Java, herramienta docente en asignaturas del área de Arquitectura de Computadores

J. A. Álvarez Bermejo, J. F. Sanjuán Estrada, D. P. Morales Santos
Dpto. Arquitectura de Computadores y Electrónica.
Universidad de Almería.
e-mail: jaberme@ual.es

Resumen

Los últimos años han supuesto un cambio notable en el currículo de ingenierías informáticas, cambio impuesto por avances tanto tecnológicos como metodológicos. Toda esta serie de avances han traído consigo la aparición de lenguajes como Java (lenguaje y arquitectura) que permiten, desde el punto de vista docente, asentar conocimientos sin sacrificar la profundidad de los mismos debido a una dificultad excesiva impuesta por el entorno usado en clase.

Este trabajo pretende mostrar algunas ventajas de Java como recurso docente, proporcionando además un compendio de recursos. Proponemos usar Java en asignaturas propias del área de Arquitectura de Computadores tales como Multiprocesadores, Redes y Estructura de Computadores. Estas asignaturas han estado tradicionalmente ligadas a complejos lenguajes que hacían, aún si cabe, más árida la materia para el alumno. Usar Java en estas asignaturas es añadir un reactivo para los alumnos por que:

- Es orientado a objetos y más sencillo de aprender que cualquier otro lenguaje plausible que se pueda usar en las citadas asignaturas.
- Muchos alumnos lo encuentran atractivo.
- La gran mayoría valora aprender un lenguaje que con bastante probabilidad usarán en su trayectoria laboral.
- El hecho de que sea tan sencillo de aprender permite centrar las clases en el diseño y solución de problemas, el lenguaje ya no es un inconveniente.

Este trabajo también presenta la contrapartida de usar Java, por ejemplo, no emplear lenguajes como Occam, pero esto no debe ser problema si el objetivo es que adquieran conocimientos y habilidad para usarlos independientemente del lenguaje de programación empleado.

1. Introducción

Al plantear Java como alternativa a las herramientas docentes ya implantadas y tradicionales (C junto con MPI (Message Passing Interface) en multiprocesadores, por ejemplo) podemos preguntarnos si en realidad no nos dejamos llevar por modas. Desde nuestro punto de vista no cabe tal cuestión y la razón puede encontrarse en nuestra resistencia a considerar a Visual Basic como herramienta de apoyo docente, sin embargo, Java es diferente, Java posee construcciones que reflejan en conjunto los avances en metodología de la programación de los últimos diez años.

El replantearse qué herramientas docentes usar para asentar los conceptos teóricos es una actividad recomendable, sobre todo para la formación de nuestros alumnos. En nuestro caso la valoración de Java como herramienta docente no tiene como justificación al lenguaje Java sino a Java como contenedor de conceptos, habilidades de programación y abstracciones que van ligadas al lenguaje. En este sentido, el hecho de que Java, por un lado sea sencillo de aprender y por otro lado sea cada vez más popular en el entorno empresarial y comercial, en detrimento de Visual Basic, son simplemente los añadidos que nos han obligado a detener nuestra marcha y recapacitar.

Java presenta dos conceptos en uno, por un lado Java es lenguaje de programación y por otro Java es una máquina virtual. Dualidad que aprovecharemos al máximo.

Antes de continuar se hace necesario aclarar que la supuesta falta de eficiencia de Java en los entornos en los que nos adentraremos (paralelismo, concurrencia,…) está originada en la implementación de la máquina virtual que interpreta el bytecode. Java no es ineficiente por si. El motivo de esta aclaración es eliminar

barreras comparativas con los entornos / lenguajes que se puedan estar utilizando en estos momentos, por tanto, el único matiz que nos puede preocupar entre Java y los entornos usados en la actualidad es el del rendimiento de cada alumno con uno y otro entorno. ¿Existen diferencias cuantitativas entre lo que se imparte usando Java y usando C/C++?, la respuesta es NO, la única diferencia radica en que al usar Java, el desgaste y esfuerzo en el alumno es bastante menor que por ejemplo usando C++. ¿Cómo lo hemos cuantificado? El tiempo que cada alumno emplea en depurar sus programas es inferior con Java.

