//NOVEMBER/DECEMBER 2014 /

Java™ magazine

By and for the Java community

THE INTERNET OF THINGS

JAVA IS EVERYWHERE

ORACLE

EDITORIAL

Editor in Chief
Caroline Kvitka

Community Editor
Yolande Poirier

Java in Action Editor
Michelle Kovac

Technology Editor
Tori Wieldt

Contributing Writer
Kevin Farnham

Contributing Editors
Claire Breen, Blair Campbell, Kay Keppler, Karen Perkins

DESIGN

Senior Creative Director
Francisco G Delgadillo

Senior Design Director
Suemi Lam

Design Director
Richard Merchán

Contributing Designers
Jaime Ferrand, Diane Murray, Arianna Pucherelli

Production Designers
Sheila Brennan, Kathy Cygnarowicz

PUBLISHING

Publisher
Jennifer Hamilton +1.650.506.3794

Associate Publisher and Audience Development Director
Karin Kinnear +1.650.506.1985

ADVERTISING SALES

President, Sprocket Media
Kyle Walkenhorst +1.323.340.8585

Western and Central US, LAD, and Canada, Sprocket Media
Tom Cometa +1.510.339.2403

Eastern US and EMEA/APAC, Sprocket Media
Mark Makinney +1.805.709.4745

Advertising Sales Assistant
Cindy Elhaj +1.626.396.9400 x 201

Mailing-List Rentals
Contact your sales representative.

RESOURCES

Oracle Products
+1.800.367.8674 (US/Canada)

Oracle Services
+1.888.283.0591 (US)

Oracle Press Books
oraclepressbooks.com

ARTICLE SUBMISSION

If you are interested in submitting an article, please e-mail the editors.

SUBSCRIPTION INFORMATION

Subscriptions are complimentary for qualified individuals who complete the subscription form.

MAGAZINE CUSTOMER SERVICE

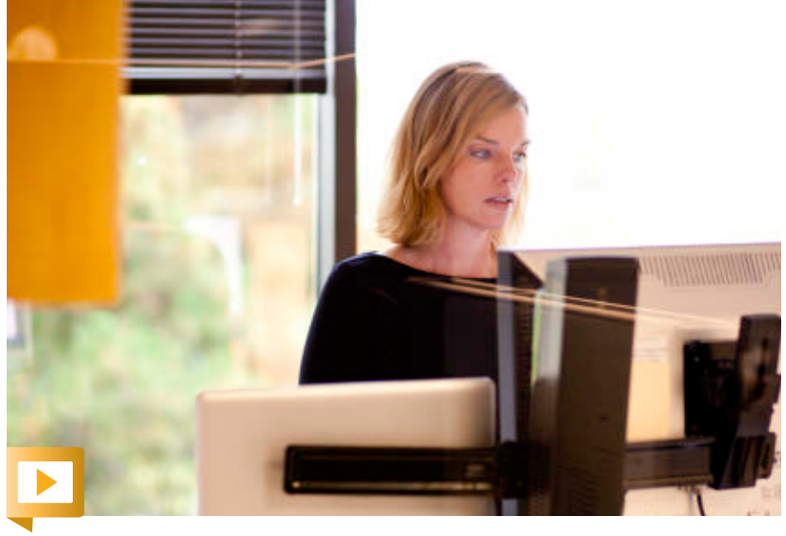
java@halldata.com **Phone** +1.847.763.9635

PRIVACY

Oracle Publishing allows sharing of its mailing list with selected third parties. If you prefer that your mailing address or e-mail address not be included in this program, contact Customer Service.

Java Magazine is published bimonthly with a free subscription price by Oracle, 500 Oracle Parkway, MS OPL-3C, Redwood City, CA 94065-1600.

Digital Publishing by GTxcel



Java is everywhere. As the Internet of Things (IoT) moves from hype to reality, we're seeing embedded Java used in a wide range of applications, from industrial automation systems and medical imaging devices to connected vehicles and smart meters. Haven't you heard? Java is the glue.

Internet of Things

In our interview with Oracle's Henrik Ståhl, we discuss the opportunities and challenges that the IoT presents for Java developers, and how changes in Java SE and Java ME (and their embedded versions) make it easier to reassemble and strip down code for smaller devices.

We have lots of additional IoT content in this issue as well: we talk to Freescale's Maulin Patel about the IoT and Java, profile IoT Developer Challenge winner Lhings Connected Table, and show you how robots make factories smarter. Plus, Vinicius Senger introduces the Device I/O API, Kai Kreuzer brings us the latest on smart homes, and Michael Kölling shows us how to program in Java *on* the Raspberry Pi.

We also take a look at the developers of tomorrow in "Java: the Next Generation." I've been talking to kids at various programming events for the last few months, and I am inspired. These kids are brave, bold, and so smart. They do not fear technology; they embrace it. I'm thrilled to see so many programs around the world that are teaching young people to code and helping them to create *their* futures.

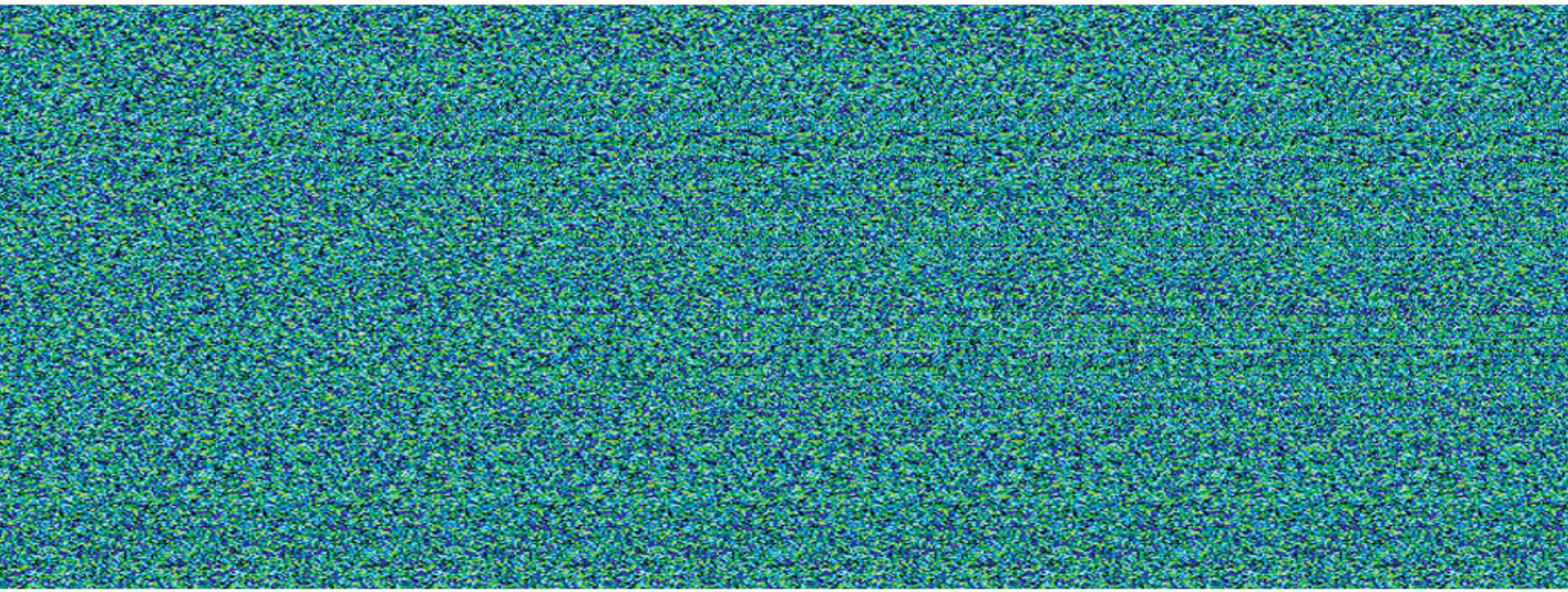
Caroline Kvitka, Editor in Chief BIO

//send us your feedback /

We'll review all suggestions for future improvements. Depending on volume, some messages might not get a direct reply.

PHOTOGRAPH BY BOB ADLER

The answer is right in front of you



Java Image Enabling SDKs that Help You See the Big Picture

At first glance it may seem difficult, but it's really quite simple. Atalasoft's JoltImage product is a proven SDK for image enabling your Java-based web applications, easily. Image enabling helps to add dimension to your data, so you can uncover insights such as correlations and causations hidden inside your 2-dimensional documents. Our SDK does the heavy lifting for you, saving time, money, and the headaches of figuring it out yourself. Backed by our highly knowledgeable & caffeinated support engineers, JoltImage will enable your success and make the big picture so much easier to see.

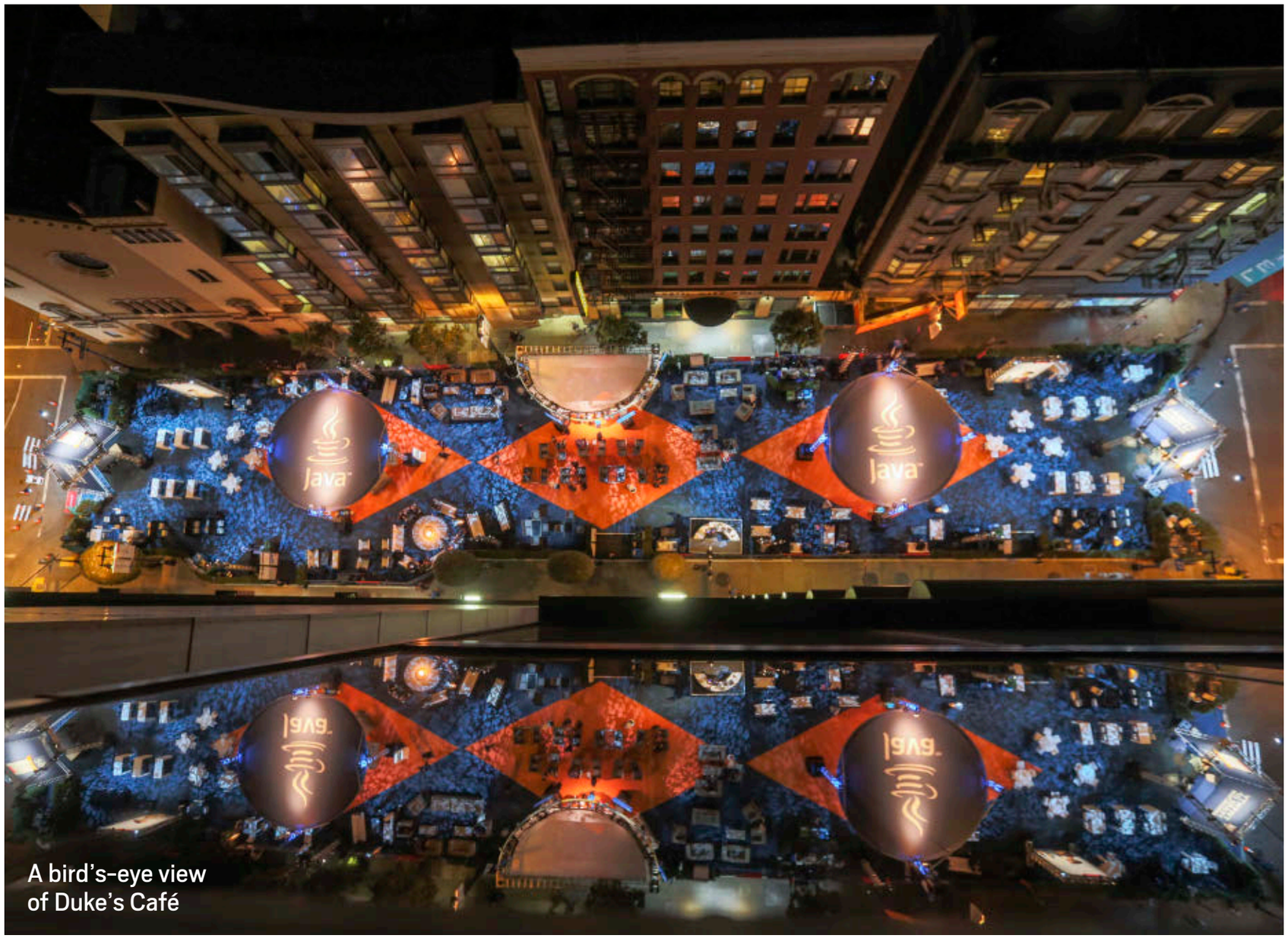
Atalasoft JoltImage

Click for tips on viewing the stereogram

Image enabling experts & bacon connoisseurs. Visit us online to see our full line of SDK products for .NET and Java

Atalasoft
A Kofax Company
www.atalasoft.com

JAVAONE TAKES SAN FRANCISCO



A bird's-eye view of Duke's Café

PHOTOGRAPH BY HARTMANN STUDIOS

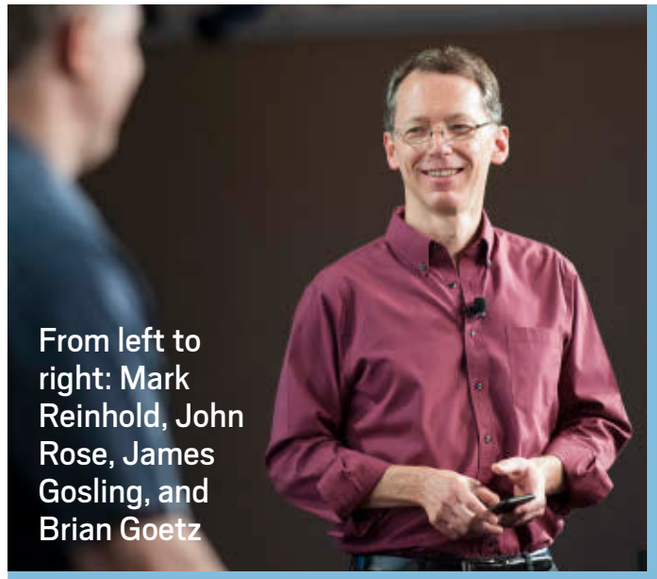
The 19th JavaOne celebrated momentum in the Java platform, innovation, and strong community. Devoxx4Kids kicked things off a day early, when 150 kids got hands-on and had fun with programming, robotics, and engineering. The opening Strategy keynote established Oracle's commitment to building a strong and flexible Java platform that will enable IT professionals to adapt to rapid technological change, from the Internet of Things (IoT) to more-powerful enterprise systems.

Oracle's **Peter Utzschneider** said Java has shown strong growth this year, with 80 new Java user groups (JUGs) established and a transparent and accessible Java Community Process (JCP). Java SE 8, released in March 2014, has changed the landscape for Java developers, thanks to lambda expressions and their ability to reduce the need for boilerplate code, said Oracle's **Georges Saab**. Utzschneider stressed that the ongoing integration of embedded Java with the rest of the platform is enabling developers who have basic

Watch the Strategy keynote.

Java skills to use those skills in the embedded space and the growing IoT.

Later, Oracle's **Cameron Purdy** emphasized how the Java platform is continuously evolving toward end-to-end capabilities. He acknowledged the help of the community and emphasized that Java EE 7, the only standards-based platform for enterprise development, is continuing to drive down the costs of building and maintaining applications. He pointed out that the JCP Executive Committee has unanimously approved the roadmap to Java EE 8—which was drawn up directly from the survey responses of 4,500 Java developers. Purdy observed that for Java EE 7, 19 JUGs made valuable contributions by adopting 14 JSRs, and he called for that kind of community participation going forward.



From left to right: Mark Reinhold, John Rose, James Gosling, and Brian Goetz

TECHNICAL KEYNOTE

The Technical keynote was held in two parts: one that followed the Strategy keynote and a second that preceded the Community keynote. Oracle's **Mark Reinhold** reflected on Java as it nears its 20th birthday about ways it can keep going for another 20 years. He addressed the challenges of enhancing Java's ability to scale down, improve security, evolve more easily, improve startup performance, and scale up to large systems. To address these issues, Java 9 is focused on Project Jigsaw, which is "an attempt to design and implement a standard module system for the Java SE platform and use that to modularize the platform itself—and applications," he said. Jigsaw will have a profound impact on how large Java systems are built and maintained and improve developer performance and productivity, Reinhold said.

Next, **Brian Goetz** offered a vision of Java extending to Java 9 and beyond that would include value classes. Look to Project Valhalla and Project Panama for more information, he said.

In part two, Reinhold gathered a panel of Java luminaries, including **James Gosling**, onstage to take questions. When asked when Java would become obsolete, Gosling confessed that for a decade he has been expecting Java's demise, but that Java is a kind of organism grounded in the community that is well understood and flexible and has strong staying power.

Watch the Technical keynote.

PHOTOGRAPHS BY HARTMANN STUDIOS

COMMUNITY KEYNOTE

At the JavaOne Community keynote, Java Evangelist James Weaver gathered a large group of innovators, who showed off Java-based projects—everything from programs that teach kids to code to automated vehicle testing systems that can save lives.

Robotics was a key theme. Some highlights:

Andra Kay, director at Silicon Valley Robotics, said, "By 2020 your household robot will be your house."

Bruno Maisonnier, CEO at Aldebaran, a world leader in humanoid robots, presented a video showing robots teaching children mathematics in schools.

Paul Perrone of Perrone Robotics lamented the 30,000 deaths from auto accidents each year in the United States, and showed a video about his automated vehicle testing system with an advanced braking system that could save lives—a first step toward cars with full autonomy.

The Community keynote ended with the traditional T-shirt toss (or catapult), led by **James Gosling**. It was the perfect ending to a great week of information sharing, learning, and community building.

Watch the Community keynote.

PHOTOGRAPHS BY HARTMANN STUDIOS

NullPointers Rock JavaOne

The NullPointers, the unofficial Java community band, rocked JavaOne at the JCP party and later in the week at Duke's Café. The band is made up entirely of JavaOne community members.

GEEKS RIDE ON

The Geek Bike Ride tradition continued at JavaOne with 35 riders from nine countries. Riders sported jerseys designed by community members in Brazil and enjoyed a sunny day of heart-pumping activity.

BIKE RIDE PHOTOGRAPH BY KEVIN NILSON; JCP PHOTOGRAPHS BY BOB LARSEN

JCP Awards

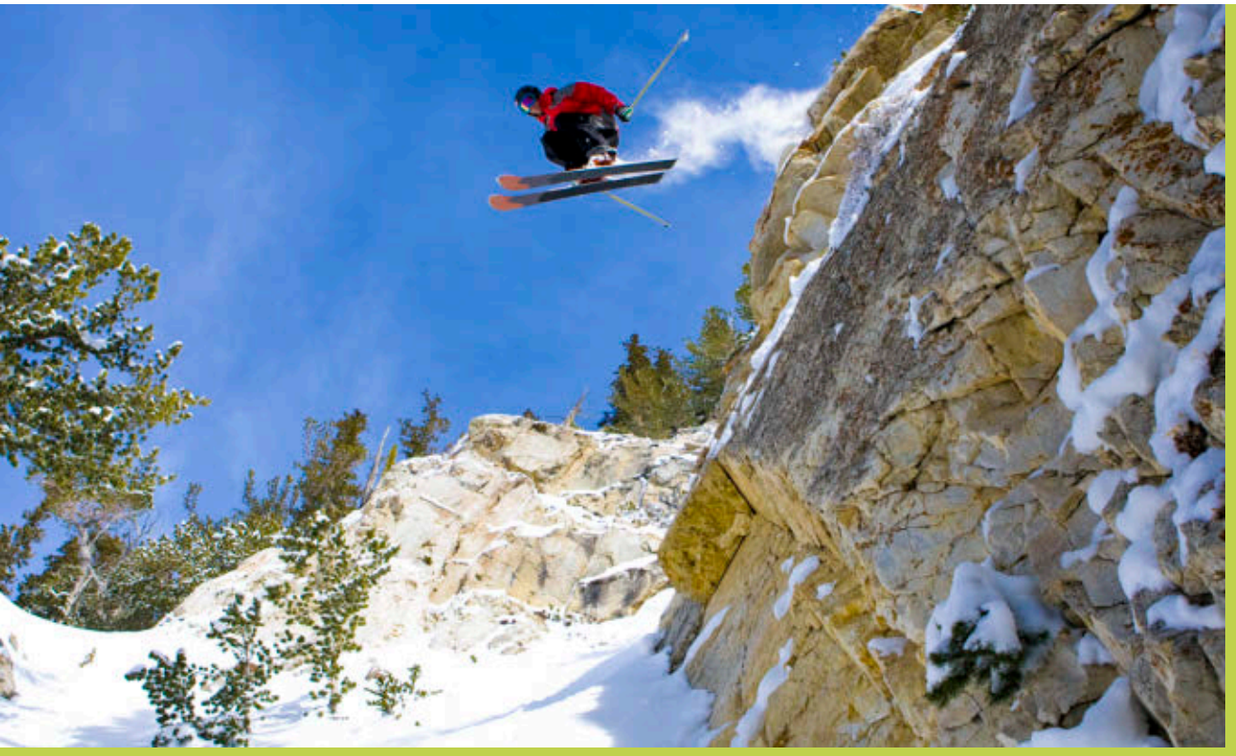
Otávio Gonçalves de Santana (top), Heather VanCura (top and bottom), and Michael Lagally (bottom)

The Java Community Process (JCP) presented the 10th annual JCP Awards and celebrated its 15th birthday at a gathering atop the Hilton Hotel on Monday night.

