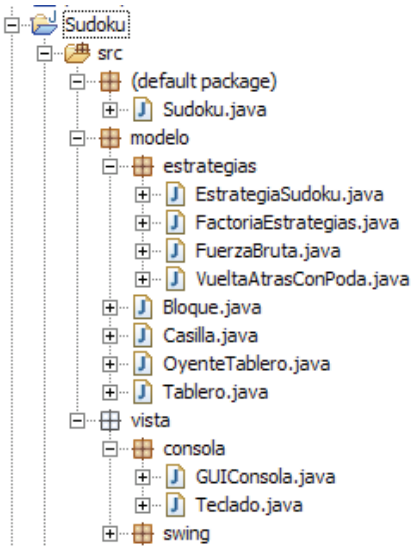

Figura 3. Arquitectura del sistema

La ventaja de este planteamiento es que la lógica desarrollada en la primera parte debe ser reutilizada en las siguientes etapas. Por lo tanto, las entregas de las dos últimas partes harán que el alumno se dé cuenta de las carencias de su implementación del modelo. De esta forma obtiene una buena experiencia de cara a futuros desarrollos reutilizables.

Para que las vistas puedan presentar el estado del modelo se utiliza el patrón *observer*. Este patrón se basa en una interfaz implementada por todos los componentes que quieren presentar el estado del modelo. Por lo tanto todas las vistas implementarán esa interfaz y se registrarán en el modelo como *oyentes* del mismo. Cuando haya un cambio en el estado del modelo, él mismo se encargará de avisar a todos los oyentes registrados. Esta parte es la que permite entender la filosofía MVC y el funcionamiento de la librería Swing y MIDP, por lo que la primera práctica se centra en su implementación.

Para permitir el fácil intercambio entre las distintas estrategias de resolución descritas en la Sección 3, en la primera parte de la práctica también se implementa el patrón de diseño S*trategy*. Este tipo de diseño establece que todas las estrategias de resolución deben implementar una interfaz con un método *resuelve()*. De esta forma cualquier estrategia podrá ser invocada de una manera estándar.

Para elegir qué estrategia utilizar, normalmente los alumnos suelen usar un gran bloque de código *if-else*. En este punto se les indica que si fuera necesario añadir más estrategias habría que seguir agrandando ese código, obteniendo finalmente un bloque bastante ilegible. Entonces se presenta la solución basada en el patrón *Factoría Abstracta.* Este diseño centraliza todo el bloque condicional en una única clase con un método abstracto que elige el tipo de estrategia dependiendo de una determinada constante.

6. Sudoku en dispositivos móviles

Cuando se elabora el temario para un curso de programación en el lenguaje Java suele seguirse el mismo esquema que para otros lenguajes como C++. Sin embargo, Java cuenta con muchas otras posibilidades que simplifican mucho el desarrollo de software en comparación con otros lenguajes de programación. Los temarios más comunes se centran sólo en una parte de toda la tecnología Java: Java 2 Standard Edition (J2SE) y, como mucho, entran en la parte de programación de aplicaciones con interfaz gráfica de usuario. J2SE incluye una API de programación similar a otros lenguajes y debe ocupar la mayor parte de la asignatura al ser el pilar fundamental de Java. Sin embargo también debemos dejar hueco a otras características importantes del lenguaje: Java 2 Enterprise Edition (J2EE) y Java 2 Mobile Edition (J2ME). La primera de ellas incluye las tecnologías necesarias para desarrollar aplicaciones Web, mientras que la segunda permite desarrollar aplicaciones para dispositivos móviles. En la Figura 4 se muestra la organización de estas tres tecnologías y las plataformas a las que están dirigidas. Aunque J2SE sea el eje central de toda la arquitectura Java, hay que tener en cuenta que las otras tecnologías están siendo muy reclamadas en el mundo laboral. La tercera parte de la práctica de Sudoku está diseñada para que los alumnos tengan una mínima experiencia en estos campos. Aunque hemos elegido J2ME, también podríamos haber desarrollado la versión Web del Sudoku con las tecnologías J2EE. Sin embargo, esta tecnología sí se cubre en otras asignaturas opcionales y, por lo tanto, los alumnos tienen oportunidad de conocerla fuera de este Laboratorio de Programación.