Algunas de las ventajas que aparecen al implantar Java y que por otro lado consideramos positivas para el alumno son:

- Construcción de software reutilizando componentes ya desarrollados. Java acentúa la importancia de localizar, comprender y utilizar software ya existente lo que exige habilidades técnicas que se le proporcionan en otras asignaturas haciendo que asignaturas aparentemente aisladas como multiprocesadores se acerquen algo más al resto de asignaturas.
- Programación y diseño orientado a objetos (O.O.). El paralelismo es una habilidad inherente a la orientación a objetos y viceversa lo que hace que Java se adapte completamente al propósito de implantación en el área de Arquitectura de Computadores. Los aspectos de O. O. que posee Java están completamente integrados en el lenguaje de manera natural haciendo innecesario explicaciones previas, a esto como facilidad añadida podemos decir que en la sintaxis Java es muy similar a C de tal forma que la curva de aprendizaje se minimiza.
- Programación reactiva. El software que se diseña hoy día responde a eventos generados vía una Interfaz de Usuario. Los estudiantes que se anclan en el uso de la consola (entornos de texto) adquieren habilidades sesgadas para mantener proyectos software. Java proporciona un buen punto de entrada en este campo con los applets. Una aplicación concurrente o paralela que además presente interfaces de usuario agradables se desarrolla invirtiendo muy poco tiempo con Java.

Ninguno de los autores de este trabajo nos atrevemos, de manera objetiva, a predecir la longevidad de Java como herramienta de desarrollo puntera en la Web, en entornos multimedia o en aplicaciones del estilo, lo que si podemos asegurar es que el nivel de integración para las tareas de programación cotidianas que posee Java no desaparecerá a medio o a largo plazo.

El aspecto común de todas las características presentadas es que las técnicas de programación habituales se apoyan cada vez más en principios de ingeniería y diseño además de los principios tradicionales de computación y algorítmica. La necesidad de comprender estructuras de datos clásicas, algoritmos, formalismos y técnicas analíticas no ha desaparecido, ni debe desaparecer. Sin embargo, estos principios no deben estudiarse de manera aislada o en exclusión de los demás. Quizás alguna vez fue cierto que implementar una estructura de datos eficiente era suficiente para poder crear un buen programa, sin embargo hoy día y siendo testigos de que la generalidad del software ha crecido también se han hecho necesarios, en todos los campos, conceptos como la encapsulación, estructuración y coordinación del software en componentes que formen parte de un sistema de nivel superior.

En este trabajo se pretenden revelar, como ya se ha adelantado, aspectos positivos de Java como herramienta docente en asignaturas propias del área de Arquitectura de Computadores, desfavorecidas en muchas ocasiones por el aislamiento y especialidad de los lenguajes empleados para practicar conceptos teóricos. En la sección 2 se presenta un compendio de recursos para las asignaturas de Multiprocesadores, Redes de Computadores y Estructura de Computadores. La sección 3 presenta las conclusiones de las pruebas realizadas al emplear Java. La sección 4 resume el trabajo futuro que los autores pretenden llevar a cabo en el campo docente con Java.

2. Asignaturas y recursos

En esta sección describiremos la forma de emplear los recursos que aconsejamos en cada asignatura. Dentro de cada asignatura hemos realizado una división en función de la complejidad de los conceptos que se barajan pudiendo así diferenciar cuando es importante el uso de Java para afianzar conocimientos y cuando es importante el uso de Java por su diseño como lenguaje y arquitectura. Así podremos diferenciar conceptos básicos y conceptos avanzados.

2.1. Multiprocesadores

Entre las características destacables de Java como lenguaje de propósito general Orientado a Objetos (O. O.) encontramos:

- Fuerte comprobación de tipos.
- Ausencia de punteros.
- Liberación de memoria desatendida.
- Biblioteca gráfica.
- Capacidad multihebra inherente.
- Bibliotecas estándar para facilitar la computación en red.