Heather VanCura received the award for JCP Program Member of the Year for her leadership in the Adopt-a-JSR program, which provides a mechanism for Java user groups (JUGs) and individuals to easily contribute to JSRs and encourages grassroots participation in crafting the future of Java.

The Most Significant JSR was awarded to JSR 360, Connected Limited Device Configuration (CLDC) 8, which brings language features, including generics, enumerations, and try-with-resources, from Java SE into Java ME. **Michael Lagally** was recognized as Outstanding Spec Lead for his efforts in spearheading JSR 360.

Otávio Gonçalves de Santana was awarded Outstanding Adopt-a-JSR Participant for his work on JSR 354, Java Money and Concurrency—specifically in migrating the codebase from Java 7 to Java 8.



FEATURED JAVA USER GROUP

UTAH JAVA USERS GROUP

The Utah Java Users Group (UJUG) was founded in 1998 by **Ben Galbraith** and **Glen Lewis**. The group is currently led by **Jason Porter** and supported by board members **Don Bogardus**, **Mike Bylund**, and **Chris Hansen**. UJUG currently has about 1,000 members. Meetings, which are normally held on the third Thursday of each month, typically draw from 80 to 180 attendees.

If this level of participation surprises you (because Utah is in the bottom 20 percent of the United States in population density), that probably means you haven't heard of "Silicon Slopes" (@SiliconSlopes), the cluster of information technology and software development firms along the Wasatch Front in north central Utah that is "leading the world in innovative memory process technology. . .and data analysis," according to CNBC.

Porter notes that, indeed, the Silicon Slopes phenomenon has had an enormously positive effect on UJUG's growth and vivacity. "Silicon Slopes has seen a tech boom, and Java is the language of choice in all the major shops, and most of the smaller ones," he says. "This makes for great local presenters, and we also get visits from out-of-towners, including **Jason Van Zyl**, **Gavin King**, and **Venkat Subramaniam**."

Each UJUG meeting has two presentations, and the presenter schedule is already filled into late 2015.

Meetings typically start with the traditional pizza, and they often end with a trip to the local pub for those interested. "Yes, there's good beer in Utah!" Porter exclaims. He adds that each UJUG meeting is unique. "Something new is tried each month to improve networking and for just plain fun—we had a paper airplane competition last month."

If you're a Java developer who loves skiing or the great outdoors, you may want to consider the Silicon Slopes next time you're seeking new work. "There's a healthy startup community and a trend for big out-of-state shops to open local development centers," says Porter. "Developer unemployment in Utah is down to 1.5 percent, with salaries continuing to rise (the latest trend is highway recruiting billboards). All the energy in the ecosystem makes for great UJUG meetings, and a tangible excitement in the community."

Learn more at UJUG's Meetup site.

SKIING PHOTOGRAPH BY MIKE SCHIRF/GETTY IMAGES

Java SE Updates

Oracle has released Java SE Development Kit 8 Update 25 (JDK 8u25) and Java SE 7u71 and Java SE 7u72.

JDK 8u25 includes important security fixes. Read the release notes and download the update.

Java SE 7u71 and Java SE 7u72 include important security fixes. Read the release notes and download the update.

Java SE 7u71 is a Critical Patch Update (CPU), which contains fixes to security vulnerabilities and critical bugs. Java SE 7u72 is a Patch Set Update (PSU), which contains all of the security fixes in the CPUs released up to that version, as well as additional noncritical fixes. Learn more about CPU and PSU releases.

JAVA CHAMPION PROFILE

BRUNO GHISI

***Bruno Ghisi** is an open source developer, technology blogger, and cofounder of Resultados Digitais, an online marketing company. He became a Java Champion in September 2007, while he was completing his undergraduate degree at the Federal University of Santa Catarina, Brazil.*

Java Magazine: Where did you grow up?

Ghisi: Florianópolis, a southern Brazil island.

Java Magazine: When and how did you first become interested in computers and programming?

Ghisi: My father was involved in the first computer science class at the Federal University of Santa Catarina, and introduced me to mainframes at his job when I was a child.

Java Magazine: What was your first computer and programming language?

Ghisi: My first computer was a 486 PC. I put Linux on it, and started programming simple scripts. My first real programming language was Java.

Java Magazine: What was your first professional programming job?

Ghisi: I worked for a mobile company that was writing some nice Java ME applications in a period when there were no smartphones or app stores. The UIs were simple, and networking was terrible. We ported some of the first social networks such as Orkut (before Facebook) and Fotolog to mobile devices.

Java Magazine: What do you enjoy for fun and relaxation?

Ghisi: I love programming and still do it as a hobby. I used to surf, but now I enjoy jogging and cooking more.

Java Magazine: What happens on your typical day off work?

Ghisi: I definitely stay with my friends and family, sometimes going to the beach. I love barbecues, and I also like to travel.

Java Magazine: What side effects of your career do you enjoy the most?

Ghisi: I meet a lot of people that I've long admired.

Java Magazine: Has being a Java Champion changed anything for you with respect to your daily life?

Ghisi: Definitely. I was pretty young when I became a Java Champion, and being one helped to guide my career and take it to the next level. Great opportunities also came from blogging and doing open source development.

Java Magazine: What are you looking forward to in the coming years?

Ghisi: For the past three years, I have been CTO [and cofounder] of Resultados Digitais. We have a product called RD Station, which is an online marketing platform, and more than 1,200 clients. I love what I do. I work with a fantastic team, and we have awesome customers. For the coming years, I hope to keep learning, experience new challenges, and probably have a baby as well.

You can find Bruno Ghisi on Twitter as @brunogh.

EVENTS

Jfokus *FEBRUARY 2–4*
STOCKHOLM, SWEDEN

Jfokus has run for eight years and is the largest annual Java developer conference in Sweden. Conference topics include Java SE and Java EE, front end and web, mobile, continuous delivery and DevOps, Internet of Things, cloud and big data, future and trends, alternative Java Virtual Machine (JVM) languages, and agile development.

PHOTOGRAPH BY EDWARD STOJAKOVIC/FLICKR

Codemotion

NOVEMBER 21–22
MADRID, SPAIN
This conference is open to users of all languages and platforms. It offers full-day workshops, keynotes, conference sessions, and hackathons across eight tracks.

Groovy & Grails eXchange

DECEMBER 11–12
LONDON, ENGLAND
This two-day conference brings together industry-leading experts and developers from around the world to learn and share everything about the Groovy and Grails ecosystems.

Riga Dev Day

JANUARY 22, 2015
RIGA, LATVIA
This event is a joint project by Google Developer Group Riga, Java User Group Latvia, and Oracle User Group Latvia. By and for software developers, it focuses on 25 of the most-relevant topics and technologies for that audience. Tracks include JVM and web development, databases, DevOps, and case studies.

JAVA BOOKS

SOA WITH JAVA

By Thomas Erl, Andre Tost, Satadru Roy, Philip Thomas, Raj Balasubramanian, David Chou, and Thomas Plunkett
Prentice Hall, June 2014

Java has evolved into an exceptional platform for building web-based enterprise services. In *SOA with Java*, Thomas Erl and several world-class experts guide readers in mastering the principles, best practices, and Java technologies required to design and deliver high-value services and service-oriented solutions. The book is appropriate for any technical professional whose aim is to design and implement web-based service-oriented architectures and enterprise solutions with Java technologies.

MASTERING LAMBDAS: JAVA PROGRAMMING IN A MULTICORE WORLD

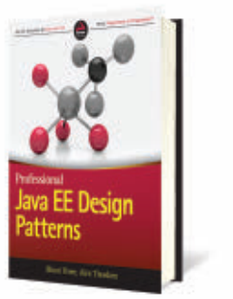
By Maurice Naftalin
Oracle Press, November 2014

This Oracle Press book provides a complete grounding in the theory of lambda expressions and the rationale for their introduction and use in practice. It reviews a simple Java code fragment iterated over a collection, poses different challenges, and then delves into the fundamentals of syntax and basic use. Maurice Naftalin recommends best practices for adding lambda expressions to an existing library and explores the changing Collections Framework.

RESTFUL JAVA WEB SERVICES SECURITY

By René Enríquez and Andrés Salazar C.
Packt Publishing, July 2014

This book will serve as a practical companion for you to learn about common vulnerabilities when using RESTful services. You will gain indispensable knowledge of the tools you can use to implement and test security on your applications. *RESTful Java Web Services Security* covers the finer details of setting up RESTful services—for example, implementing RESTEasy, securing transmission protocols such as OAuth, and integrating OAuth with RESTEasy.



PROFESSIONAL JAVA EE DESIGN PATTERNS

By Murat Yener and Alex Theedom
Wiley, December 2014

Professional Java EE Design Patterns is the perfect companion for anyone who wants to work more effectively with Java EE. It's the only resource that covers both the theory and application of design patterns in solving real-world problems. The authors guide readers through the fundamental and advanced features of Java EE 7, presenting patterns throughout and demonstrating how they are used in day-to-day problem solving.

JAVA EE APPLICATIONS ON ORACLE JAVA CLOUD

By Harshad Oak
Oracle Press, September 2014

Learn how to build highly available, scalable, secure, distributed applications on Oracle Java Cloud with Java Champion Harshad Oak. He leads you through the entire Java EE cloud-based application lifecycle, from development to deployment. Filled with real-world examples, ready-to-use code, and best practices, *Java EE Applications on Oracle Java Cloud* is an invaluable resource for anyone looking to meet the growing demand for cloud-based development skills.

JAVA DEVELOPMENT FOR THE INTERNET OF THINGS

Oracle's **Henrik Ståhl** discusses the Internet of Things for Java developers.

BY TIMOTHY BENEKE

Internet of Things

The Internet of Things (IoT) is poised to blow into our daily lives and make many of the things we interact with more efficient, faster, and more fun. Analysts predict that by 2020 there will be 50 billion devices wirelessly connected to each other and making smart decisions for us. We're seeing early adoption in healthcare, including lifestyle health devices, patient monitoring, and home healthcare. Another big area is telematics in the automotive industry—not to mention home automation. Suffice it to say that everything from our homes to healthcare to the workplace will change due to the IoT.

For Java developers, many opportunities and questions present themselves. How can they deploy their present skills to better apply them to IoT development? What changes

in the Java platform should future IoT developers know about? What are the security issues?

We met up with Henrik Ståhl, vice president of product management for Java and the IoT at Oracle, to discuss these and other issues.

Java Magazine: What is Oracle doing to help Java developers enter the IoT space?

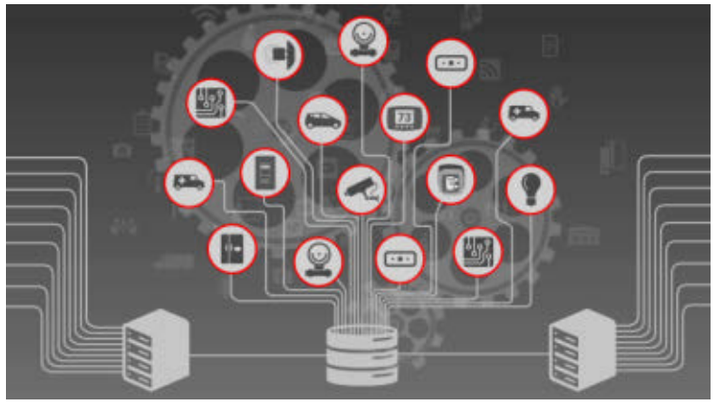
Ståhl: Java developers can already use their skills to develop back-end applications for the IoT. We want them to also be able to develop software on the various devices that constitute the devices—the *things* in the IoT—using the same language, the same APIs, and compact runtimes suitable for the embedded space, which in the past has involved a steep learning curve for everyone who ventures into it. Instead, developers have had to learn low-level languages that impede productivity and lead to relatively high maintenance costs. Now, because embedded development often includes integration with various types of specialized peripherals and sensors, it will likely always require some amount of low-level programming. We are not aiming to replace that, but rather to enable an application developer to work in Java, freeing up the system developer to focus only on the hardware enablement. This distinction does not currently exist in the embedded space.

To achieve this vision, it is vital that we get to a common programming model across all devices, regardless of size or device capabilities. With the release of Oracle Java ME Embedded 8 and Oracle Java SE Embedded 8, we have made the features that developers are used to in Java available on embedded platforms with as little as a couple of hundred kilobytes of memory. A major goal of these two releases was to make them as compatible with each other—and with the mainstream Oracle JDK 8—as possible.

To get toward this goal while achieving low memory and disk footprints, we have introduced the concept of Compact Profiles in Java SE 8. These give developers a choice of deploying the entire set of Java SE APIs, or one of three different subsets that are incrementally smaller. These subsets have been selected very carefully to match what APIs are needed for common use

Henrik Ståhl at work at Oracle's Stockholm, Sweden, office

Ståhl (center) shows a demo to colleagues Klara Ward and Erik Helin.



Watch the video: *Java Embedded for IoT*

cases such as running an OSGi [Open Service Gateway initiative] stack. The Compact Profiles are a stepping-stone toward more-granular modularity that will be introduced with Project Jigsaw in Java SE 9.

As for Oracle Java ME Embedded 8, we have come a very long way from its roots as a platform for feature phones. The APIs and the application deployment model have been massively updated to fit their target to run on small embedded devices, including medium- to high-end microcontrollers, smart sensors, and small embedded gateways. This brings the Java ME APIs much closer, but not quite on par, with Java SE. We plan to continue this development and are looking at things missing from Java ME such as lambdas, reflection, and better integration with native code.

These embedded Java platforms are the number one thing that we are providing to embedded developers.

And as we include tools—for example, on the Java ME side so that it catches up with the likes of Java SE—it will be really easy for the 9 million Java developers to start taking on embedded projects, which is something that we are obviously trying to encourage. We have seen very good uptake of Java already in the higher-end devices, such as wireless modules (small chips that integrate a low-power CPU, some memory, and a 3/4G modem for communications) and larger SoCs [systems on a chip] including the popular hobbyist-targeted Raspberry Pi. Our traction in the microcontroller space is more limited to date, but with the release of Oracle Java ME Embedded 8 we believe that our platform is well suited to meet market demands. And so, we think that with enough energy and support coming from the Java community and Oracle, we can get Java developers to start focusing on embedded. This brings an immediate benefit to the IoT because Java greatly facilitates key device-side requirements such as communications, integration, security, and a flexible application model. The notion of having billions of things out there on constrained form factors means that we need programming models to push business logic to the network edge and make decisions at the perimeter of the network as opposed to centralizing all communication behind the customer firewall and the cloud. So, the embedded Java platform is only going to

DEVELOPING ON DEVICES

"Java developers can already use their skills to develop back-end applications for the IoT. We want them to also be able to develop software on the various devices that constitute the devices in the IoT."

become more important moving forward, as Java developers start to key in on the IoT.

Java Magazine: What's happening with Java ME?

Ståhl: We released Java ME 8 and we are planning another release in 2015, where you will see a lot of incremental enhancements that build on the capabilities of the Java ME 8 platform. You're going to see additional platform support, improved runtime performance—all of the things that we're focusing on or that we have focused on in Java SE 8. You are also going to see uptake in the field by hardware manufacturers that integrate Java ME directly in their devices, greatly increasing the adoption rate.

Java Magazine: What do you want Java developers to understand as they apply their skills in an embedded context?

Ståhl: With our embedded platforms, Oracle is enabling all of the applications in the Oracle portfolio and all of the applications being built by the Java community to talk to the IoT. So if you have a retail application, you can now extend that retail application to iBeacon and do proximity marketing. If you have an application that does input payment or serves up ads for automobile customers, you can now extend that all the way to the automobile and actually start getting back information from its various systems that will allow you to offer a pretty good maintenance plan.

And if you want to better orchestrate the subsystems in a building, you can now do that by tying in to the HVAC system and the lighting system, and doing that across vendors with a standard platform such as Java. The ability to connect to edge devices—for example, an actuator, a valve, or whatever, depending on the use case—and control these systems is one of the key enablers of the IoT.

I am quick to caution that it's not necessarily the biggest value proposition. It's necessary but insufficient to derive value from the use cases. It's really the rest of the application that becomes more valuable. For example, a retail application—from Oracle or from somewhere else—becomes much more valuable when you can use it to do proximity-based marketing or provide proximity-based offers depending on where somebody is in a store.

So embedded Java becomes the enabling technology. For developers who pay attention, this will allow them to tap into an entirely new class of appli-

Ståhl talks with colleague Stefan Särne.

cations that extend existing solutions.

Java Magazine: What key features of Java make it a good fit for the IoT?

Ståhl: What's important about Java is the abstraction and portability that the runtime environment provides. Organizations that need to deploy devices as part of an IoT solution can decouple their applications and libraries from specific devices, enabling reuse of code between different devices and between different generations of the same device. This has been hugely beneficial to Java for server applications. But it is even more important for embedded because there is such a massive diversity of hardware architectures, operating systems, peripherals, and so on. The simple benefit of being able to reuse application code is in itself a very strong argument for Java. Of course, the rich set of standard APIs and the predictable development cycle and lifetime of the Java platform itself are also strong arguments.

Another benefit of the runtime environment is encapsulation. The fact that we're running within the JVM [Java Virtual Machine] and restricting the developer from doing things that are not performed via some well-defined API means that we're providing an additional layer of security. In the IoT space, security repeatedly comes up as the underlying limiting concern that people have about jumping into this industry.

Java Magazine: Tell us more about security and the IoT.

Ståhl: The IoT introduces risk on a number of levels. There are three different approaches that we're taking, and we're relying on standards wherever possible to make sure we provide a proven solution.

First, there is a vulnerability on the device itself. So, for example, if you had a vending machine to hook up to the internet and you could use that to access other systems, then hardening that system at the edge is hugely important. This is further aggravated by the fact that the devices are often not physically secure and—even worse—in some cases, they are under the physical control of the entity most interested in hacking into them. I often use smart meters as an example: The person in whose house the smart meter is installed is the person who benefits most from hacking it.