El desarrollo de aplicaciones en J2ME tiene otra ventaja nada despreciable: la motivación extra de los alumnos. No cabe duda de que los jóvenes de hoy en día tienen una gran afición por los teléfonos móviles. La posibilidad de programar un juego que luego puedan instalar y ejecutar desde sus terminales hace que los estudiantes presten una especial atención a las prácticas.

6.1. Programación con J2ME

La arquitectura de J2ME [9] define *configuraciones* (configurations), *perfiles* (profiles) y paquetes opcionales, como elementos básicos para desarrollar aplicaciones que se ajustan a las características de un amplio rango de dispositivos. Cada combinación se optimiza según la memoria, la capacidad de procesamiento y de entrada/salida de una categoría específica de dispositivos.

Las configuraciones están compuestas por una máquina virtual y un conjunto mínimo de bibliotecas de clases, que hacen de mínimo denominador común con que contarán todos los dispositivos de una configuración dada. Como muestra la parte inferior de la Figura 4 existen dos tipos de configuraciones: *Connected Limited Device Configuration* (CLDC) y *Connected Device Configuration* (CDC). La primera de ellas está diseñada para dispositivos pequeños con procesador lento, poca memoria y conexiones de red intermitentes: teléfonos móviles, PDAs, etc. La configuración CDC está orientada a los aparatos de potencia comprendida entre los que trata CLDC y los ordenadores domésticos. En nuestra práctica del Sudoku utilizaremos la configuración CLDC.

Con objeto de ofrecer un completo entorno de ejecución específico para cada categoría de dispositivos, las configuraciones se deben combinar con un conjunto de APIs de alto nivel (perfiles), que definen el modelo de ciclo de vida de la aplicación, el interfaz de usuario y el acceso a las propiedades específicas del dispositivo. Existen varios perfiles que posibilitan distintos tipos de desarrollos: programación de PDAs con grandes prestaciones, programación de juegos, etc. En nuestro caso elegiremos el *Mobile Information Device Profile* (MIDP) diseñado específicamente para teléfonos móviles.

Respecto a las herramientas necesarias para desarrollar aplicaciones J2ME sólo es necesario *Wireless Toolkit* [10], un entorno de desarrollo gratuito de Sun. Este paquete incluye las herramientas de compilación y las APIs necesarias para la programación de dispositivos móviles inalámbricos. Aunque este entorno permite compilar y ejecutar las aplicaciones en un emulador de terminales móviles, no incorpora ningún IDE que ofrezca posibilidades de edición y depuración de código. Para resolver este inconveniente hemos utilizado Eclipse, empleado durante todo el curso, junto con su plugin para aplicaciones J2ME, EclipseME [11]. Este plugin tiene una instalación y configuración muy sencilla, lo que permite que el alumno empiece el desarrollo de software sin perder apenas tiempo en configurar el entorno.

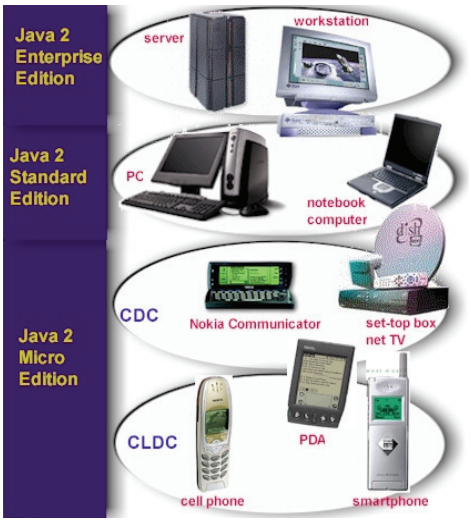

Figura 4. Arquitectura de Java 2

6.2. Adaptación del Sudoku a MIDP

Una vez configurado el entorno de desarrollo podemos empezar la programación de nuestro Sudoku para móviles. Partiremos del código del modelo y crearemos una nueva vista. Al programar sobre CLCD+MIDP sólo tendremos un subconjunto de clases de J2SE disponible. Por ejemplo, este perfil sólo incluye un tipo de lista *Vector* por lo que los alumnos deberán adaptar los otros tipos de datos utilizados. Aunque estas limitaciones puedan parecer un inconveniente no lo son ya que el Sudoku es perfectamente programable utilizando las clases que ofrece MIDP. Si el alumno utiliza otro tipo de estructuras, éstas serán redundantes o innecesarias. Por lo tanto, la programación con un API limitada enseña al alumno

a programar con los recursos más apropiados según sus objetivos.