Las tres últimas características hacen de Java un buen candidato para esta asignatura. La capacidad multihebra [8] nos permite introducir, casi desde el principio, al alumno el concepto de hebra y como consecuencia directa la concurrencia y la sincronización. Las bibliotecas para programación en red permite que se aborden proyectos de comunicaciones de manera sencilla y se pueda comprender así la forma en la que éstas se llevan a cabo. Concepto importante en asignaturas como multiprocesadores.

Una carencia importante de Java en este campo es la ausencia de capacidad para computación de altas prestaciones (High Performance Computing, en adelante H.P.C.), en máquinas virtuales convencionales, carencia que podemos suplir con bibliotecas y herramientas diseñadas en Java para multiprocesadores y computadores de memoria compartida [13] [6] así como con máquinas virtuales de propósito especifico como [12] y [41] que aíslan convenientemente al alumno de la complejidad de las bibliotecas adicionales.

Como se acaba de citar, Java tiene como una de sus características principales a la programación hebrada y la capacidad de programación en red, pero Java como recurso para multiprocesadores adolece de la ausencia de mecanismos H. P. C. integrados en las máquinas virtuales por defecto. Algunas de las características que lastran a Java para en el campo de la H. P. C. son:

- Java es interpretado.
- El soporte multihebrado para sistemas de memoria compartida es relativamente nuevo [27].
- El modelo de hebras Java hereda los defectos de diseños ancestrales como semáforos y monitores y además es difícil de usar [22] [33] [12].

Los dos primeros problemas se solventan con relativa facilidad gracias a los compiladores JIT[1]. Sin embargo, el último punto versa sobre la robustez de la programación hebrada en Java y ha de ser resuelto antes de poder usar Java con eficacia para la docencia de paralelismo. A este problema se le trata de dar solución en los siguientes párrafos tras mostrar de manera más detallada el problema.

El modelo de programación O. O. de Java difumina la tradicional dicotomía existente entre el hardware y los modelos de programación de multiprocesadores tanto el modelo de memoria compartida como el de memoria distribuida. A pesar de todas las diferencias en ambos modelos, los interbloqueos y en general cualquier tipo de problema de sincronismo son aspectos extremadamente importantes a la hora de desarrollar programas. Además existe el añadido de la latencia generada por el paso de mensajes en sistemas de memoria distribuida. Esta dicotomía se ve ensombrecida si delante de ella colocamos un modelo orientado a objetos.

Obsérvese que invocar un método en un objeto es conceptualmente equivalente a mandar un mensaje que tiene por contenido al objeto que se pasa como argumento en la invocación del método. Por tanto se puede pensar que Java proporciona, por ser O. O., una nueva filosofía para la programación paralela (tradicionalmente se mostraba en esta asignatura usando C y la biblioteca de paso de mensajes MPI) delegando los detalles sobre eficiencia para la implementación de la máquina virtual propia de cada arquitectura. Se puede ver que si los objetos están en máquinas diferentes tendremos el equivalente a un Remote Procedure Call. en toda regla.

Llegados a este punto queda patente la necesidad de una API para que en la programación concurrente y paralela se solucionen los problemas de las primitivas de sincronización a bajo nivel antes mencionados. Para tal efecto existe una API llamada JCSP

[1] JIT: Compiladores "Just In Time" compilan bytecode directamente a código nativo de la plataforma.

basada en el lenguaje teórico CSP (Communicate Sequential Processes) sobre el que se construyó Occam2. JCSP nos proporciona las herramientas necesarias para dotar de la robustez que se buscaba en la programación paralela y concurrente con Java. La documentación y el software de JCSP se pueden localizar en [1] [2] [10] [21]. Gracias a JCSP se muestra todo el fundamento teórico que sirvió de base a Occam2. El alumno, por tanto, conoce su fundamento teórico y sabe que Occam2 existe.