Second, the middle piece of security is data transmission, where we are relying on internet standards; encrypting things on the edge and then on the back end; and sending the data over the encrypted channel. With traditional internet security, we have a concern that there's a man in the middle who's going to intercept something and replay it as an attack. With the IoT, the concern is more that you think you're talking to a smart toaster in your house when really you're talking to a hacker in his mom's

Ståhl notes that in the IoT space, security repeatedly comes up as an underlying limiting concern.

Ståhl says that Oracle is redoubling its efforts to keep Java vibrant and is continuing to add new productivity features.

basement. That's the big concern.

Third is data security. This goes back to basic concepts such as privacy, data integrity, and accountability for and auditability of access to sensitive data. If you carry around a medical sensor, which reads your vitals, you don't want data from it to be available to anyone except your healthcare provider and your close friends and family. With today's complex infrastructures and business models, how can you make sure that data read from a sensor ends up in the right back-end system, that it has not been modified or tampered with on the way, and that only authorized individuals have access to it?

Of course, all of these areas have to be managed such that an administrator—or, in the case of personal data, an individual—can control the security configuration, the authentication, and the authorization through policies.

Java Magazine: Any final thoughts for Java developers?

Ståhl: Oracle is redoubling its efforts to keep Java vibrant and is continuing to honor its commitment to add new productivity features—for example, things such as lambda expressions—across the board, including in the embedded computing space, down the road. It's a big commitment. But in this age where there are a ton of new languages being created, and a ton of new runtimes that are popping up all over the place, the reality is that our biggest customers always come back to Java. And the reason that's happening is because of the continued investment that we make in Java from a performance and security standpoint, and that is not going to change. Oracle's belief is that by continuing to invest in these areas, Java will remain vibrant, the community will remain engaged with Oracle, and Oracle and the community will continue to have a creative and beneficial ongoing relationship. I would like to finally put to rest the FUD [fear, uncertainty, and doubt] about Java being a legacy language and platform. The reality is that Java is rejuvenating itself across the board with new innovation and new growth in community activity. This includes the addition of new Java user groups, and an unprecedented rate of adoption of Java 8 including in hot areas such as the cloud and, yes, the IoT. Developers have the tools and a commitment from Oracle and the community to do everything possible to make them successful. `</article>`

MORE ON TOPIC:

Internet of Things

Timothy Beneke is a freelance writer and editor who has written extensively about Java. His articles, which cover a wide range of topics, have appeared in *Mother Jones* and the *East Bay Express*.

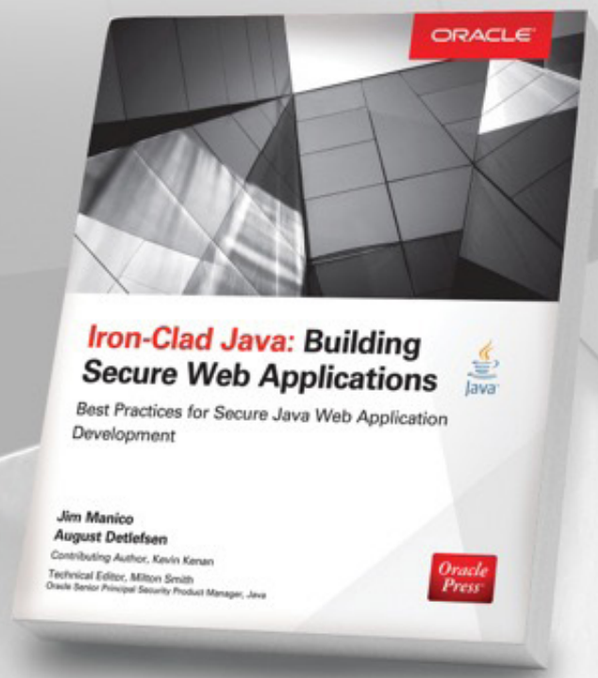
LEARN MORE

- "Navigate the Internet of Things"

Oracle Press™

Your Destination for Java Expertise

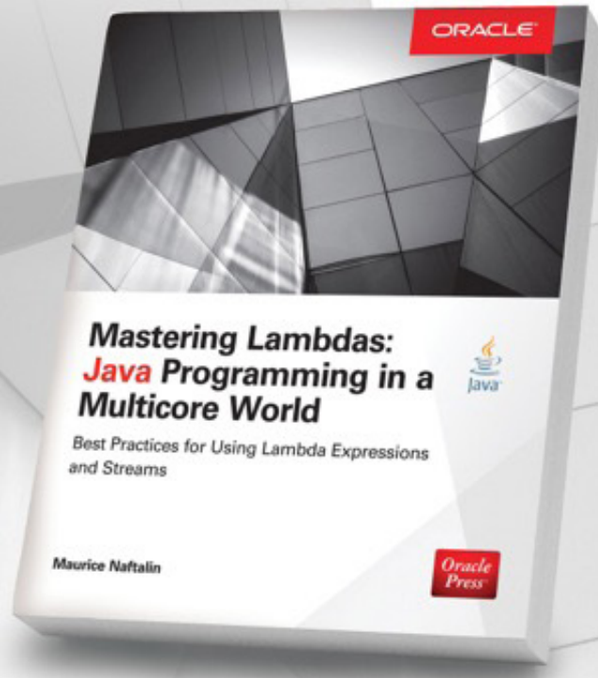
Written by leading Java experts, Oracle Press books offer the most definitive, complete, and up-to-date coverage of Java available.



Iron-Clad Java: Building Secure Web Applications

Jim Manico, August Detlefsen

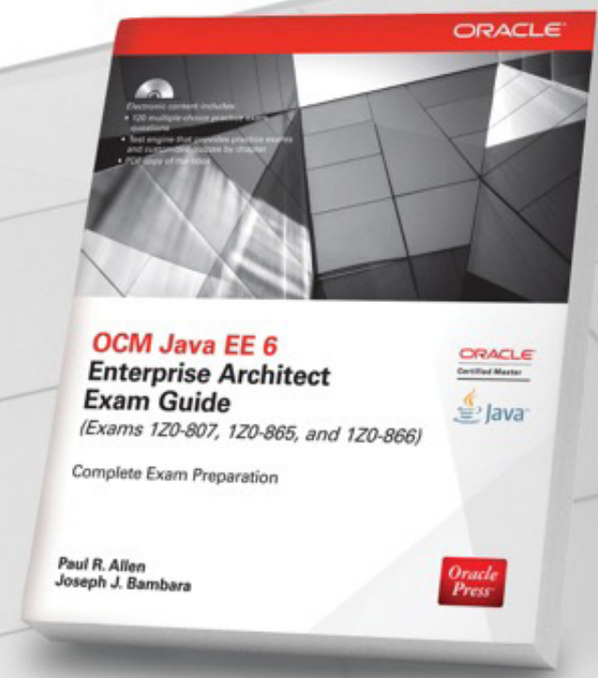
Develop, deploy, and maintain secure Java applications using the expert techniques and open source libraries in this Oracle Press guide.



Mastering Lambdas: Java Programming in a Multicore World

Maurice Naftalin

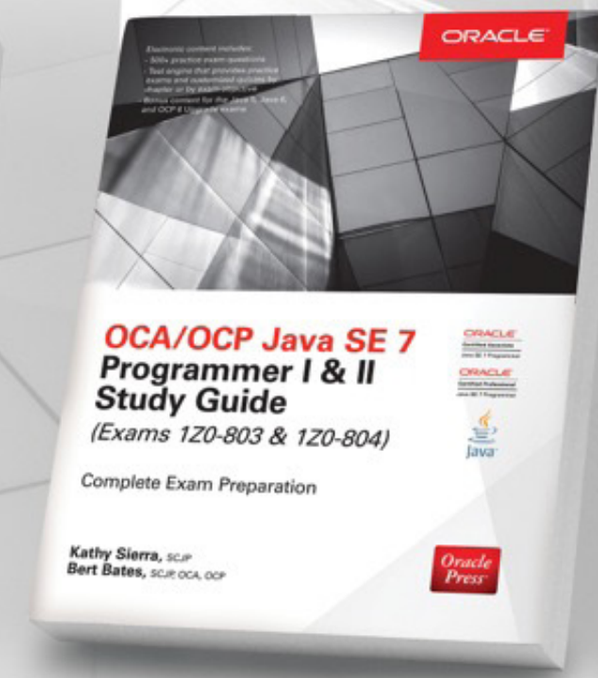
Effectively use lambda expressions to maximize performance improvements provided by multicore hardware.



OCM Java EE 6 Enterprise Architect Exam Guide

Paul Allen, Joseph Bambara

Get complete details on Oracle Certified Master Java EE 6 Enterprise Architect Exams 1Z0-807, 1Z0-865, and 1Z0-866. 120 multiple-choice practice exam questions are included.



OCA/OCP Java SE 7 Programmer I & II Study Guide

Kathy Sierra, Bert Bates

This self-study tool offers full coverage of OCA/OCP Exams 1Z0-803 and 1Z0-804. Electronic content includes 500 practice exam questions.

Oracle Press™ 20 YEARS

Available in print and as eBooks

Visit us at OraclePressBooks.com • Follow us @OraclePress and Like us OraclePress

At Java programming workshops at Oracle headquarters and JavaOne, kids got hands-on with coding, robotics, and more.

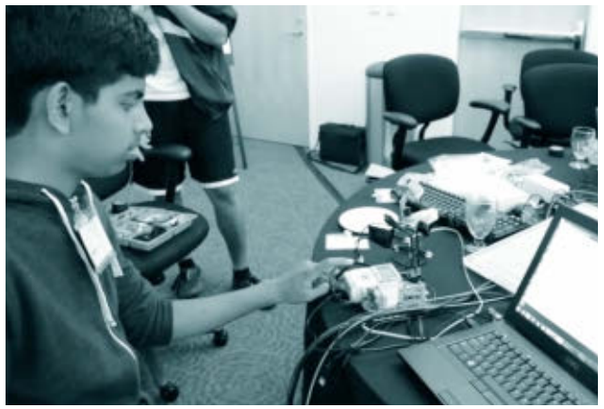
JAVA: THE NEXT GENERATION

Teach kids to code and give them tools for success. **BY CAROLINE KVITKA**

PHOTOGRAPHS BY ORACLE/RON SELLERS, HARTMANN STUDIOS

Kids are coding. They are creating animations, modifying Minecraft, building LEGO Mindstorms, programming NAO robots, and more. And they are doing it in Java.

With the number of computer science jobs on the rise, making sure the next generation can fill these positions is critical. According to the organization code.org, there will be 1 million more computer science jobs than computer



Kids at the Create the Future Java Workshop talk with Caroline Kvitka.

At the Devoxx4Kids event before JavaOne, kids chose from workshops in programming, robotics, and engineering.

PHOTOGRAPH BY HARTMANN STUDIOS

science students in the United States alone by 2020.

Whether kids are destined for careers in computer science or something outside of technology, programming skills are useful in many parts of their lives, says Wanda Dann, associate professor at Carnegie Mellon University and director of the Alice program. Alice is a tool that focuses on storytelling in a 3-D environment to teach kids the basic concepts of object-oriented programming.

"Learning to program actually is learning to decompose a problem into simpler steps," Dann says. "And by accomplishing those steps, one accomplishes the entire solution to the problem."

However, only 1 in 10 schools in the United States offers computer science classes, according to code.org, which wants US schools to follow the example set by China, the UK, and Vietnam, where coding classes are offered as early as elementary school.

That's where extracurricular programs come in. Although opportunities for kids to learn to code were limited just a few years ago, today programs abound and kids (and sometimes their parents) are flocking to them.

In August, 672 kids between the ages of 13 and 18 came to Oracle headquarters in Redwood Shores, California, for the Create the Future Java Workshop. Over three days, they learned to program in Alice 2.0 and Alice 3.0, and in Java on LEGO Mindstorms.

Alice 2.0 focuses on logical and computational thinking skills and programming fundamentals, while Alice 3.0 emphasizes object-oriented concepts and a full transition to the Java programming language. LEGO Mindstorms combines building LEGO robots with programming.

"We have used Java as our tool for implementation, and we write our code so that after students have used Alice they can go into a classroom

where Java is being taught and feel comfortable that they understand what those statements do because they actually have seen them before," Dann says. "I think one of the reasons that Java is so well liked as a programming language for teaching younger students is because it is possible to model everyday kinds of objects. Students can then relate to those objects, which makes it easier for them to see how the skills they're learning in computer programming are usable in many different career paths," she adds.

More recently, the next generation of Java developers got a taste of programming at a Devoxx4Kids day the Saturday before JavaOne San Francisco. At this event, a collaboration with Oracle Academy, 150 kids, ages 10 to 18, got hands-on and had fun with programming, robotics, and engineering. Topics included Alice, Arduino, Greenfoot, LEGO Mindstorms, Minecraft Modding, NAO humanoid robot, Python, Raspberry Pi gaming, and Scratch. Kids attended four sessions of their choice.

"Teaching children how to program must be a priority in a society where technology is becoming more and more important," says Daniel de Luca, worldwide manager of the Devoxx4Kids initiative, which started in 2012 in Belgium with programming workshops for kids. The program aims to teach and inspire kids about computer programming while they are having fun. Since its founding, Devoxx4Kids has shared its curriculum with Java user groups and other organizations around the world. To date, more than 80 Devoxx4Kids workshops have taken place, with 2,500 participants.

"We need to train our kids in technology, and have them stay engaged in technology at an early age. If we catch them raw and show them it's fun, it's possible, they won't be scared," adds Arun Gupta of Devoxx4Kids Bay Area.

And from the looks of the San Francisco event, the kids did have fun. "It's been really cool . . . I really liked it," says Tim Gonzales, a 14-year-old participant from San Francisco who plans to pursue a career in technology. "The door to opportunity is *so* open. I just need to find out my passion within technology." Programs such as

Cool Tools for Kids

Scratch
A drag-and-drop programming tool, Scratch teaches young kids how to program interactive stories, games, and animations.

Greenfoot
This visual and interactive integrated development environment (IDE) teaches object orientation with Java.

BlueJ
This standalone Java IDE is used to teach the beginnings of programming.

SnapCode
Featured at the JavaOne Community keynote, SnapCode is a new, free tool for building Java applications with simple drag and drop.

PHOTOGRAPH BY HARTMANN STUDIOS

Twenty-one girls from the organization Black Girls Code participated in the Devoxx4Kids event at JavaOne.

Left: Kids took a stretching break at the Devoxx4Kids event. Below: Chris Hollinger of Oracle Academy talked about Alice at the Create the Future Java Workshop.

Devoxx4Kids can help him get where he needs to go, he adds.

Of the Devoxx4Kids attendees, 21 were girls from the San Francisco chapter of Black Girls Code, an organization that seeks to introduce girls from underserved communities to computer science and technology with a goal to increase their representation in the STEM (science, technology, engineering, and mathematics) fields. "The event was a wonderful opportunity for our students to experience a wide array of technology tools in a youth-friendly atmosphere," says Black Girls Code Founder Kimberly Bryant.

AROUND THE WORLD

Computer programs for kids are popping up all over the world. In the Philippines, JEDI4KiDS educates children to be more creative using computers and provides them the opportunity to learn computer programming in a fun and interactive manner.

JEDI4KiDS is part of the larger JEDI initiative, which provides free and open source computer science and IT

Kids and organizers talk about their experience at the Devoxx4Kids event.

PHOTOGRAPHS BY HARTMANN STUDIOS, ORACLE/RON SELLERS

More Organizations

Girls Who Code
This US-based nonprofit organization works to educate, inspire, and equip high school girls with the skills and resources to pursue opportunities in computing fields.

CoderDojo
CoderDojo is a global network of volunteer-led, independent, free computer programming clubs for young people between 7 and 17 who learn how to code; develop websites, apps, programs, and games; and explore technology.

Maker Kids
The Toronto, Canada–based Maker Kids builds maker learning activities for kids and educators and runs one of the world's only "makerspaces" for kids. Curriculum and training are available to educators, parents, and interested adults in any location.

Hour of Code
This global programming movement is reaching tens of millions of students in every country. Hour of Code takes place during Computer Science Education Week, December 8 to 14, 2014. Hour of Code offers one-hour tutorials in more than 30 languages for ages 4 and up.

PHOTOGRAPH BY ORACLE/RON SELLERS

Daniel de Luca, worldwide manager of Devoxx4Kids (top row, center, in white), and Arun Gupta of Devoxx4Kids Bay Area (top row, second from right) with Devoxx4Kids day participants, volunteers, and NAO robots.

training to colleges and universities. JEDI4KiDS was developed when the JEDI founders realized that there was a need to teach programming at an early age. The program offers one-day workshops in Greenfoot, LEGO Mindstorms, and Scratch in partnership with Devoxx4Kids.

"Coding allows children to think creatively and use their brains in a way they have never imagined," says Gerald Concha, founding partner of JEDI4KiDS. "It allows them to feel confident that they can solve problems. If there is something they don't like, they can fix or change it. If there is something they wish they had, they can create it."

The momentum around exposing more kids to computer technology opens the door to a bright future—both for Java and the next generation of Java developers, technologists, and creative minds. `</article>`

Caroline Kvitka is editor in chief of *Java Magazine*.

Maulin Patel with team members Greg Hemstreet and Mary Thomas at Freescale's Austin, Texas, headquarters.

JCP Executive Series

The Java Advantage for IoT

Freescale's Maulin Patel discusses the Internet of Things (IoT) and how the JCP helps to facilitate evolving technologies. **BY STEVE MELOAN**

PHOTOGRAPHY BY PAUL S. HOWELL

Internet of Things

Maulin Patel manages the global Software and Applications team for the Microcontrollers group at Freescale, a leader in embedded processing solutions. He oversees the engineering focus of Freescale's IoT strategy. With

more than 25 years of experience in software engineering, he has held various leadership roles at IBM, Intel, NXP Semiconductors, Conexant Systems, and Trident Microsystems. Patel earned a bachelor's degree in electrical engineering from SP University in India and a master's degree in electrical engineering from Kansas State University. In addition, he holds 11 patents in a variety of software fields such as system management, reliability, availability, and serviceability.

Java Magazine: What motivated you to join the JCP [Java Community Process], and how has that involvement benefited your company and your work within the company?
Patel: For more than two years we've been engaged with Oracle through the JCP. That partnership has been very fruitful toward ensuring that our products work seamlessly with a variety of industry partners in the IoT space.