La implementación de la nueva vista en MIDP no presenta grandes problemas a los alumnos. Al igual que otras librerías para el desarrollo de interfaces de usuario podemos utilizar una clase formulario junto con sus diferentes elementos gráficos de interacción (widgets) como etiquetas, cuadros de texto, etc. La gestión de eventos se organiza de forma similar a la de Swing por medio del patrón *Model-Delegat*e. De nuevo, el estudio de los Patrones de Diseño nos permite que los alumnos entiendan esta nueva librería sin dificultad.

En pocas sesiones los alumnos dispondrán de un juego totalmente funcional que podrán empaquetar e instalar en sus propios teléfonos móviles.

7. Conclusiones

En este artículo hemos presentado un conjunto de prácticas que se pueden integrar en las asignaturas de programación en Java. El marco propuesto permite ejercitar a los alumnos en los Patrones de Diseño y las arquitecturas Modelo-Vista-Controlador. También permite aplicar los métodos algorítmicos estudiados en las asignaturas teóricas de la carrera y comprobar de primera mano las consecuencias de los costes computacionales de los mismos. Por último, hemos aprovechado la arquitectura propuesta para estudiar otras tecnologías ofrecidas por la plataforma Java. Concretamente, como elemento motivador, invitamos a los estudiantes a adaptar la aplicación para que pueda ejecutarse en sus teléfonos móviles. Esta parte se aprovecha de la popularidad de este tipo de dispositivos entre los jóvenes y nos permite mostrar a los estudiantes la importancia de la optimización de recursos y algoritmos en ciertos entornos limitados.

La utilización del juego Sudoku como marco de aprendizaje de conceptos avanzados de programación en Java es el resultado de la evolución de la asignatura durante varios años. En las sucesivas ediciones se ha venido utilizando la temática de los juegos (cuatro en raya, hundir la flota, buscaminas, …) para ilustrar los conceptos presentados en este artículo.

Referencias

[1] Sánchez Allende, J., Huecas, G, Fernández Manjón, B., Moreno, P., *Java 2: Iniciación y Referencia (2ª edición)*. Ed. McGraw-Hill. 2005.

[2] Gervas, P, Gómez Martín, M.A., Sarasa, A. *Ingeniería del Software: ¿Basta con desarrollar proyectos o haría falta probar a implantar procesos de desarrollo a largo plazo?* VII Jornadas de Enseñanza Universitaria de la Informática (JENUI), 2001

[3] Becker, K., Parker, J. R. *All I Ever Needed to Know About Programming, I Learned From Re-writing Classic Arcade Games*, International Academic Conference on the Future of Game Design and Technology, Michigan, USA, 2005.

[4] Gamma, E., Helm, R., Johnson, R., Vlissides, J.. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Professional Computing Series. 1994.

[5] Entrada "Sudoku" de la Wikipedia (versión inglesa) http://en.wikipedia.org/wiki/Sudoku*

[6] Levitin, A. *Analyze that: puzzles and analysis of algorithms*. In Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education. ACM Press .2005.

[7] Gómez Martín, M.A., Díaz Agudo, B. *Evaluación de estrategias de razonamiento para sistemas basados en reglas*. XII Jornadas de Enseñanza Universitaria de la Informática (JENUI), 2006

[8] Nguyen, D., Wong, S. B. *Design patterns for games*. In Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education. ACM Press. 2002.

[9] Ortiz, E. *A Survey of J2ME Today*. http://developers.sun.com/techtopics/mobility/getstart/articles/survey/*

[10] Java Wíreles Toolkit. http://java.sun.com/products/sjwtoolkit/*

[11] Eclipse ME. http://eclipseme.org/*

(*todos los enlaces fueron accedidos en Abril de 2006)