Existen proyectos alternativos a JCSP como MPJ [6] y JMPI [13] que didácticamente pueden ser más enriquecedores ya que se incluye indirectamente MPI (interfaz de paso de mensajes).

Conceptos básicos

Cualquier asignatura en la que se use Java puede incluir, con más o menos coherencia, una introducción a la programación paralela. En estas asignaturas en las que los conocimientos de los alumnos son básicos podemos optar por usar directamente hebras con sus primitivas tales como *join()* o la API JCSP si el tiempo de que se dispone para introducir la programación paralela es más significativo. En este tipo de asignaturas (asignaturas de introducción al paralelismo como multiprocesadores) la tónica a seguir podría ser la de realizar programas con una hebra principal que inicie otras concurrentemente e introducir conceptos como el de sincronización a través de barreras (proporcionado por *join()*), etcétera. Es importante también reducir las secciones de código crítico (bloqueos para escribir en memoria) porque implicaría la introducción de monitores (concepto engorroso para este nivel). También se puede optar a construir demostraciones gráficas, usando applets, de conceptos tales como el interbloqueo, etc.

Como material didáctico a usar en este punto se ha preparado un juego de 109 transparencias que cubren los conceptos básicos necesarios para la asignatura de multiprocesadores [42].

Conceptos avanzados

Cuando el alumno, por su experiencia curricular, sea capaz de manejar conceptos complejos como control de concurrencia, paso de mensajes (adquirido en asignaturas que proporcionan conceptos básicos), etcétera, se podrán tratar problemas de actualidad en el paralelismo como intentar evitar regiones criticas, evitar dentro de lo posible las hebras tradicionales ancladas en el concepto de monitor, etc. Así en este punto se amplían y necesitan conocimientos tales como :

- Relación de la programación paralela con los sistemas operativos avanzados, en la que las primitivas de sincronización (*join(), wait(), notifyAll(),* etc.) pueden ser de ayuda para estudiar el control de la concurrencia a bajo nivel. El uso adecuado de estas construcciones puede seguirse de la explicación de la problemática que generan. Esto dará lugar a la justificación en el uso de APIs adicionales como JCSP como abstracciones que mejoran el control de la concurrencia. Dentro de este punto, otro tema a estudiar es el uso del planificador de las hebras concurrentes. La relación entre el planificador de Java y el del S.O., la relación entre la división del tiempo, prioridad y su dependencia del S.O. subyacente son aspectos que también pueden estudiarse en este contexto [13] [6] [28].
- Relacionando la computación paralela y computación distribuida (computación de altas prestaciones); asignaturas en las que poder emplear JCSP ya que proporciona un paradigma de paso de mensajes susceptible de ser usado tanto en multiprocesadores de memoria compartida como en redes estaciones de trabajo. En este tipo de asignaturas tienen cabida la computación con objetos distribuidos, movilidad de objetos, etc., con tal fin podemos usar un conjunto de librerías que permiten la computación distribuida con objetos en [7] [8] donde se entra a fondo en este proyecto, Proactive PDC. Obligada referencia merecen proyectos como los que nos facilitan "empaquetadores" Java para entornos de pasos de mensajes tan conocidos como PVM o MPI [6] [13] [28].

2.2. Redes de computadores

Java posee gran capacidad para la programación en red, la funcionalidad que nos ofrece Java para este tipo de asignaturas varía desde los sockets hasta la posibilidad de implementar protocolos de comunicaciones de manera relativamente sencilla.

Conceptos básicos

Uso de las bibliotecas de clases para desarrollar programas que accedan a la red. El primer acercamiento puede hacerse a través de los applets ya que se incrustan en páginas HTML que son muy familiares a los alumnos. El introducir los conceptos de comunicaciones con servidores a la par que se descarga el applet del servidor, se ejecuta localmente y devuelve resultados de cómputo al servidor resulta trivial [14]. A través del applet es posible introducir conceptos como el de URL, también es posible apoyarse en el empleo de applets como herramienta gráfica que simule el comportamiento de los protocolos de comunicaciones, mostrar el tránsito de información entre la capas de protocolos y su encapsulación para viajar a través de la red.