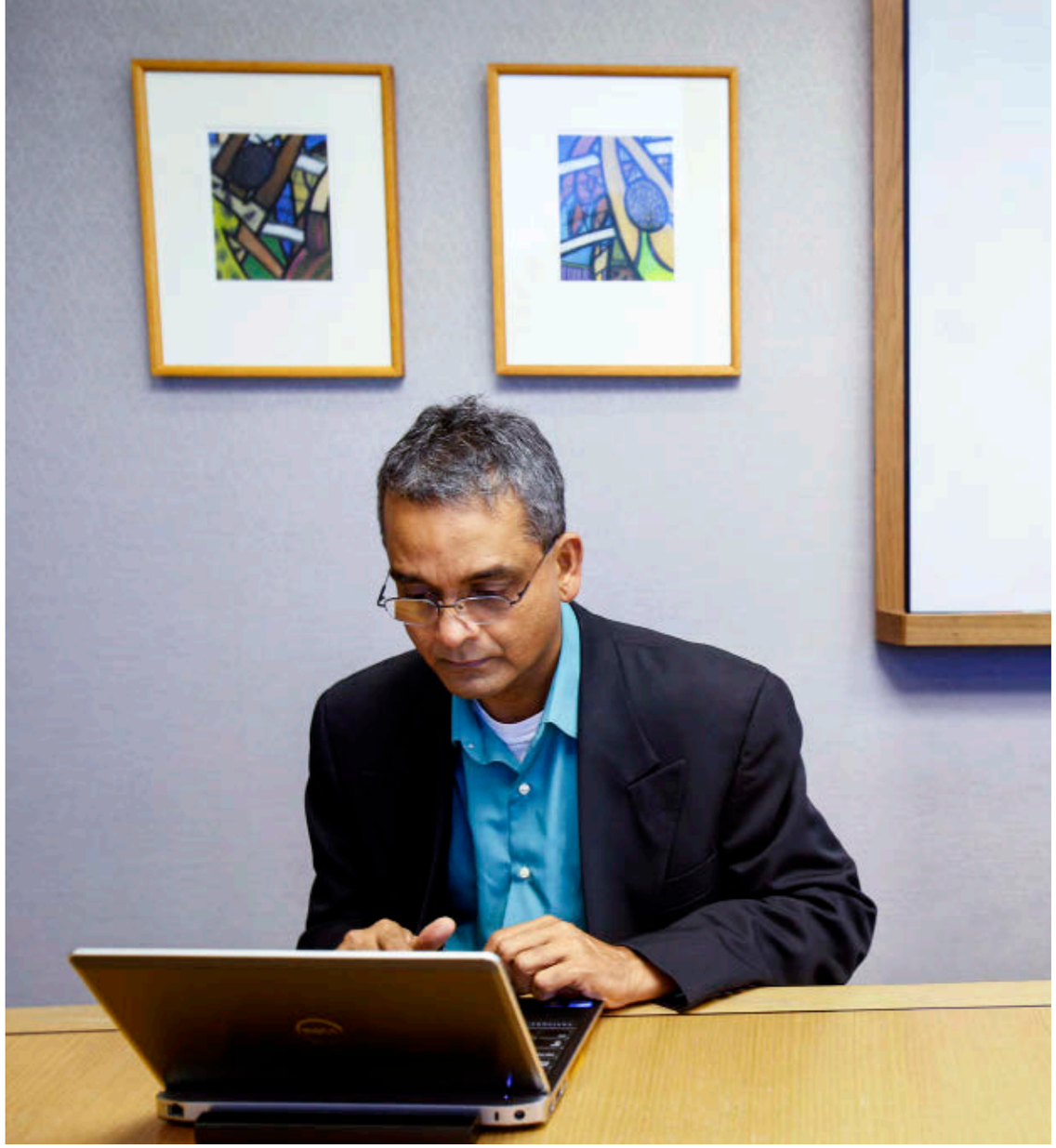
Java provides a framework for our end-to-end IoT schema, consisting of edge (sensor endpoints), gateway (communications layer), and cloud. Our portfolio plays in all of these segments.

One of the primary motivating factors in joining the JCP has been to contribute our IoT expertise and experience along with other industry leaders. Our goal is to ensure that Java evolves in a way that is beneficial to our area of focus, while maintaining the broad-based applicability that has defined the Java ecosystem.

Since joining the JCP, we've been able to engage with technology partners toward influencing key elements of Java architectural JSRs. We were also influential in some of the JSRs related to Java 8.
Java Magazine: Can you give us some more detail on your involvement with various JSRs?
Patel: I have contributed to JSR 358 and JSR 364 as an Executive Committee member. JSR 358 focuses on a number of important changes and adjustments to the JCP, including revision of the Java Specification Participation Agreement [JSPA], which is crucial to broadening participation.

The focus of JSR 364 is to broaden JCP participation by defining new membership classes, modifying existing membership categories, and ensuring appropriate intellectual property commitments from JCP members.

Executive Committee member discussions represent a broad range of voices, including large companies such as IBM, SAP, Intel, HP, and Oracle, as well as a host of smaller players. I believe we arrive at compromises and creative solutions that best suit the entire community.
Java Magazine: What are some of your most pressing areas of concern in evolving the IoT space and Java?
Patel: Key areas of focus for us are the footprint, the power scheme management, and data security across the entire domain, end to end. Security issues are of paramount importance at this stage of the IoT's development,



Patel says that "Java provides a framework for our end-to-end IoT schema."

PATEL ON SECURITY

"Security issues are of paramount importance at this stage of the IoT's development, specifically the interplay between Java and the IoT framework."

Patel says that the Java architecture allows customers to create vertical applications that span data collection at edge nodes, to gateways, and up to the cloud.

specifically the interplay between Java and the IoT framework. The entire Java community needs to become more aware of IoT issues, because growth is going to be substantial. And the whole Java ecosystem will be strengthened by adding additional effort and attention to these areas.

Java Magazine: What are some of the advantages Java brings to the table for IoT developers?

Patel: The Java architecture allows any customer to create vertical applications that span data collection at the edge nodes, to the gateway communication layer, and up to the cloud, using a consistent architecture that all plays together seamlessly. That's a huge advantage. You can begin data collection with Java ME, move to Java SE for gateway services, and move to Java EE for data management and processing. This frees Java IoT developers from worrying about hardware dependencies and allows them to create applications much more quickly.

The Java platform also ensures data security across the entire vertical span. It is built in as an integrated architectural attribute. With all the data hacking we constantly hear about, security is on people's minds. Java security integrity will continue to evolve through the JSR process. It's a very dynamic set of activities.

Java Magazine: Are there any changes you would like to see in the JCP?

Patel: This topic was addressed at the last meeting of the Executive Committee. We were discussing ways to bring new voices from the IoT perspective into the Executive Committee through the election process.

As a functional body, the JCP is working well. But we need to make participation easier for individuals who may not be affiliated with large corporations. Many of these developers have brilliant IoT and Java-related ideas and expertise to contribute. Also, Java user groups span the globe. We need to incentivize this amazing pool of talent to bring more JSRs to the table.

The JCP is still a bit complex and time consuming. JSRs 348, 358, and 364 have improved things greatly, but we're constantly trying to improve efficiency and inclusiveness, while preserving the integrity of the processes.

Java Magazine: What are your thoughts about attracting existing Java developers who are not very familiar with the IoT space?

Patel: That's a good question, and something that should be a topic of conversation within the Executive Committee. Expert Java developers will want to become involved in the IoT domain, because it's one of the most exciting application areas right

now. And it's poised for explosive growth. But I do think the importance of IoT should be reinforced at events such as JavaOne and Oracle OpenWorld, with keynote addresses and sessions. That would be a great forum for this kind of promotion.

I would compare IoT right now to the initial phase of the smartphone app market. There was some initial resistance, and nobody knew exactly how things would evolve. But apps very quickly became a multi-billion-dollar market. I believe that will happen for IoT as well.

Java Magazine: Could you discuss the interplay between standards and innovation in the world of Java and the JCP?

Patel: Too much rigidity can be the enemy of innovation. But we also need well-crafted standards to attract developers around a common framework. Millions of developers worldwide are affected by the standards developed through the JCP. The platform must function as an integrated totality. And it does. But it must also be flexible enough and nimble enough to respond to dynamic market forces. We talk about that frequently within the JCP community. And I think we're generally doing a good job of achieving those goals.

PATEL ON THE JVM

"In the world of IoT development, there is sometimes a concern that the JVM will be an impediment to efficiency. But evolving Java standards are making Java ME applications very efficient."

In the world of IoT development, there is sometimes a concern that the JVM [Java Virtual Machine] will be an impediment to efficiency. But evolving Java standards are making Java ME applications very efficient.

It's also important to consider overall system cost. This is where the Java platform excels. Java development saves time and promotes reusability. These are huge advantages in the broader scheme.

Java Magazine: Do you have any closing remarks?

Patel: If we deploy IoT to the lowest common denominator in the human population, that means the household. For instance, if you were driving to work and you had forgotten to set your home alarm, you could pull over and do that on your smartphone. And you could check surveillance cameras to make sure all was well, and then adjust the thermostat or close the garage door. Those myriad devices must be seamlessly connected all the way to the cloud.

There are still connectivity issues to be dealt with in making this happen. A common connectivity framework and architecture will be essential. Multiple connectivity standards are out there—Wi-Fi, Bluetooth, and ZigBee, for instance. Wi-Fi is becoming the most prevalent, but it's not the de facto standard. Connectivity needs to be ubiquitous, highly reliable, and well organized.

And as I've mentioned, security is another concern of primary importance. Customers must be able to trust that their data is secure for IoT applications to really proliferate and become a part of daily life. The evolution of Java through the JCP will be a key element in that success formula. `</article>`

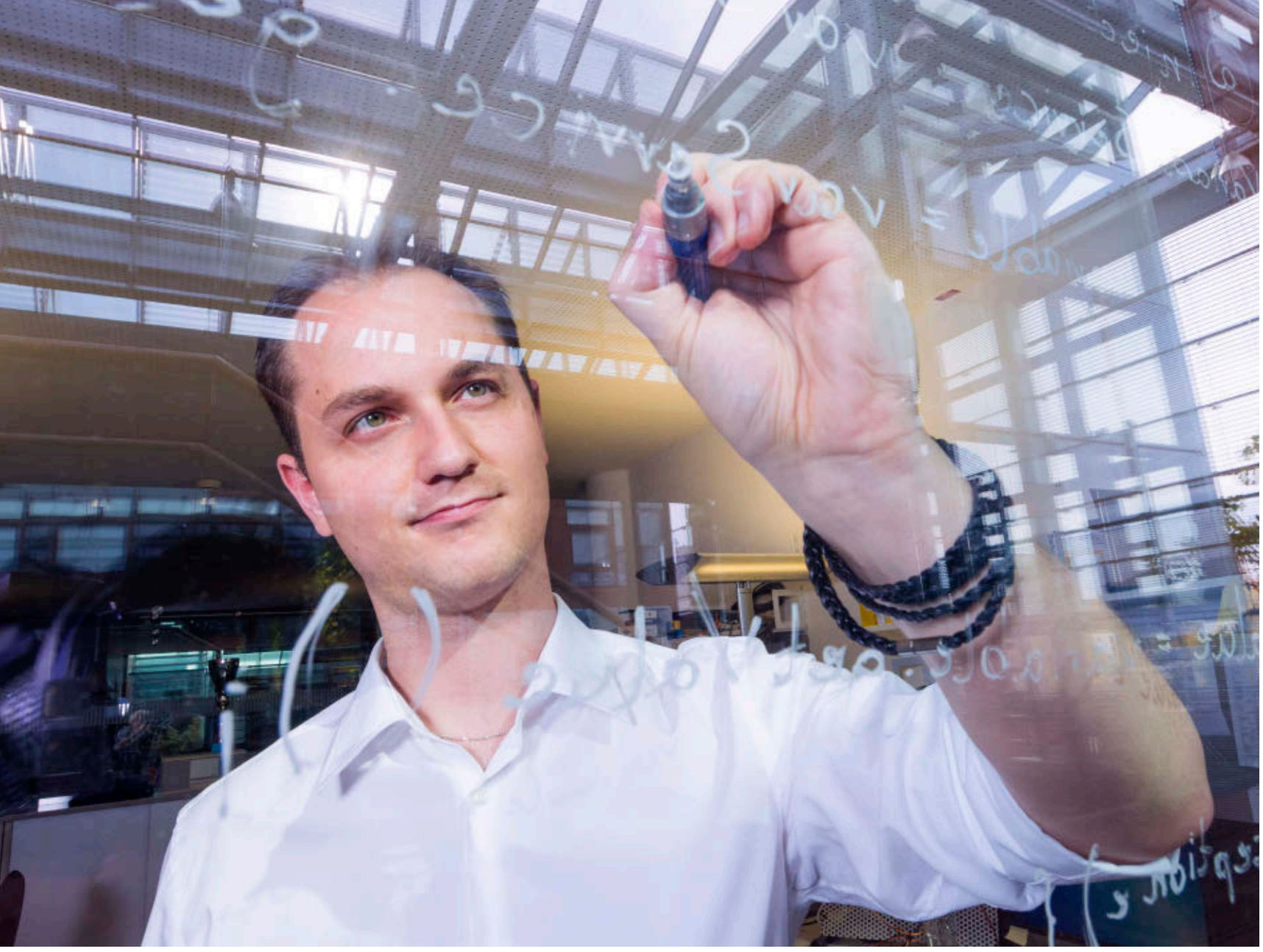
MORE ON TOPIC:

Internet of Things

Steve Meloan is a former C/UNIX software developer who has covered the web and the internet for such publications as *Wired*, *Rolling Stone*, *Playboy*, *SF Weekly*, and the *San Francisco Examiner*. He recently published a science-adventure novel, *The Shroud*, and regularly contributes to *The Huffington Post*.

LEARN MORE

- Freescale's Internet of Tomorrow Tour



Keba Software Developer Hannes Bachmayr sketches out some code at company headquarters in Linz, Austria.

ROBOTS MAKE FACTORIES SMARTER

Keba's systems help usher in the next industrial revolution. **BY DAVID BAUM**

PHOTOGRAPHY BY TON HENDRIKS

Internet of Things

Few industries have seen as much change and upheaval in the last decade as the manufacturing sector. Today's modern factories bear little resemblance to the mills, plants, and assembly lines that defined industrial production throughout the twentieth century. Steady advancements in technology and increasing globalization have transformed not only our factories, but our manufacturing workforce as well. Take a tour of a modern automobile plant, for example, and you will see more robots than people. Routine production and assembly operations are performed by computer-driven machines. The factory workers who remain are mostly highly skilled technicians, designers, and engineers.

Many of these machines are equipped with advanced sensors that give them an uncanny awareness of their operating environments, as part of

KEBA AG
keba.com

Headquarters:
Linz, Austria

Employees:
900

Revenue:
€181 million

Java technologies used:
Java SE 6, Java SE 7, and Java SE 8 with JavaFX, OSGi, and Eclipse e4

Keba software engineering team members (from left to right) Hannes Bachmayr, Florian Erler, Martin Fischer, Fabian Schöppl, and Robert Pöchtrager share ideas.

the rapidly growing Internet of Things (IoT)—physical objects with network connectivity that can send and receive data. In a modern factory, sensors not only guide the machines, but also provide information to fine-tune the operation as a whole. For example, if the heat or humidity on a production line varies outside of an acceptable range, that line can report this condition and adjust its activities automatically.

The intellectual capital in these versatile factory environments doesn't lie so much with the equipment as with the software that controls it—a pervasive fabric of instructions that binds each robot to the global supply chain. Collecting and analyzing real-time production data dictates many aspects of the manufacturing cycle, from sourcing materials and customizing products to interacting with partners.

Keba AG, an engineering firm based in Linz, Austria, that designs and produces high-tech equipment for the industrial, energy, and banking sectors, is in the vanguard of these modern factory developments. In the field of industrial automation, Keba has built products for controlling injection molding machines and robotic systems. In the energy sector it is well known for its climate control systems and elec-

tric vehicle charging stations. Keba also develops bank terminals, parcel machines, lottery terminals, and custom hardware and software systems governing their use.

For the last several years, Keba's software engineers have been immersed in the development of a Java-based environment for human-machine interfaces (HMIs) that operate the activities of all types of industrial equipment. One popular system, dubbed KeStudio View Edit, is a point-and-click environment that guides engineers as they develop HMIs for robots and other systems. Application engineers of Keba use KeStudio to design everything from motion controllers to keyless security systems.

"We are the guys setting up the software that enables people to interact with machines," says Hannes Bachmayr, a software developer at Keba specializing in these human-machine connections. "Many of our important software components are based on Java."

From left to right: Fabian Schöppl, Marion Lindenmair, Hannes Bachmayr, and Martin Fischer chat over coffee at Keba.

GLOBAL CONNECTIONS, LOCAL CONTROL

According to a report issued by the McKinsey Global Institute, the expense of owning and operating industrial robots has fallen by as much as 50 percent compared to human labor since 1990. This is due in part to sheer efficiency: Machines can operate around the clock and achieve unusually high precision in production and assembly processes. They don't talk back or require a coffee break, and many of them can work in a lights-out environment, saving energy costs.

Many of these machines are controlled by HMI applications that dictate how they behave, such as positioning a robotic arm in an injection molding machine or training an infrared sensor to follow a heat source. KeStudio makes it easy to create HMI applications by shielding product designers and hardware engineers from the underlying code.

"KeStudio is easy to use because you don't need to understand programming," says Fabian Schöppl, a software architect in Keba's industrial department who works closely with Bachmayr. "All of the controls are based on Java. When a user clicks Generate, KeStudio creates the Java code that controls the machines. And when they click the Start button, KeStudio fires up Java applications that govern a robot's control, sensory feedback, and information-processing capabilities."

Some of Keba's products are powered by multipurpose hardware devices such as the CP-36 system KeControl C3. This versatile platform can be programmed to control 3-D printers, presses, and other machine tools using the KeStudio environment. KeControl C3 is powered by an Intel processor and runs Windows, Linux, and other operating systems. "Java is perfect for this multiplatform system because we can compile applications once and run them everywhere," Bachmayr says. "Java also makes it easy to move applications from one computing environment to another."

To ensure that Keba's hardware and software environments are compatible with global manufacturing standards, Keba's software engineers pay close attention to the proceedings of the OPC Foundation, an industry consortium chartered with ensuring compatibility among various types of industrial equipment. Adhering to OPC (which stands for *object linking and embedding for process control*) standards is especially important for the work Keba does with German automotive manufactur-

ers, many of which are part of Industrie 4.0, a project sponsored by the German government to encourage the creation of "smart factories." An important standards body for Industrie 4.0 and the IoT, OPC helps ensure compatible communication among industrial equipment and processes.

DAWN OF AN INDUSTRIAL ERA

Manufacturing consultants at Germany Trade & Invest, a government agency within the Federal Republic of Germany, believe Industrie 4.0 will pave the way for a fourth industrial revolution over the next couple of decades. As they see it, the first industrial revolution, which extended from the late eighteenth century to the mid-nineteenth century, was defined by the shift from handcrafted goods to mechanical production methods. The second industrial revolution gained momentum at the start of the twentieth century with refined methods for industrial assembly and the mass production of products. Industrial revolution #3 started in the late 1960s with the rise of electronics and the consequent infusion of information technology into industrial processes.

From Industry 1.0 to 4.0

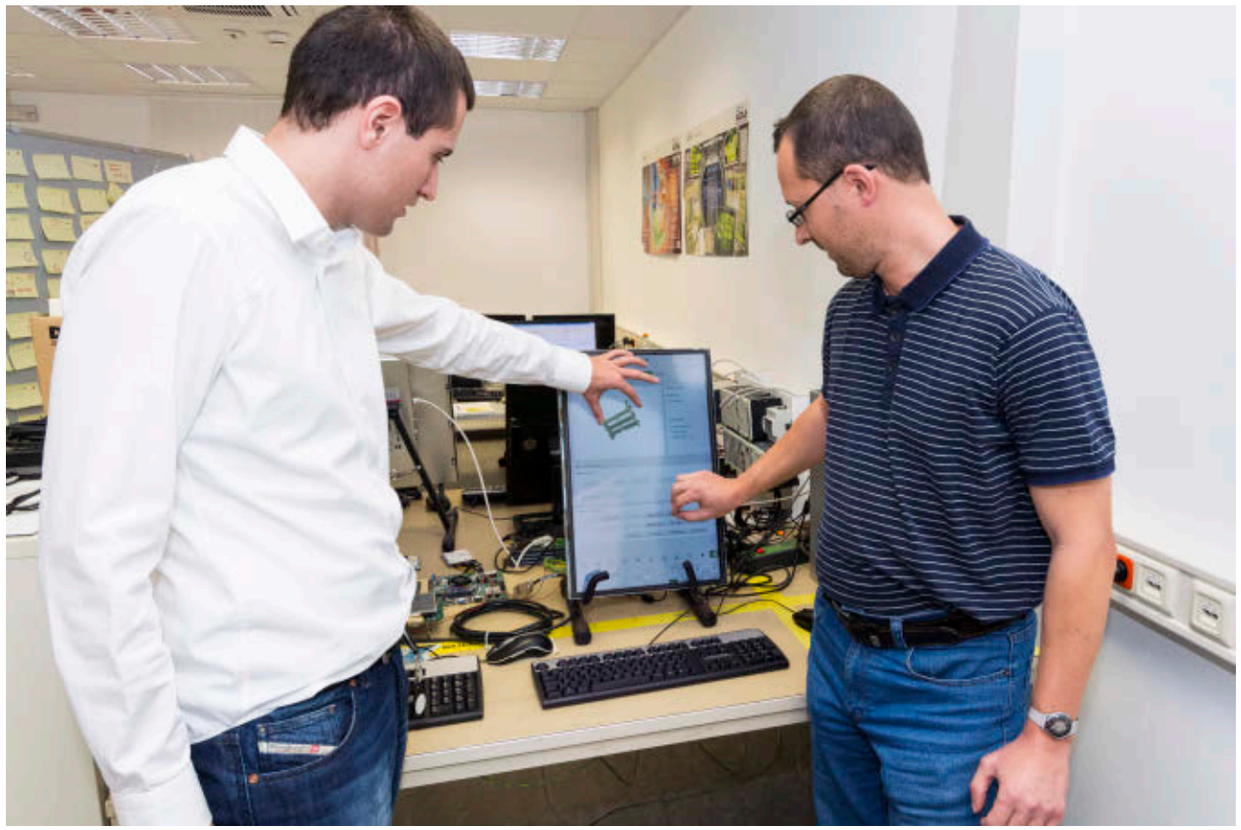
First industrial revolution: Use of mechanical production facilities driven by water and steam power.

Second industrial revolution: Introduction of division of labor and mass production with the help of electrical energy.

Third industrial revolution: Infusion of information technology to further automate production.

Fourth industrial revolution: The dawn of cyberphysical systems to facilitate interaction between the virtual and physical worlds.

(Source: German Research Center for Artificial Intelligence)



Keba's Fabian Schöppl and Josef Milt test out a demo application.

As we stand on the threshold of a fourth industrial revolution, we can expect a manufacturing panacea that will combine today's ubiquitous communication networks with the equipment in smart factories. Germany Trade & Invest is encouraging the development of technologies that can independently exchange information and manage their own industrial processes. Researchers use the term *cyberphysical systems* to describe this interplay between the virtual and physical worlds. These systems facilitate these connections in much the same way that the internet transformed personal communication and interaction.

Similar movements are afoot in the US with the Smart Manufacturing Leadership Coalition, a nonprofit organization of manufacturers, suppliers, and technology companies that is supported by a consortium of universities, government agencies, and laboratories. The aim of this coalition is to foster collaborative R&D practices and encourage the formation of advocacy groups that can enhance manufacturing intelligence via common standards and platforms and shared infrastructure.

Industrial giant GE uses slightly different terminology to describe today's

Hannes Bachmayr demonstrates a touchscreen system.

manufacturing opportunities and upheavals. During a presentation at the D-Eleven conference in May 2013, CEO Jeffrey Immelt said GE pioneered the "Industrial Internet" to combine the positive effects of two transformative revolutions: the myriad machines, facilities, fleets, and networks that arose from the Industrial Revolution, and the more recent advances in computing, information, and communication systems that characterized the Internet Revolution.

One of GE's goals is to bring designers, suppliers, and production engineers together to design and test goods virtually, without requiring them to be in close proximity to either the materials or the machines. This type of workforce, united by common projects and interests rather than location, points to another important attribute of today's advanced manufacturing companies: their ability to break down geographic boundaries and create a smoothly functioning global workforce. For example, Bachmayr's distributed team of software engineers includes science partners at the Software Competence Center Hagenberg in northern Austria and independent researchers at Johannes Kepler University. An additional software engineering team in other countries handles the finer nuances of Eclipse and JavaFX development.

"We get together every three to six weeks to present ideas, meet customers, and discuss requirements," says Bachmayr. "In the interim we structure our development process based on well-structured increments called *sprints*."

VERSATILE SOFTWARE FOR A NEW ERA

Bachmayr is adamant about the benefits of using Java for creating HMIs, and he has built a European team that reflects tremendous depth of experience with the Java ecosystem. "We selected Java for its ease of migration and platform independence, and because it has tons of plugins and third-party tools," he states. He adds that the Keba team especially appreciates multitouch support with JavaFX, a software platform the team uses for creating rich desktop clients. "Java is an easy programming language to learn. Oracle is behind it, which benefits the community, and it's part of a steady progression of new technologies such as the Leap Motion Controller and other modern input devices. We compared Java with .NET on multiple criteria, and Java won out 12 to 9."

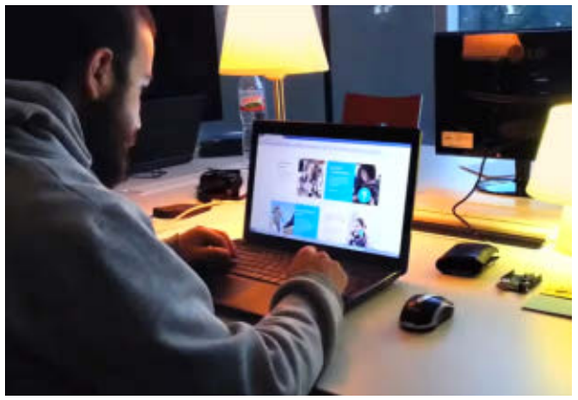
Despite differences in location, terminology, and equipment standards, Java unites Keba's software engineers and, by extension, paves the way for the factories of tomorrow. "We have licensed Java Virtual Machines for ARM versions 5 and 7 so we can easily install our applications on robotic devices with no modification at all," concludes Bachmayr. "Java runs on hundreds of mobile devices, operating systems, and embedded devices, so it was the obvious solution for these industrial applications." `</article>`

MORE ON TOPIC:

Internet of Things

David Baum writes about innovative businesses, emerging technologies, and compelling lifestyles.

The Lhings team at JavaOne in San Francisco (from left to right): José Antonio Lorenzo, David Peñuela, and José Pereda



Watch the Lhings Connected Table in action.

IoT Developer Challenge Winner

Lhings Connected Table

A custom office, wherever you are **BY KEVIN FARNHAM**

The Lhings Connected Table integrates embedded Java, the Raspberry Pi, and the Lhings cloud library with a set of connected devices (RFID cards, lighting, a coffeemaker, and more) to provide a customized office environment that's transferrable to any location. Lhings was one of three IoT Developer Challenge winners in the professional category.

"We wanted to explore how IoT [Internet of Things] could be applied to new areas," explains Lhings CTO José Antonio Lorenzo. "We thought about travelers who work in distinct locations each day and how convenient it would be if their office workspace could be replicated around the world."

The Lhings Connected Table is a step in that direction. The table lets a user log in to a shared workspace using an RFID card. The user ID information is transferred to the Lhings cloud platform, the user's preferences are retrieved, a text message welcoming the user is sent to his or her mobile phone, and the Raspberry Pi that controls the workspace implements the user's preferences (illumination, documents, printing preferences, and potentially anything else that could be controlled by an IoT device). When the workday is over, the worker scans the RFID card over the tabletop and the workspace returns to its default state, awaiting its next user.

The Lhings platform receives information from internet-connected devices, remotely controls them, and lets users build logic to make the devices work together. "Lhings allowed us to develop the first prototype in only two weeks," says Lorenzo.

"The IoT is totally changing the way people interact with their environment," he adds. "The table is an example of IoT that goes beyond the most-obvious examples. It provides an integral experience in which several devices work together to achieve a common goal, offering the user a customized workspace; delocalizing the desktop; and offering a set of services that make working simpler, convenient, and more productive." `</article>`

Internet of Things

FAST FACTS

- The Lhings Connected Table lets users customize their workspaces by defining preferences for potentially anything that can be controlled by an IoT device (for example, desktop lighting).
- Preferences are stored in the cloud, and are available around the world at any workspace where the Lhings Connected Table platform is installed.
- *Lhings* is a contraction of *Links things*.

PHOTOGRAPH BY BOB ADLER/GETTY IMAGES

MICHAEL KÖLLING

BIO

Internet of Things

PHOTOGRAPH BY JOHN BLYTHE

Part 1

Code Java on the Raspberry Pi

BlueJ brings Java SE 8 development directly to the Raspberry Pi.

Java programming directly on the Raspberry Pi? Is it possible?

Yes, it is! If you want to find out how, read on.

The Raspberry Pi probably does not need much introduction anymore among readers of *Java Magazine*. It is a small, roughly credit card–size computer (see **Figure 1**) that plugs into your monitor and keyboard. Most often, it is used for electronics projects or to learn about programming. And it is cheap—you can get one for around US$40.

Various Java enthusiasts have done amazing things with the Raspberry Pi—look, for example, at Simon Ritter's "Carputer" project. It connects the Raspberry Pi to your car using Java. A few minutes with your favorite web browser will enable you to discover various other cool projects.

Yet the official Raspberry Pi website—when it comes to programming—states that the computer is "a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python." Scratch and Python? What? No Java?

How can this be?

Java on the Raspberry Pi

The issue here is that you can easily develop Java *for* the Raspberry Pi, but much less so *on* the Raspberry Pi.

In the early days of the Raspberry Pi, you could install and run OpenJDK. OpenJDK was not optimized very well for the platform, and performance was too poor for many tasks. As a result, the Raspberry Pi Foundation did not promote Java much for the Raspberry Pi.

Then, in the fall of 2013, Oracle released a JDK optimized for the Raspberry Pi ARM platform. Making use of the hard-float API and optimizing many other parts, Java then became viable. It runs very well (within the restrictions of the platform), and the Raspberry Pi Foundation started to include Oracle's JDK in its software stack (the Raspbian images distributed to run the Raspberry Pi) by default.

So all is well, then? Not quite.

Writing Java *for* the Raspberry Pi

When the Raspberry Pi was launched, the founders had a clear vision: They wanted kids to start tinkering with computers and with programming again. The family computer, so the thinking went, is too precious; parents won't let young aspiring programmers play with it for fear that they will muck things up. No

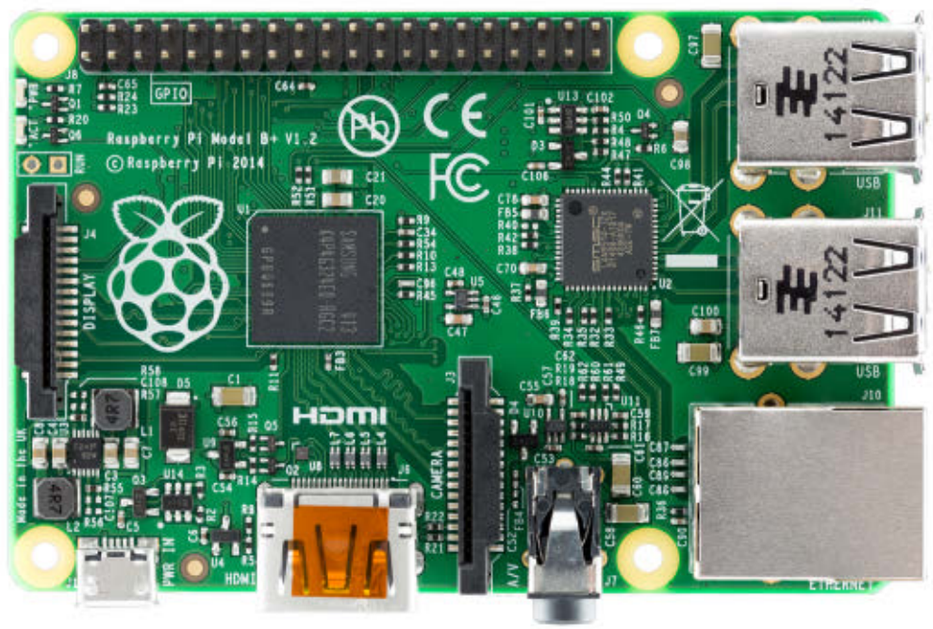


Figure 1

experimentation allowed.

The solution: Give kids their own, cheap computer that they can play with all they want. What if they delete half the operating system? No problem—reinstall and start again. Experiment, play, tinker, and learn to program.

It's a great idea, except that for Java, it does not work well. Java development typically takes place in an integrated development environment (IDE), for example, NetBeans IDE, Eclipse, IntelliJ IDEA, or something like it. None of these popular IDEs runs acceptably on the Raspberry Pi; they are all too big and resource hungry to make using them on the Raspberry Pi a realistic proposition.

BLUEJ FOR PI

With BlueJ version 3.14, you can now program in Java SE 8 directly on the Raspberry Pi.

Of course, you can always use a plain-text editor and command-line compiler. But apart from the fact that this is not how modern software development works—and, therefore, not what we should teach our kids—it is also not conducive to learning. Too many accidental complexities are put into the path of young learners, leading to frustration and failure, rather than to experimentation and excitement. Setting up the CLASSPATH correctly for the inevitably required libraries to access the hardware components alone is a stumbling block that will put an end to many attempts at tinkering.

As a result, until now Java development for the Raspberry Pi took place on a separate computer. The program was developed on a standard laptop or desktop, and the completed executable was then transferred to and run on the Raspberry Pi. This constituted development *for* the Raspberry Pi, but not *on* the Raspberry Pi.

While this is a viable workflow for knowledgeable hobbyists and enthusiasts, it misses the original vision. Rather than being able to use the Raspberry Pi *instead of* the full computer, you now need to use it *in addition to* your main machine. More, not less, hardware was now needed.

Writing Java *on* the Raspberry Pi

To change all this, BlueJ for the Raspberry Pi has now been released.

The latest BlueJ release (with the fortuitous version number 3.14) has been optimized to run well on the small computer directly. It also includes a library by default, Pi4J, which enables access directly to the hardware components of the Raspberry Pi.

Thus, with BlueJ installed, you can now program the Raspberry Pi using Java SE 8 and using all the standard Java/BlueJ examples that were previously available for standard desktops. Your Raspberry Pi can finally replace your desktop.

But you can also do more than that. You can create interactive objects that represent hardware components connected to the Raspberry Pi, such as buttons or LEDs, and have a more interactive and experimental setup for this kind of development than ever before.

For the rest of this article, and continuing in a future article in the next issue of *Java Magazine*, we will explore installing and using BlueJ, from BlueJ basics to exciting interactions with the Raspberry Pi hardware.

Download and Installation

For the purpose of this article, it is assumed that you have a Raspberry Pi and you have already figured out how to set it up and get started. (If not, you might want to start here.) Make sure that your Raspberry Pi is connected to the internet.

We will use Java SE 8 and BlueJ for the following examples. Most likely, you do not need to install Java—if you got your Raspberry Pi within the last year, Java SE 8 will be preinstalled on your system. If your system is older, you can install Java SE 8 by opening a terminal and typing the command shown in **Listing 1**.

If you are not sure whether Java is installed (or which version is installed), type the following command:

```
java -version
```

After making sure that you have Java, you are ready to download BlueJ. Open your web browser, go to bluej.org, and download the Ubuntu/Debian Linux version of BlueJ. (You can also use this direct download link, but if you read this long after the initial publication of this issue of *Java Magazine*, it would be better to check the BlueJ site for newer versions.)

LISTING 1

```
sudo apt-get update &&
sudo apt-get install oracle-java8-installer
```

Download all listings in this issue as text

Once you have downloaded the file, install BlueJ using the following command:

```
sudo dpkg -i bluej-314.deb
```

Then start BlueJ by typing the following:

```
bluej
```

After BlueJ starts, you can open an existing project or create a new one. Some sample projects are included in the examples folder under /usr/share/doc/bluej. With a project open, BlueJ will display a class diagram of the current project in its main window (see **Figure 2**).

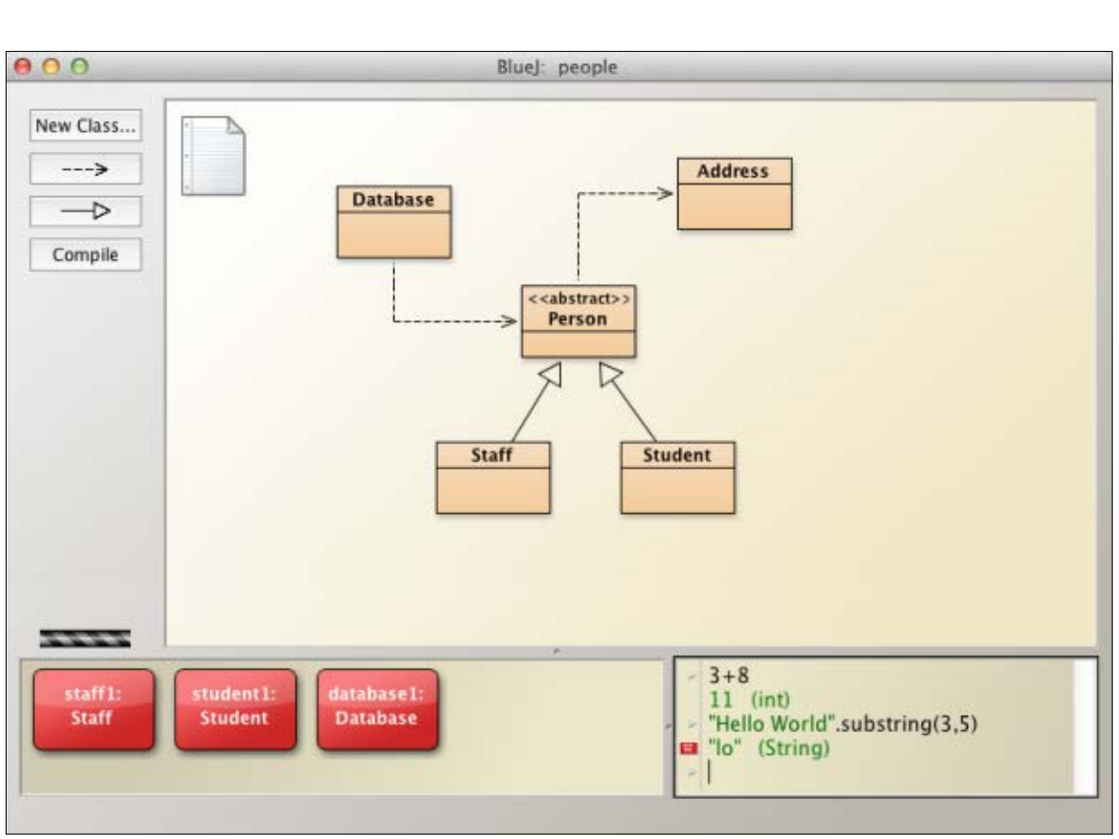


Figure 2

Getting Some Sample Programs

If you are new to BlueJ or to Java programming, you can download some more sample programs from the BlueJ book website—look for the "Book Projects" link.

Getting Started with BlueJ

If you are not familiar with BlueJ, it is worth spending a little bit of time finding out what it can do for you. It is a standard Java IDE, so you can write all the Java programs that you can write in other IDEs, but it can also do more. In BlueJ, you can interactively instantiate individual objects and call individual methods. This enables much easier testing, tinkering, and experimentation.

These BlueJ features have been described in previous issues of *Java Magazine*, and you can start there to learn more. The July/August 2012 and the September/October 2012 issues include an introduction to BlueJ, and the May/June 2014 and July/August 2014 issues discuss the interactive features of BlueJ in more detail.

Another good starting point is the BlueJ book website. The first two chapters of *Objects First with Java* are freely downloadable for evaluation and provide a good introduction. And, of course, you can always get the full book if you get hooked.

Making Use of the Raspberry Pi Hardware

So far, we have only looked at the original promise of the Raspberry Pi: using it to replace your desktop computer. We have used BlueJ in the same way we would have used it on other platforms. We have seen how you can now do Java programming on the Raspberry Pi without a second computer. Your options for learning to program with the Raspberry Pi now include Java.