Conceptos avanzados

La programación de sockets (TCP y UDP) se ve afectada por la aparición de Java en favor del alumno ya que la sencillez y limpieza de la programación de sockets en Java frente a la programación con C/C++ es evidente, aportando ambos entornos los mismos conceptos desde el punto de vista docente.

Una manera de introducir sockets TCP de modo sencillo es la de construir un cliente usando la clase *Socket* y un servidor *ServerSocket*. Se pueden emplear servidores http sencillos y/o servidores Proxy [16] [19] [30]. Si se opta por servidores http y/o Proxy sería provechoso el introducir el concepto de hebra al explicar el modo en que los servidores atienden a los clientes. Tras TCP se puede introducir UDP con las clases *DatagramPacket* y *DatagramSocket*.

Otra mejora evidente en este campo es JavaRMI que, además de la computación en red, permite la introducción indirecta de la computación distribuida.

Aspectos como la seguridad y criptografía en redes también pueden abordarse con Java (*java.security*) para desarrollar políticas de seguridad en servidores. En la referencia [15] se pueden encontrar capítulos con ejemplos de uso del paquete. Para abordar el aspecto del procesamiento en el lado del servidor, concepto de CGI, se usa el paquete *java.Servlet* [19].

Es posible también introducir conceptos complejos como el de *marshalling/unmarshalling*[2] o el de serialización[3] a través de JavaRMI, conceptos que guardan relación con la conversión de tipos de los objetos (o datos en general) para ser transportados a través de la red desde una arquitectura a otra. Una vez introducido JavaRMI el paso para introducir el mecanismo de RPC (Remote Procedure Call) es inmediato. Con estos dos conceptos (JavaRMI y RPC) se entrega al alumno el concepto de objeto distribuido, con el que se pueden construir los conceptos de aplicaciones de dos capas y de tres capas, aplicaciones donde tiene cabida el acceso y conectividad a bases de datos. Como colofón a esta torre de conceptos que se pueden encadenar y construir con Java, podemos introducir un concepto más que une esta asignatura a la de computación distribuida y ambas con la Ingeniería del Software; es posible a través del modelo de componentes (ha de estudiarse en asignaturas de Ingeniería del software) de este modo el alumno se vería capacitado para ver la gran imagen de una aplicación que usa la red, desde la capa de aplicación donde se encuentran los componentes de negocio hasta las capas de enlace del protocolo pasando por las TCP o UDP que él mismo se ha encargado de probar con los recursos vistos anteriormente.

Con el modelo de componentes JavaBean se les puede mostrar como llegar a obtener componentes reutilizables (concepto de reutilización en Ingeniería del Software) a la par que implementan componentes con capacidad de acceso a la red. Y con los JavaBean se les puede introducir a los Enterprise JavaBeans que se caracterizan por estar en el lado del servidor y por ser extremadamente útiles en la gestión de transacciones de red (en este punto podemos establecer el nexo de unión con la conectividad a bases de datos, donde las transacciones son más que necesarias, imprescindibles).

2.3. Estructura de computadores

[2] Empaquetamiento de elementos de datos en un buffer previo a la transmisión del buffer a través de un canal de comunicaciones.

[3] Representación de una estructura de datos compleja en un formato independiente de la plataforma que la trata para poder transmitirla.

Estructura de Computadores es una asignatura que, desde nuestro punto de vista, puede beneficiarse sobremanera con el uso de Java, pero esta vez no sólo con Java como lenguaje sino también como arquitectura.