But in addition to replacing the desktop machine, the Raspberry Pi can do more. Because of its exposed hardware, you can also start playing with other physical I/O devices: LEDs, buttons, motors, and so on.

In the next issue of *Java Magazine*, we will continue this discussion of BlueJ on the Raspberry Pi and explore some examples of combining BlueJ's interactivity with the Raspberry Pi's peripherals to start interactive programming with LEDs, buttons, and more.

Conclusion

It was high time to bring Java to the Raspberry Pi as a first-class citizen. External Java development with the Java executable transferred to and run on the Raspberry Pi is fine for many electronics hobbyists, but it does not allow kids to practice their programming skills on the Raspberry Pi itself.

With BlueJ version 3.14, you can now program in Java SE 8 directly *on* the Raspberry Pi—no second computer is needed. `</article>`

MORE ON TOPIC:

Internet of Things

LEARN MORE

- BlueJ
- Raspberry Pi

BEN EVANS

BIO

jdeps, Compact Profiles, and Java Modularity

A look at the future of Java modularity

In "Exploring Java 8 Profiles," we explored the idea of Compact Profiles—reduced versions of the Java runtime environment (JRE) designed for smaller-footprint deployments, such as headless or server-only deployments.

This article continues the discussion of Compact Profiles and digs deeper into what can be achieved using the jdeps tool. It also describes developments across the landscape of Compact Profiles and better modularization of the JDK.

The jdeps Tool

Let's start by revisiting jdeps, which is a static analysis tool for examining the dependencies of packages or classes. The tool can be used in a number of different ways, from identifying which profile an application needs to run under; to identifying developer code that makes calls into the undocumented, internal JDK APIs; to helping trace transitive dependencies.

In its simplest form, jdeps takes a class or package and provides a brief list of packages that are the dependencies for that class or package, as shown in **Listing 1**.

The -cp option, as usual, provides a classpath to use to find the appropriate class or package. The -P switch (shown in **Listing 2**) shows which profile is needed for a class (or package) to run.

jdeps also has the ability to delve deeper into dependencies. The -v and -V switches show dependencies to the level of individual classes and packages, respectively.

One of the most important things that jdeps can do is to test for the usage of internal APIs by user code. This is easily accomplished by using the -jdkinternals switch. This can easily be scripted into a very useful form via a shell script such as the one shown in **Listing 3**, which identifies any usage of internal APIs in any .jar file contained in the current directory or any of its subdirectories.

This technique for finding internal API usages works best for library code, for the simple reason that most applications will very likely rely on a framework or library that depends on an internal API somewhere.

For example, the Netty library depends on the internal implementation of Java's X.509 security libraries, Logback depends on low-level details of sun.reflect.Reflection, and a large number of libraries use the well-known sun.misc.Unsafe class.

Having said that, the -jdkinternals switch is a useful sanity check for developers who develop libraries and free and open source software. Code that is designed for use by other developers (especially by developers who are external to your organization) should stick to the public APIs unless doing otherwise is absolutely necessary.

Otherwise, if you rely on an internal API that changes or is removed from a new version of Java, users of your code will be unable to upgrade to the new Java version until you release a new version of your library that works around the changed or removed API.

Some limited support for detecting the use of internal APIs is also now provided by javac, the Java compiler, which

PHOTOGRAPH BY JOHN BLYTHE

currently flags the use of some, but not all, internal APIs by default. javac emits a warning if it detects the usage of any internal API from a list of internal APIs that were added to the JDK before JDK 6.

While this is helpful, one area where tooling could do more to support library (and application) developers is by analyzing dependencies. This could be achieved by using something similar to the jdeps tooling interface to build tree views of the dependencies of particular packages and classes. By doing such an analysis, developers could make their users aware of key facts, such as the following:

- This code is guaranteed to be API-safe, uses no internal APIs, and can be cleanly upgraded.
- This code can be run under a Compact Profile, such as compact1, and requires very little in the way of core dependencies.

The NetBeans IDE already has limited support for Compact Profiles, but only via manual selection and a rebuild. There is clearly scope for a more sophisticated and complete profile analysis plugin.

Stripped Implementations

There have also been some developments on another subject mentioned in the last article: the concept of a Stripped Implementation, which is essentially a reduced JRE or other Java API that does not necessarily contain a complete implementation of all the relevant Java technology. Under normal circumstances, using a Stripped Implementation would not be appropriate or desirable, because the whole point of Java standards is to provide a stable, unfragmented platform that developers and businesses can rely upon.

Instead, the primary use case for a Stripped Implementation is to provide a solution for a Java application that does not want to rely on the existence of a Java environment on a target machine, and instead wants to bundle a private environment, the precise specifications of which are known.

As it stands, the only way to do this is by bundling a complete JRE. Even the smallest Compact Profile (compact1) is around 11 MB, which can be a large overhead, especially for embedded deployments. For applications that have a GUI, there's no current option but to bundle the full JRE, because the current Compact Profiles do not subset the GUI classes in the JRE.

This is the problem that Stripped Implementations aim to solve by providing a mechanism through which standards producers can define how stripping can work for their particular standards. Now that Java SE 8 is out, the necessary changes to licensing and testing are well under way and expected to be completed soon.

LISTING 1 LISTING 2 / LISTING 3

```
$ jdeps -cp $HOME/projects/scales/target/classes
:$JAVA_HOME/jre/lib/rt.jar
:$M2_REPO/javax/inject/javax.inject/1/javax.inject-1.jar
:$M2_REPO/javax/ws/rs/javax.ws.rs-api/2.0/javax.ws.rs-api-2.0.jar
:$M2_REPO/org/slf4j/slf4j-api/1.7.7/slf4j-api-1.7.7.jar
com.scales.svc.SimulationService
/Users/ben/projects/scales/target/classes ->
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/
Home/jre/lib/rt.jar
/Users/ben/projects/scales/target/classes ->
/Users/ben/.m2/repository/javax/ws/rs/javax.ws.rs-api/
2.0/javax.ws.rs-api-2.0.jar
/Users/ben/projects/scales/target/classes ->
/Users/ben/.m2/repository/javax/inject/javax.inject/1/
javax.inject-1.jar
/Users/ben/projects/scales/target/classes ->
/Users/ben/.m2/repository/org/slf4j/slf4j-api/1.7.7/
slf4j-api-1.7.7.jar
   com.scales.svc (classes)
      -> java.lang                 rt.jar
      -> java.time                 rt.jar
      -> java.time.format          rt.jar
      -> java.util                 rt.jar
      -> javax.inject              javax.inject-1.jar
      -> javax.ws.rs               javax.ws.rs-api-2.0.jar
      -> javax.ws.rs.core          javax.ws.rs-api-2.0.jar
      -> org.slf4j                 slf4j-api-1.7.7.jar
```

Download all listings in this issue as text

Java and Modularity

Finally, let's take a look beyond Compact Profiles, and look into the future of Java modularity. Java SE 8 has only just been released, but there have already been some announcements about the content of the Java SE 9 release.

Java Chief Architect Mark Reinhold announced the return

of modularity in a batch of Java Enhancement Proposals (JEPs) that make up the first announced features that are intended to become part of Java SE 9.

This batch includes JEP 201, titled "Modular Source Code." Despite the name, this JEP does not introduce modularity in any sense that is visible to a developer of Java applications, and the OpenJDK 9 builds as they stand have no trace of modularity in the binaries.

Instead, JEP 201 is a first step that has implemented some necessary housekeeping and has rearranged the layout of the OpenJDK source tree, so that it is now in modules. This JEP has already been completed and is integrated into the JDK 9 source repositories. Let's look at how these new modules are laid out in the OpenJDK 9 source directory (see **Listing 4**).

Inside each module, the directory structure has also been rearranged, with one interesting side effect. The code that was historically common between UNIX-like operating systems was contained in a directory called solaris. With this reorganization, this confusing (and rather legacy) name has been changed to the more accurate unix.

Reinhold also announced a new version of Project Jigsaw, the project aiming to deliver a modular JDK. His vision is of four fairly large JEPs that will deliver full modularity. The first of these JEPs is, unsurprisingly, JEP 201. The second is JEP 200, which aims to describe and set out the module structure without actually implementing it. The summary of this JEP says,

> *"Define a modular structure for the JDK. Make minimal assumptions about the module system that will be used to implement that structure."*

Within those minimal assumptions is the idea that any module that starts with java is one that will be governed by the Java Community Process (JCP) and by Java Specification Requests (JSRs). For developers unfamiliar with the standardization process, what this means is that the java modules are those that are intended to be portable across Java implementations (in a future world, in which Java has been fully modularized).

In contrast, the modules we see that start with jdk represent internal implementations that are specific to OpenJDK. JEP 200 is still in development, but we can see the first traces of it in the delivery of JEP 201, where the big split between java modules and jdk modules is obvious.

Conclusion

The arrival of Compact Profiles and jdeps represents a great first step toward a more modular future, but we still have a long way to go before we see a fully modularized JRE. In particular, the build process needs to be updated to produce modular binary artifacts (and the end of a monolithic rt.jar), and the class-loading subsystem needs to become aware of modules as a primary mechanism for loading code into a Java process. In the coming months, a great deal of activity is to be expected, with further updates from the platform team as Project Jigsaw progresses and the modularity train starts to pick up steam again. `</article>`

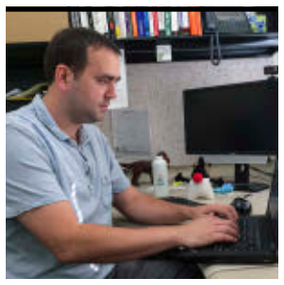
LISTING 4

```
ben$ pwd
/Users/ben/projects/openjdk/jdk9/jdk/src
ben$ ls
bsd                 java.rmi            jdk.charsets
jdk.jconsole        jdk.security.auth
demo                java.scripting      jdk.compiler    jdk.jdi
jdk.snmp
java.base           java.security.acl   jdk.crypto.ec
jdk.jdwp.agent      jdk.zipfs
java.corba          java.security.jgss  jdk.crypto.mscapi
jdk.jvmstat         linux
java.desktop        java.security.sasl  jdk.crypto.pkcs11
jdk.localedata      sample
java.instrument     java.smartcardio    jdk.deploy.osx
jdk.naming.dns      solaris
java.logging        java.sql            jdk.dev
jdk.naming.rmi
java.management     java.sql.rowset     jdk.hprof.agent
jdk.rmic
java.naming         java.xml.crypto     jdk.httpserver
jdk.runtime
java.prefs          jdk.attach          jdk.jcmd
jdk.sctp
```

Download all listings in this issue as text

LEARN MORE

- jdeps
- JEP 201: Modular Source Code
- JEP 200: The Modular JDK



ERIK COSTLOW AND ERIC RENAUD

BIO

JDK 8u20 Improves Performance, Security, and Manageability

Learn how to better control managed systems that run multiple rich internet applications.

Following the Java 8 release earlier this year, Java SE Development Kit 8, Update 20 (JDK 8u20) continues to improve upon the significant advances made in the Java SE platform. As the latest minor release of Oracle's implementation of Java SE, JDK 8u20 was architected, in part, to provide enterprise system administrators with the ability to better control managed systems on which users run multiple rich internet applications (RIAs), specifically Java applets and Java Web Start applications.

In this article, we will explore the two new tools most beneficial to such administrators, the Java Advanced Management Console and the Microsoft Windows Installer Enterprise JRE Installer, along with other important improvements in the JDK 8u20 release.

REDUCE RISK

By using the Advanced Management Console, administrators can reduce security concerns by effectively creating whitelists and blacklists.

Advanced Management Console

First, let's address the Advanced Management Console. It is available in the Oracle Java SE Advanced products, which provide enterprises and independent software vendors (ISVs) with support and specialized tools such as the Advanced Management Console and Oracle Java Mission Control.

The functionality of Advanced Management Console 1.0 can be boiled down to two areas. It directly tracks usage data for Java applications, and it enables administrators to act on that data in a guided manner. Using the Advanced Management Console results in a more easily controlled and more secure environment that provides an improved end user experience, as we will see below.

A primary benefit of the Advanced Management Console is the ability to learn which RIAs (Java applets and Java Web Start applications) are being run in an enterprise as well as which Java runtime environments (JREs) are used. Additional information—such as the location of each application, the vendor, the permission level, and the number of times the application has been run—is also provided, all of which can be gathered from a large number of clients across an enterprise.

How is this accomplished? It's done by way of the Usage Tracker, a Java feature that enables the Java clients on the desktops within an enterprise to report RIA and JRE usage data. The usage data gathered (including the type of virtual machine start, the date and time of the start, the host name and IP address, the application name, and much more) is stored in a normalized database for performance reasons, which means that a single Usage Tracker can handle a large number of clients.

(As an aside, the Usage Tracker is off by default. It is

ERIC RENAUD PHOTOGRAPH BY VOLTAIRE YAP

enabled by creating the properties file <JRE directory>/lib/management/usagetracker.properties. Simply placing that file on a client informs the system to report information to the Usage Tracker. The information is then sent via the User Datagram Protocol [UDP] to prevent any delay on the client. A bit of serendipity is that the properties file can be added by using the new Microsoft Windows Installer Enterprise JRE Installer tool we'll discuss below.)

The data from the Usage Tracker is collected by the Advanced Management Console's Collector, stored in the Advanced Management Console's database, and displayed in the Advanced Management Console's user interface (UI).

Figure 1 and the following steps describe the process for transmitting (reporting) and storing the usage data and making it available for viewing and analysis (visualization).

1. *Reporting:* Java clients are configured with a properties file, either manually per client or automatically across a single or multiple clients through the new Microsoft Windows Installer Enterprise JRE Installer, which tells the clients the location of the Usage Tracker. This file must be in place for reporting to occur. Then, when the clients start, they provide information about the applications being launched. (As described earlier, the asynchronous UDP packets do not affect the startup performance of the applications.)
2. *Storage:* The Advanced Management Console's Collector gathers information about applications running in the enterprise from Java Usage Tracker via UDP packets and stores the data in the Advanced Management Console's database, managing the growth that can occur for a large user base.
3. *Visualization:* The Advanced Management Console displays usage information about which applications were used, how often they were used, and with which JRE they were used.

So what does all this usage data do for administrators? It helps them identify the applications that users need to do their work and the versions of Java required to run those applications. And, with security in mind, administrators can also learn whether users are running applications that are not approved or they are running approved applications but using outdated versions of Java.

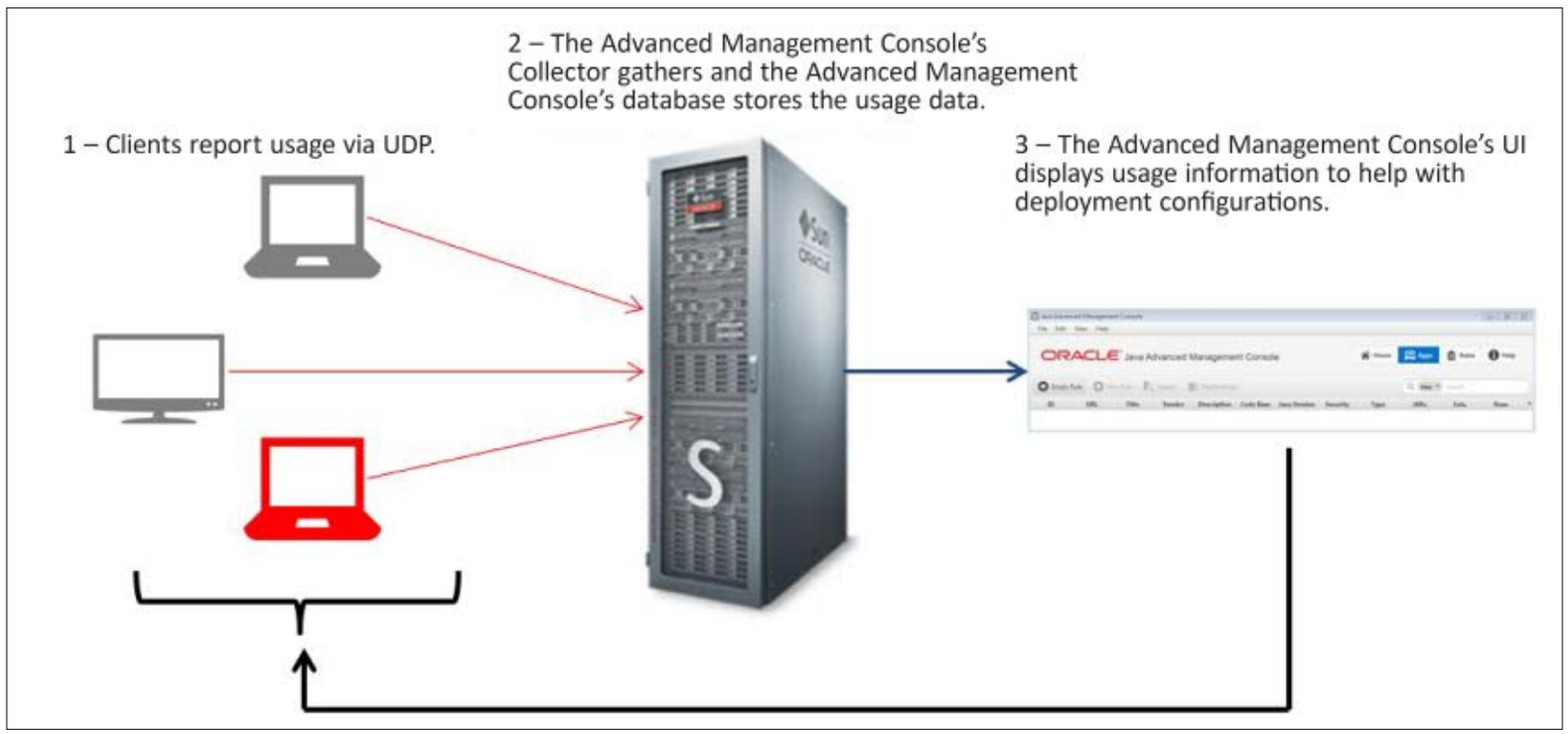


Figure 1

Figure 2 shows a view of the Advanced Management Console's UI showing the area where usage data would be displayed about applications running on the desktops across an enterprise.

Now, let's look at how the Advanced Management Console enables administrators to take action on the usage information.