Siguiendo con la tónica general de este trabajo se pueden desarrollar recursos Java para auxiliar conceptos básicos, el profesor puede adoptar la iniciativa de crear applets que los alumnos puedan consultar, incluso desde casa, en los que observen de manera gráfica e intuitiva los movimientos de bloques de datos entre caché y memoria principal, o la eficacia de las diferentes políticas de gestión de caché, etc. En [40] encontraremos infinidad de recursos de este tipo programados en Java.

En [38] se pueden encontrar varios applets junto con el desarrollo teórico de temas cubiertos en Estructura de computadores. También es posible encontrar abstracciones tan útiles como la que se presenta en el libro [31] en el que se introduce el concepto de instrucción, arquitectura, y lenguaje ensamblador usando una máquina virtual Java o JVM abstracción tan útil que logra presentar al alumno todos los conceptos evitando que el alumno conozca arquitecturas reales que son tan complejas que se suele recurrir al uso de simuladores excesivamente sencillos y sin utilidad futura para los alumnos (PCSpim). Sin embargo IJVM (la citada abstracción) si tiene utilidad por que Java es algo que usarán en la universidad y fuera de ella. La máquina virtual java (JVM) actúa leyendo ficheros binarios CFF (class file format), editar y visualizar estos ficheros es tedioso. Se crea Oolong un ensamblador para la JVM de tal forma que es posible desarrollar programas para la JVM en ensamblador.

Conocimientos básicos

Aspectos relativos al punto flotante, little endian vs. big endian, representación interna son aspectos que se pueden analizar en clase inspeccionando la clase Float. Para la inspección del conjunto de instrucciones se puede recurrir a IJVM (conjunto de instrucciones de la máquina virtual java [31]) que reside en los ficheros de configuración de la máquina virtual. He aquí precisamente otra de las ventajas de Java, usar la maquina virtual permite inspeccionarlo todo, al contrario que ocurre con los ejecutables de PcSpim o con el hardware real.

Un tema clave en la asignatura es el ensamblador, como ya se ha mencionado la JVM incorpora la posibilidad de usar Oolong. La elección de la máquina virtual para docencia es también importante pues la inspeccionaremos a medida que se avanza en los conocimientos. Con lo que conseguimos enlazar la práctica con la teoría (ver cómo funciona la máquina y programar su ensamblador) al contrario de lo que ocurre hasta ahora, se ven conceptos teóricos en clase y en prácticas se emula un procesador o arquitectura en la que no podemos, por ejemplo, investigar cómo opera el punto flotante.

3. Conclusiones

Java puede ser un excelente recurso docente para alumnos de primeros cursos. Sobre todo para salvar conceptos tradicionalmente complejos para ellos.

Por asignaturas podemos concluir que la asignatura de multiprocesadores no experimenta ningún riesgo al emplear Java pues la dificultad de los lenguajes paralelos queda anulada, ganamos así tiempo para profundizar en conceptos. En esta asignatura es importante destacar la portabilidad de java, aspectos como la aparición de speedup siguen existiendo respecto a versiones secuenciales, y entre versiones paralelas se pueden estudiar con reimplementaciones de RMI (FastRMI,etc) y nuevas técnicas para H. P. C.

Respecto a la asignatura de Redes, se asume perder por completo la independencia que hasta ahora poseía, pero teniendo en consideración la evolución de la computación hacia Internet, es necesario incluir la red y todos sus conocimientos en relación con el resto de las asignaturas que cursan los alumnos como Bases de Datos, Ingeniería del Software, etc.

Tradicionalmente la asignatura de redes se ha impartido desde la capa física a la capa de aplicación (down-top) cuando lo natural es lo opuesto, primero capa de aplicación e ir desgajando conocimientos conforme se va descendiendo en las capas (top-down). Java se ajusta a esta última metodología, ya por un lado el alumno conoce Java y por otro lado asumir la abstracción de capa de aplicación es más sencillo que asumir la abstracción de la capa física.

Respecto a Estructura de Computadores, algunos alumnos tienen serias dificultades para entender que una máquina virtual es una “simulación de una arquitectura”, punto este que

resolvemos con trabajos y exposiciones en clase. La utilidad radica en que no aprenden ensambladores que apenas usarán como el del MIPS, aprenden un ensamblador para una máquina virtual, máquina que pueden modificar a voluntad. La gran ventaja es que pueden inspeccionar una máquina virtual simplemente manejando software y sus propios conocimientos, objetivo que no se consigue con otros simuladores.

Java es aconsejable cuando los conocimientos del alumno priman y su experiencia es escasa. Se ha citado en algún punto de este trabajo que C y Java son similares desde el punto de vista de la sintaxis por lo que el paso de uno a otro es cuanto menos inmediato. Es vital no perder las fronteras con otros lenguajes, la pseudo relación entre Java y C permite relacionar al alumno de manera relativamente rápida con otros lenguajes de programación.

4. Trabajo futuro

Invirtiendo nuestros esfuerzos en nuevas herramientas docentes como JLógica presentada en Jenui 2003 tratamos de generar entornos en los que el alumno sea el que construye su conocimiento (conocimiento constructivista) dotándoles con entorno visuales y basados en componentes en los que ellos rescriban el código e implementen las funciones necesarias, por ejemplo la fórmula R=V/I en un simulador de circuitos analógicos, para que un determinado componente funcione correctamente, en este caso una resistencia. El trabajo en curso que resume esta idea es JAnalógica, simulador visual de circuitos analógicos, con el que hacer participe al alumno en la dinámica de los circuitos.

Como muestra del esfuerzo realizado por nuestro departamento en la experimentación docente con Java sirva como ejemplo, además de JLogica y JAnalógica, la adopción de Java en la asignatura *Periféricos Avanzados* [5] [39] ligada íntimamente al sistema operativo y al hardware, en la que se ha usado una de las numerosas APIs de Java (comm) para introducir al alumno en el acceso a los puertos RS-232 e IEEE 1284 (serie y paralelo respectivamente) sin que los alumnos necesiten saber nada del mecanismo de interrupciones. Posteriormente, una vez afianzados los conocimientos, abordan una práctica de contenidos similares pero accediendo a los mismos puertos desde C.

Referencias

[1] Álvarez Macías, José L. Página programación concurrente en Java, prácticas. Universidad de Huelva.http://www.uhu.es/josel_alvarez/Prog/descargas.htm

[2] Austin, Paul. Página Web sobre JCSP. http://www.cs.ukc.ac.uk/projects/ofa/jcsp

[3] Bergin, Joseph. http://csis.pace.edu/~bergin

[4] Boroni, Christopher M., Frances W., "A Paradigm Shift! The Internet, the Web, Browsers, Java and The Future of Computer Science education", Proceedings of the Twenty-Eighth SIGCSE Technical Symposium on Computer Science Education, 1997, pág.145.

[5] Buchannan, William. PC Interfacing, communication and windows programming. Addison Wesley.

[6] Bryan Carpenter, Geoffrey Fox, Sung-Hoon Ko y Sang Lim. mpiJava 1.2: API Specification. Octubre 1999. http://grids.ucs.indiana.edu/ptliupages/projects/HPJava/reports/mpiJava-spec/mpiJava-spec.pdf

[7] Caromel, Denis, y Vayssiere, Julien. ProActive. http://www.inria.fr/sloop/javall

[8] Caromel, Denis, y Vayssiere, Julien. A Java Framework for Seamless Sequential, Multi-threaded, and Distributed Programming. ACM Workshop: Java for High-Peformance Network Computing (Stanford University, Marzo, 1998) págs. 141-150.

[9] Ceri, S. y Pelagatti, G. Distributed Databases: Systems and Principles. McGraw-Hill, 1984.

[10] Chalmers,Alan. Página web JavaPP. http://www.cs.bris.ac.uk/~alan/javapp.html

[11] Chang, D. y Harkey, D. Data Access with Java and XML. John Wiley and Sons, Inc. 1998.