The Advanced Management Console uses the Deployment Rule Set security feature, which was introduced in Java SE Development Kit 7, Update 40 (JDK 7u40). The goal of this feature was to help administrators enforce policies regarding which RIAs are allowed or disallowed and which JRE ver-

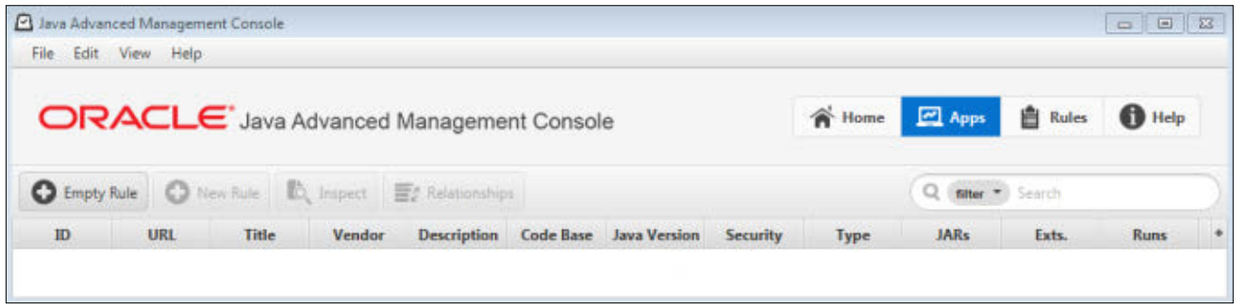


Figure 2

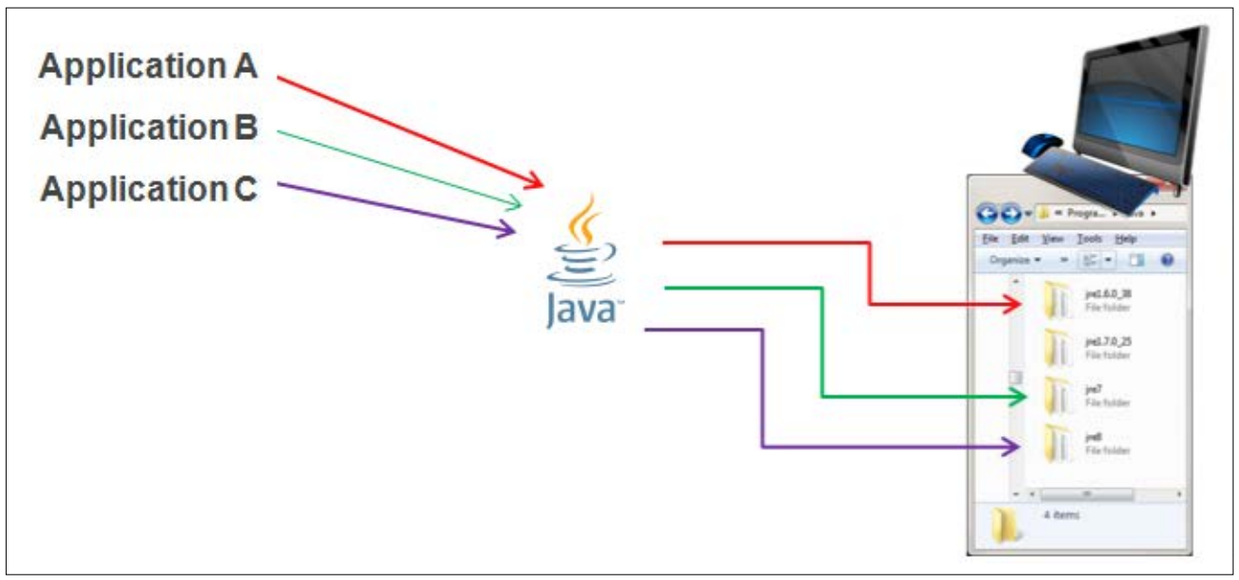


Figure 3

sions can be used to launch them. An additional advantage is that the end user experience is enhanced, because the Deployment Rule Set feature can also control which security prompts users see.

As illustrated in **Figure 3**, the Advanced Management Console's UI provides administrators with the ability to create deployment rules and deployment rule sets and store them in the Advanced Management Console's database, thereby achieving greater control over a more secure network infrastructure.

There are many uses for the Advanced Management Console, but its main advantages are that it equips administrators with a tool set to match applications to desired JREs, to maintain a record of the deployment rule sets that are deployed, and to gather data about applications that are run in the enterprise.

In JDK 8u20, a new "force" feature was added for use with deployment rule sets, which provides administrators with the ability to specify that an application be run with a different JRE version than the version specified in the application itself.

Deployment rule sets also offer a secure way of managing compatibility with older versions of Java. By using a deployment rule set, the latest and most secure version acts as a proxy to allow only "known to be safe" applications to run with older, compatible JRE versions. As a result, most applications use the current, secure JRE and older JREs are limited to running "known to be safe" applications. Similarly, a deployment rule can be deployed that causes the "last known to run" JRE to be used for a particular application while keeping all the other applications on up-to-date JREs. Thus, by using the Advanced Management Console, administrators can reduce security concerns by effectively creating whitelists and blacklists.

An important item to note is how guided rule creation and packaging support greatly simplify developing deployment rule sets. The Advanced Management Console can also be used to determine which rules and rule sets an application matches, helping system administrators understand the impact of installing a particular rule set prior to actually testing the rule set in user environments. During the guided creation of deployment rule sets, Usage Tracker data identifies applications by certificate hash and by location. In addition, a comparison tool is available to verify rules against tracked data, thus enabling easier testing. While administrators can always create deployment rule sets by hand using a text editor, these new tools greatly reduce the time and effort required and make the process far less error-prone.

In summary, the Advanced Management Console enables system administrators to easily identify RIAs and JREs, and it provides tools for controlling the com-

TRACK AND ACT

The functionality of Advanced Management Console 1.0 can be boiled down to two areas. It directly tracks usage data for Java applications, and it enables administrators to act on that data in a guided manner.

patibility and availability of older Java installations through deployment rule sets in a scalable manner across the enterprise, all of which results in a streamlined experience for users. **Figure 4** shows a view of some rules and rule sets that have been deployed in the Advanced Management Console.

Microsoft Windows Installer Enterprise JRE Installer

The new Microsoft Windows Installer Enterprise JRE Installer integrates into various desktop management tools, making it easier to customize and roll out different Java SE versions.

Available in the Oracle Java SE Advanced products for Windows 64- and 32-bit systems, this new installer provides a number of benefits for system administrators who customize or manage software in the enterprise at scale. Unlike the basic installer that most users obtain from Java.com or Oracle Technology Network, this installer is built around customization and integration with various desktop management products such as the Microsoft System Center Configuration Manager (SCCM), and it provides automated, consistent installation of the JRE across all the desktops in an enterprise.

System administrators who use the Microsoft Windows Installer Enterprise JRE Installer can use every capability provided by the standard Windows Installer, such as silent installations and upgrades, low-privileged installations, and self-repair capabilities. Other common features—such as rolling back unsuccessful installations, repairing broken installations, and installing over existing broken installations—are all available with the Microsoft Windows Installer Enterprise JRE Installer. Integrated with the installer is the Java Uninstall tool, which provides the option to remove older versions of Java from a system.

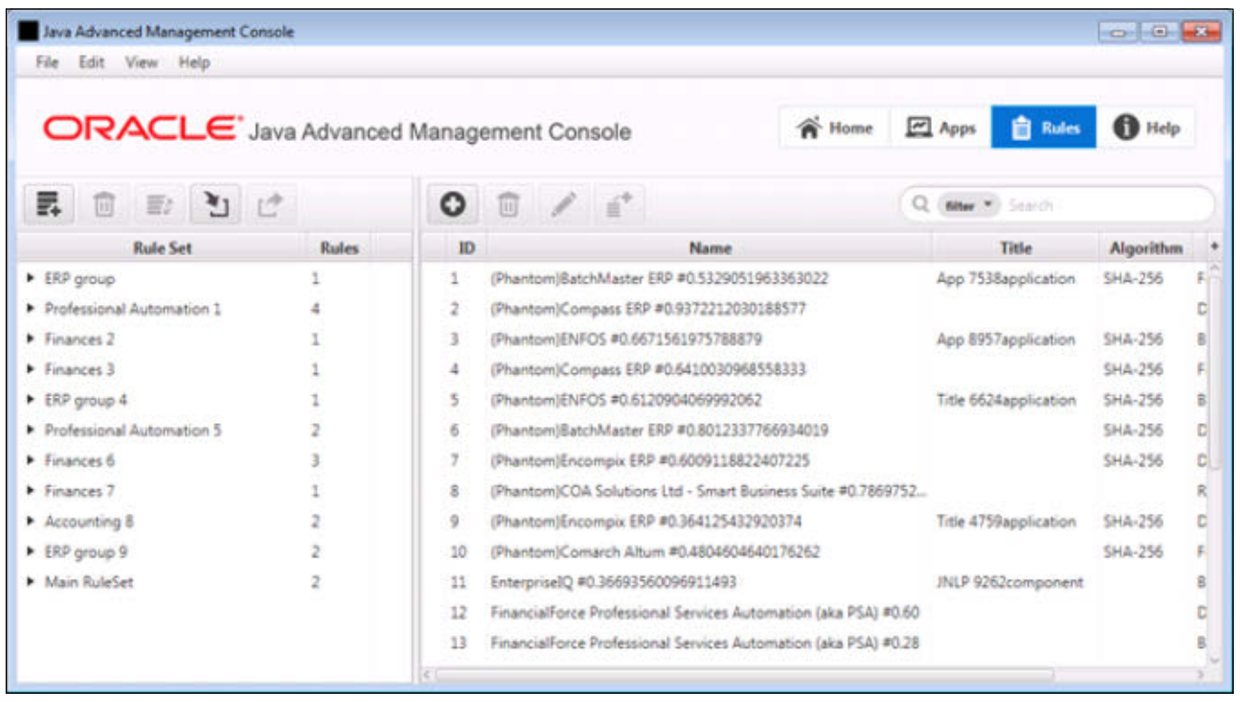


Figure 4

Other Improvements

JDK 8u20 also provides enhancements that improve performance, such as a reduced memory footprint, several patches geared to next-generation CPUs, and improved support for garbage-first (G1) garbage collection for long-running applications.

For G1 garbage collection, the newly added String deduplication capability described in JDK Enhancement Proposal (JEP) 192 does not actually deduplicate the String objects; it deduplicates only their backing character arrays. However, the resulting optimization is elegant, removes inefficiency, and results in heap reduction.

Conclusion

While the enhancements and optimizations in JDK 8u20 are numerous and impactful, it is the new tools it provides in Oracle Java SE Advanced products that add the luster to this release. The new tools make it easier for system administrators to identify and control client installations at scale. Administrators at organizations that want either the tools or associated commercial support should consider using Oracle Java SE Advanced products.

Oracle Java Mission Control (featured in the July/August 2014 issue of *Java Magazine*) continues to be available as a commercial feature in the Oracle Java SE Advanced products. The new version, Oracle Java Mission Control 5.4, is bundled with JDK 8u20 and includes several enhancements to improve usability.

Advanced Management Console and Oracle Java Mission Control require an Oracle Java SE Advanced product license for production use, but each tool is available for download for development and evaluation purposes from Oracle Technology Network. `</article>`

LEARN MORE

- JDK 8u20 Update Release Notes
- Oracle Java SE Advanced

KAI KREUZER

BIO

A Smart-Home Platform for the Mass Market

Eclipse SmartHome bridges the gap between tech-savvy users and average users to provide a smart-home platform for everyone.

There are plenty of smart-home offerings on the market, but most are tailored toward certain use cases and a fixed set of supported devices. The Java-powered open source project openHAB (open Home Automation Bus) has established itself as a Swiss army knife for realizing overarching use cases. Previously addressing only tech-savvy users, it has further evolved into the new Eclipse SmartHome project, which has a flexible platform that allows building mass-market solutions.

2014 is the year of the Internet of Things (IoT). According to Gartner, hype about the IoT is currently at a peak, which means the market has huge expectations that soon might give way to some disillusionment. Smart-home technology is often regarded as a subset of the IoT—the part that deals with consumer products for the home. Luckily, Gartner mentions *connected home* as a separate category that is still in the innovation phase. We can, therefore, expect more public awareness about smart homes in the future. And this year was not short on related news with Google buying Nest, Apple introducing HomeKit, and Samsung acquiring SmartThings. It is going to be an exciting future.

An Emerging Market

But let's begin with what is currently available. Smart-home technology has existed for a few decades already, but it has never found its way into the mass market. There are actually two main segments:

- The high-end market for luxury homes, in which cost does not matter and homeowners have professional installers customize everything to their needs. In this segment, smart-home technology is often coupled with alarm systems and private security services. Because installers require training and prefer tested and reliable solutions, this segment is usually not where innovation happens.
- The "do it yourself" (DIY) market, in which tech-savvy people love spending their time but not too much of their money. This group usually comes up with creative and innovative ideas and prefers systems that can be easily changed, reconfigured, and adapted.

The mass market in between these two segments is going to be conquered from the ground up. Just as personal computers were initially assembled by nerds and were then adapted for the average consumer, the same is likely to happen for smart-home technology.

openHAB has attracted a large community of DIY users. The project serves as a central integration point where different home automation systems and protocols can be combined into one overarching solution that provides uniform user interfaces across all devices and, more importantly, a possibility to define automation logic that bridges all gaps. All this is done through textual configuration and scripting, which is flexible and powerful for the

Internet of Things

PHOTOGRAPH BY TON HENDRIKS

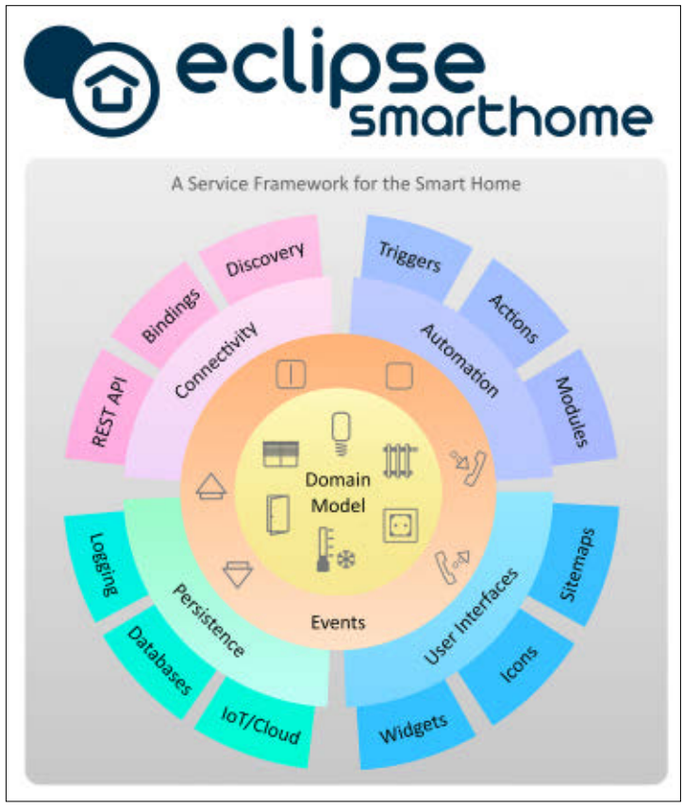


Figure 1

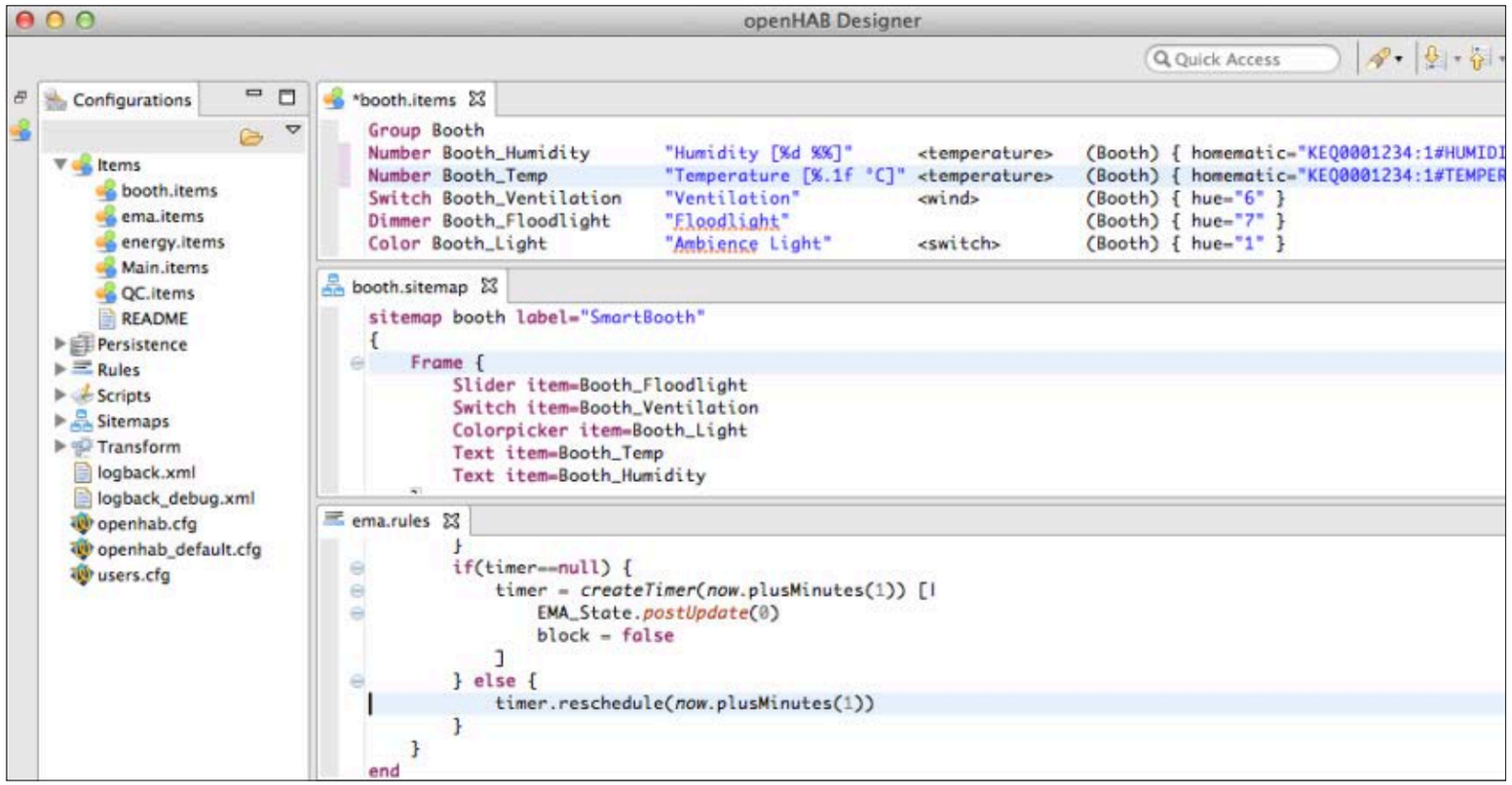


Figure 2

tech-savvy user but daunting for the average user. To address this problem and broaden the potential user base, the code for the openHAB framework was contributed to the Eclipse Foundation in early 2014 as an initial seed for the Eclipse SmartHome project.

A Home Gateway Stack

The Eclipse SmartHome project is meant to be a software framework tailored for home gateways. It provides APIs and services for four categories: connectivity, automation, user interfaces, and persistence (see **Figure 1**).

Connectivity covers not only communication via a certain protocol or a proprietary interface among devices or systems, it also deals with making the gateway accessible in a homogenous way for client applications. Just as the framework expects other systems to provide open APIs, it itself exposes its functionality through a REST API.

Automation is all about rules and the smart parts of a smart home. In essence, this is the code that brings the whole system to life. Homes are extremely individualistic, and no two setups will ever look alike. Therefore, a paramount requirement is that the framework offers flexible mechanisms for implementing logic. This requirement can be reached only by using some kind of programming language.

User interfaces (UIs) are the visual component of a smart-home solution. To appeal to users, it is important that UIs offer a consistent way of accessing devices and visualizing the current state of devices. Ideally, there is one UI to address all needs, not a plethora of different apps. As a result, such a UI must be generic enough to easily adapt to new kinds of devices and use cases.

Persistence transforms ephemeral events into permanent information. Persistence can be provided through local databases or cloud services, and the data can be used for reports or charts or for inferring automation actions. Such persistence services must be configurable with respect to which data should be collected (for example, every changed value) and with which kind of strategy (for example, at a fixed frequency).

All these aspects were addressed in a textual way in openHAB, as can be seen in **Figure 2**. For Eclipse SmartHome, completely new APIs

and services have been developed with the goal of enabling the creation of easy-to-use setup and configuration tools that are compatible with end user needs. So what exactly are the challenges for achieving this goal?

Architectural Challenges

One major task when setting up a system is to define all the devices that are supposed to be part of the solution. The most user-friendly way is obviously autodiscovery. Nonetheless, users might not want to use every device that is available on the network, so they should have a chance to approve the use of each device. In Eclipse SmartHome, this is done through an inbox, which keeps track of discovered things and adds them to the system only upon manual approval.