[12] Christopher, Thomas W. High Performance JAVA platform computing.The Sun Microsystem Press (Java Series). Prentice Hall.

[13] Crawford,George. Descripción JMPI. http://sunsite.iisc.ernet.in/java/mpiJava/docs/mpiJava.html

[14] Decker R.y Hirshfield S.. programming.java , PWS Publishing, Boston, MA, 1998.
[15] Farley J., Java Distributed Computing, O'Reilly, Sebastopol, CA, 1998. http://www.oreilly.com/catalog/javadc/noframes.html.Códigofuente:ftp://ftp.ora.com/published/oreilly/java/javadc
[16] Flanagan D. Java Examples in a Nutshell, O'Reilly, Sebastopol, CA, 1997. http://www.oreilly.com/catalog/jenut/noframes.html.Códigofuente:http://www.oreilly.com/catalog/jenut/examples
[17] Gosling,Joy. Steele. The Java Language Specification, Sun Microsystems, 1998.
[18] Hamilton, G., Cattell, R., y Fisher, M. JDBC Database Access with Java: A Tutorial and Annotated Reference. ACM Press, 1997.
[19] Harold, E.R. Java Network Programming, O'Reilly, Sebastopol, CA, 1997.http://www.oreilly.com/catalog/javanetwk/noframes.html.Codigo,fuente:ftp://ftp.ora.com/pub/examples/java/java.netprog
[20] Hartley, Stephen. Concurrent Programming: The Java Programming Language, Oxford: Oxford University Press, 1998.
[21] Herderink, Gerald, Página JavaPP en la Universidad de Twente. http://www.ce.utwente.nl/
[22] Hoare, C.A. Monitors: an operating system structuring concept. CACM, (Octubre, 1974) 17(10):549-557.
[23] Horstman, C. y Cornell, G. Core Java 1.1 Volume II: Advanced Features (3rd ed.)
[24] Jepson, B. Java Database Programming. John Wiley and Sons, Inc. 1996.
[25] Larman, Gary, Applying UML and Patterns, Prentice-Hall, 1997.
[26] Lea, Doug. Concurrent Programming in Java. Nueva York: ACM Press, 1996.
[27] Lewis, Ted. The Binary Critic. IEEE Computer (Marzo, 1997), 17(3):134.
[28] Mitchell,Christine.JavaPVM.http://www.chmsr.gatech.edu/jPVM/
[29] Orfali R. y Harkey D. Client Server Programming with Java and CORBA (2nd Ed.). John Wiley and Sons, Inc. 1998.
[30] Sun Microsystems. Java Tutorial, Networking subtutorial, Javasoft, 1998. http://www.javasoft.com/docs/books/tutorial/networking/index.html
[31] Tanenbaum, Andrew S. Organizacion de computadoras, un enfoque estructurado. Cuarta Edición. Pearson Educación. págs 218 en adelante.
[32] Ullman, J. Principles of Database and Knowledge-Base Systems Vol.1. Computer Science Press, 1988.
[33] Welch, Peter. Java Threads in the Light of Occam/CSP. En A. Bakkers, editor, Parallel Programming and Java, Proceedings of WoTUG 20, v 50 of Concurrent Systems Engineering, Amsterdam:IOS Press, 1997.
[34] http://edlab-www.cs.umass.edu/~lerner
[35] http://java.sun.com/docs/books/tutorial/essential/exceptions
[36] http://www.artima.com/flexiblejava
[37] http://www.caip.rutgers.edu/~marsic/Teaching/ISE.html
[38] http://www.math.luc.edu/~laufer/courses/473
[39] http://www.ace.ual.es/~jroca/docencia/pr/perifericos/programa.html
[40] http://www.cs.iastate.edu/~prabhu/Tutorial/title.html
[41] High Performance Java Computing http://www.cs.ucsb.edu/conferences/java98/papers/hpjavampi.pdf
[42] Material didáctico para Multiprocesadores. http://www.ace.ual.es/~jaberme/multiprocesadores/transparencias.ppt