An important design goal is to make life for developers as simple as possible, and a good way to achieve this is to offer abstract superclasses. **Listing 1** is an example implementation of a discovery service that shows what developers have to implement in order to provide a discovery mechanism for a dedicated type of device they want to support.

A common way to discover IP-enabled devices on the network is through Universal Plug and Play (UPnP). The UPnP specification is quite complex, and it does not make sense that every device binding comes with its own implementation or bundled library, but there is already a shared UPnP discovery service in place. Developers can hook into this service to participate in analyzing the received information and to derive discovery results from it.

This shared service builds on top of the open source jUPnP library and simply hands discovered UPnP devices over to the participants, which only have to implement a simple interface. A real-life implementation for the Philips Hue system is shown in **Listing 2**, which demonstrates how easy it is to add support for a new device without having to deal with the inherent complexity of the UPnP protocol.

Discovering and identifying devices is not enough, though. To enable the system to make use of the devices, the devices' functionality must be described in a formal way.

Eclipse SmartHome defines a Java object model for this and provides an XML schema and parser as a default mechanism for describing things. **Listing 3** shows an example description for an LED bulb. Since a common ontology for all the things in a smart home is an ongoing topic of research, this model is based on a capabilities-based approach,

LISTING 1 LISTING 2 / LISTING 3

```
public class MyDiscoveryService
        extends AbstractDiscoveryService {


 private static final int SCAN_TIMEOUT = 5; // in seconds
 private static final ThingTypeUID MY_THINGTYPE =
new ThingTypeUID("my:thingtype");

 public MyDiscoveryService() throws IllegalArgumentException {
  super(Collections.singleton(MY_THINGTYPE), SCAN_TIMEOUT);
 }

 protected void startScan() {
    // do something to get hold of the thing
    // ...
    DiscoveryResult result = DiscoveryResultBuilder
                .create(new ThingUID("my:thingtype:1"))
                .withLabel("My great device")
                .withProperty("ip", "10.0.0.1")
                .build();
   thingDiscovered(result);
  }
}
```

Download all listings in this issue as text

Figure 3

which allows describing new kinds of things without having to extend the existing model.

A general problem for applications is to know the semantics of a device. When bundling hardware for a given use case, this is simple: The software can expect that the user uses the bundled devices (for example, a switchable socket for controlling a lamp).

But for a general-purpose solution, the context is often unknown, and it cannot be statically defined by the device description. Only the users know what they have plugged into a switchable socket. If the device is a refrigerator, it obviously should not be included in an all-off scenario, while a lamp should be included. Eclipse SmartHome suggests a simple but powerful solution to this problem: Functionalities can be tagged with tags such as "light," "security," and "energy" so that this rough categorization provides some context to applications and to rule logic.

In addition to devices, all kinds of services (such as persistence and notifications) usually require some user configuration. To cater to the extensibility of solutions, generic configuration UIs are required. Describing configuration properties as HTML5 input types already covers important aspects such as a type system, "required" flags, and input constraints.

Nonetheless, a metadata facility has to provide options to also group and order properties, so that a nice layout can be achieved. Also, special properties, such as geo-locations or OAuth2 access tokens, require very special widgets, such as an embedded map or an external web flow. It is, therefore, a big challenge to cover these requirements in a way that addresses all relevant use cases while keeping the implementation complexity for configuration UIs at an acceptable level.

The Power of Scripts

Automation rules can be highly individualistic, and very often the best settings are found only by trial and error. This makes it important to have short round-trip cycles—code compilation and deployment should not be necessary in order to apply a new set of rules. Scripting languages can accomplish this.

Eclipse SmartHome comes with an Xtend-based scripting language. However, in order for scripts to be interpreted at runtime, a rather large number of third-party libraries is required, which is not ideal for embedded use.

Java SE 8 comes to the rescue: It ships with the very performant JavaScript engine, Nashorn. Nashorn compiles JavaScript down to Java bytecode and, thus, can run scripts directly on the Java Virtual Machine (JVM). Nashorn furthermore comes with a transparent mapping from Java to JavaScript code and vice versa. It is, therefore, a perfect match for a future rule engine.

Having scripting capabilities is not enough for a mass-market solution, though. Instead, scripts need to be parametrizable, so that they can be created by developers and then reused by consumers. Such parametrizable scripts can be regarded as rule templates that serve a certain use case, but which can be individually adapted. Such an infrastructure is going to be introduced in Eclipse SmartHome in conjunction with Java SE 8 and Nashorn.

NERDS FIRST

Just as personal computers were **initially assembled by nerds** and were then adapted for the average consumer, the same is likely to happen for smart-home technology.

Building Commercial Solutions

As a pure framework, Eclipse SmartHome's scope is limited to functional categories. Because it is a Java software stack that uses Open Service Gateway initiative (OSGi) for modularization, building a home gateway (such as the one shown in **Figure 3**) requires the rest of the stack to be defined as well. Eclipse SmartHome offers a wide variety of options here; all it requires is a Java SE 7–compliant JVM and an OSGi R4.2 framework implementation.

Typically, a home gateway with an embedded ARM CPU runs a Linux OS with the JDK and a small-footprint OSGi framework, which can be an open source framework, such as Apache Felix, or a commercial one, such as the implementation from ProSyst. However, remote management of the gateway—for example, applying updates, doing backups, or providing remote access—is out of scope. Various commercial offerings can be chosen for that functionality.

Because the footprint of the software stack always matters, Java SE 8 brings a new and elegant way of reducing it: Compact

Figure 4

Profiles. Compact Profiles reduce the JVM library to the parts that are essential for an embedded solution. All AWT, Swing, and other headful parts are stripped out, which cuts the size easily by more than 50 percent. In conjunction with Oracle Java SE Embedded, this results in a very small and lightweight JVM, which still allows Java SE applications to be run on it. The goal of Eclipse SmartHome is to be compliant with the compact2 profile. (The compact1 profile requires JAXB for the JAX-RS–powered REST API, so it would not be feasible.)

One of the first companies to integrate Eclipse SmartHome in a commercial home gateway is Deutsche Telekom. Its QIVICON platform is clearly targeted at the mass market as a self-installable solution. It has an easy setup process, so it is a perfect fit for the mass market. QIVICON comes with its own UIs that make use of the Eclipse SmartHome APIs. This allows QIVICON to keep its unique look and feel, but to still benefit from the extensibility of the underlying framework. A few companies are currently working on Eclipse SmartHome–based gateways, too.

Conclusion

Eclipse SmartHome is a modern framework whose strength is a modular architecture that provides APIs tailored to the expectations of developers. Abstracting device descriptions and their semantics is the most crucial part for allowing multipurpose solutions to be built on top of the framework. Such solutions additionally benefit from the existing developer community and from the framework's customizability. As **Figure 4** shows, even smart telephone booths can be realized with it. Leveraging Java SE 8 features in the future will make it even more powerful and attractive. `</article>`

MORE ON TOPIC:

Internet of Things

LEARN MORE

- openHAB
- Eclipse SmartHome

The Device I/O API

A standard API for peripherals and low-level hardware control just arrived for Oracle Java SE Embedded.

VINICIUS SENGER

BIO

If you recall the history of Java, you might remember that in the beginning, each database had its own access API, and if you needed to migrate to another database implementation, it was necessary to learn that database's API and refactor all the database access code. To solve this problem, JDBC was launched in 1997 to specify how vendors should implement their drivers. Nowadays, JDBC is implemented by the most-popular database vendors, so this is a great success story for the Java Community Process (JCP).

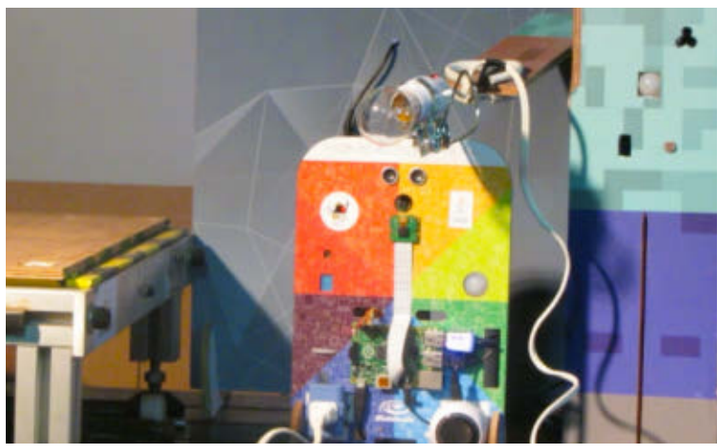
However, now we have a similar problem using Java SE in the embedded space, where there is a need to access and control many different low-level devices using interfaces such as general-purpose input/output (GPIO), inter-integrated circuit bus (I^2C), serial peripheral interface (SPI) bus, universal asynchronous receiver/transmitter (UART), and so on. The number and types of single-board computers are growing fast, and each type provides a different API for using its GPIO pins.

Naturally, for each different board, a new Java API needs to be designed. A popular example is the Pi4J project, which is a nice API for the Raspberry Pi that allows you to manipulate GPIO pins and use I^2C, SPI, and UART components. But if you want to migrate to another board, you will need to rewrite all your GPIO access code.

In May 2014, Oracle Vice President of Product Management in the Java Platform Group Henrik Ståhl announced the Device I/O API and also made some comments about overlap between the Pi4J project and the Device I/O API:

"*There is some overlap between our Device I/O and Pi4J. Pi4J is (and will remain) much richer in functionality, but was really intended for the Raspberry Pi and not for other platforms. We have been discussing with Robert Savage (Pi4J author and frequent contributor on this forum) how to handle this overlap and the current thoughts are that Device I/O may be the right place to handle the basic I/O with Pi4J as a rich layer on top. We have a similar thread going with some of the LeJOS (Java on Lego Mindstorms) community members.*"

This is all very exciting for the embedded Java community, because the Connected Limited Device Configuration (CLDC) for Java ME and also Java SE are target platforms for embedded projects with single-board computers or microcontrollers. In practice, if you have a Java ME or Java SE project that reads an



Vinicius Senger demonstrates the Device I/O API with home automation devices.

Internet of Things

I²C sensor using the Device I/O API, your code will be 100 percent portable to different target boards such as the Raspberry Pi, BeagleBone Black, UDOO, and dozens of other boards.

This article will present Device I/O API details, code for a practical project, and also some examples of industry applications.

Device I/O API Features

The Device I/O API abstraction is very complete and includes support for analog-to-digital converters (ADCs), digital-to-analog converters (DACs), AT command devices, counters, GPIO pins, I²C devices, pulse-width modulation (PWM) generators, SPI devices, UART devices, watchdog timers, and modem signal control. Beyond providing low-level control, the Device I/O API also provides support for security, concurrency, and power management, which will bring the next level of device access to Java SE.

ON THE RISE

The number and types of single-board computers are growing fast, and each type provides a different API for using its GPIO pins.

Another important thing about this API is that it is open source. Feel free to participate and collaborate; the official website is https://wiki.openjdk.java.net/display/dio/Main. There is an amazing team supporting and writing this API.

Getting Started with the Device I/O API and the Raspberry Pi

You can start playing with the Device I/O API and the Raspberry Pi today, because Oracle is writing the implementation for Linux with hard-float support and is testing the implementation with Raspberry Pi boards (based on the BCM2835 system on a chip [SoC]). You will need the following items:

- One Raspberry Pi, Model A, B, or B+
- One secure digital (SD) card with a Raspian image
- An LED or any other component you want to control
- Wires
- A protoboard (optional, but recommended)

Once you have your board up and running, you will need to download and install JDK 8 for ARM.

Additionally, you need to install Mercurial to check out the Device I/O API source code. To install it, just type the following command at your Raspberry Pi terminal:

```
sudo apt-get install mercurial
```

Then, you can get the source code by typing the command shown in **Listing 1**.

LISTING 1

```
hg clone http://hg.openjdk.java.net/dio/dev
```

Download all listings in this issue as text

You might need to install the GNU Compiler Collection (GCC) or some additional tools, but typing the following commands should be enough to build a Device I/O project for most Raspbian distributions:

```
export PI_TOOLS=/usr
export JAVA_HOME=<your jdk path>
cd dev
make
```

Now you can run some sample applications, check the API details, and start coding. The most important files you will need are the following:

- /dev/build/so/libdio.so is the compiled C native code.
- /dev/build/jar/dio.jar is the complete Java Device I/O API.
- /dev/build/jar/dio-samples.jar is a sample application using an LED and an SPI.
- /dev/samples/gpio/gpio.policy is the policy file template.
- /dev/config/dio.properties-raspberrypi specifies the default configuration for the Raspberry Pi.

Inside the dev directory, you will find the following subdirectories:

- /dev/src, which contains the Java and C native code
- /dev/test, which contains the test suite
- /dev/build, which contains all the compiled files, the linux.so library, and the .jar files
- /dev/build/deviceio, which contains compiled files with directory structure prepared to be installed in your JRE
- /dev/config, which contains the specific property configurations for Raspberry PI devices and GPIO

The src subdirectory is organized as follows:

```
|----se
|    |----classes
|    |----native
|    |----linux
|
|----share
|    |----classes
|    |----native
|    |----linux
```

The share subdirectory contains

all the base source code for the Device I/O API that can be shared with other platforms, while the se subdirectory contains specific classes and native code for the Oracle Java SE Embedded platform. The main package for the Device I/O API is jdk.dio, and you will find all the interface definitions inside share/classes/jdk/dio.

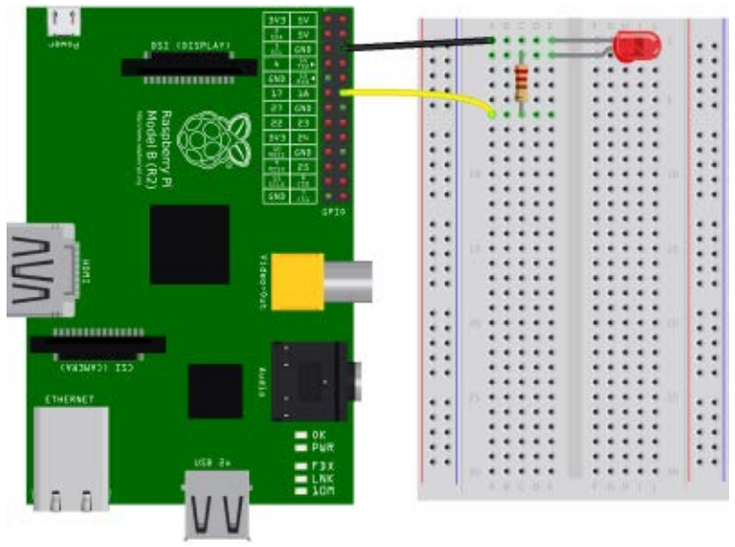


Figure 1

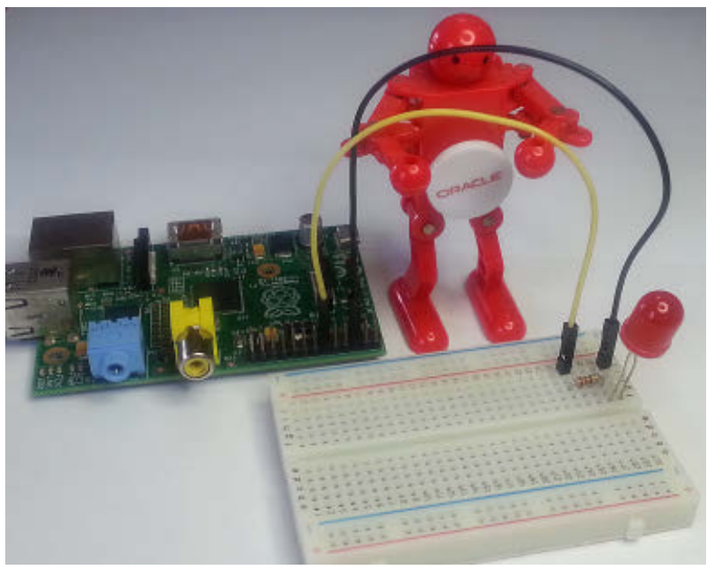


Figure 2

Writing Your First Device I/O Application

To run your first application using the Device I/O API, you will need the following:

- Any standard Java class with a main method
- java.policy, which is a file that contains the permissions configuration
- dio.jar, which contains the Java Device I/O API itself
- libdio.so, which contains native code
- dio.properties, which contains board-specific configuration (for the Raspberry Pi)

Let's write the code for using a blinking LED with an embedded version of the well-known embedded Hello World program.

In terms of hardware, you will need the following:

- A Raspberry Pi, any model
- An SD card
- A power supply (5 volts, 1 amp)
- One LED
- One resistor (220 ohms)
- A protoboard and male-female wire

You can wire your protoboard as shown in **Figure 1** and **Figure 2**.

As you can see, the code in **Listing 2** is very simple. We have the DeviceManager class as a starting point to establish a connection between the Java object named pin and the physical device, in this case, GPIO pin number 18.

To run the application, you must run the command with the arguments shown in **Listing 3**.

To make things easy, you can install Device I/O files in your Java runtime environment (JRE), because we have the directory /dev/build/deviceio prepared to install dio.jar and libdio.so in your JRE, and it includes libdio.so in lib/ext/arm and dio.jar in lib/ext. To do this, just type the command shown in **Listing 4**. Then, you won't need to specify dio.jar and the additional library path (see **Listing 5**).

LISTING 2 LISTING 3 / LISTING 4 / LISTING 5

```
package javamagazine.dio;

import jdk.dio.DeviceConfig;
import jdk.dio.DeviceManager;
import jdk.dio.gpio.GPIOPin;
import jdk.dio.gpio.GPIOPinConfig;

public class LedBlink {

  public static void main(String[] args) throws Exception {
    GPIOPin pin = null;
    pin = (GPIOPin) DeviceManager.open(18);
    System.out.println("Blinking LED");
    for (int x = 0; x < 20; x++) {
      pin.setValue(x % 2 == 0);
      Thread.sleep(1000);
    }
    pin.close();
  }
}
```

Download all listings in this issue as text

Your jar file must specify a manifest .mf with the Java class main method. NetBeans IDE does that automatically.

Tip: If you interrupt your application without executing the pin.close() method, the pin will remain in exported state and you will not be able to use it again until you "unexport" it manually by running the command shown in **Listing 6**.

We could also specify additional pin configurations beyond the pin number, for example, signal direction (for instance, input only), mode (for instance, pull-up or pull-down), and so on. Let's look at the sample code in **Listing 7**, which could replace the code inside the main method in **Listing 2**.

Because we already know how to use libdio.so and dio.jar as external libraries and as part of our JRE, let's look inside the gpio.policy policy file and the Raspberry Pi properties file.

The security policy allows the Java Virtual Machine (JVM) to access a restricted set of devices that you must specify in the gpio.policy file. **Listing 8** shows an example policy file that grants all the permissions for GPIO pin 18.

To finish, let's see what the Raspberry Pi properties file looks like (see **Listing 9**). This file declares and configures all the devices we have on our board, from pins to watchdog timers, UART devices, and so on. The device numbers will vary according to the board or SoC you are using, so you must check the data sheet and manuals if you are going to use another board or SoC.

PORTABLE CODE

In practice, if you have a Java ME or Java SE project that reads an I²C sensor using the Device I/O API, **your code will be 100 percent portable to different target boards** such as the Raspberry Pi, BeagleBone Black, UDOO, and dozens of other boards.

Device I/O API Demos

The Device I/O source code includes a sample application that uses an LED and an SPI implementation for the MCP3008 ADC. You will find the generated .jar file in the build/jar directory:

```
/dev/build/jar/dio-samples.jar
```

To run this application, you can use the command shown in **Listing 10**. You should see your LED blinking on port 18.

LISTING 6 | LISTING 7 | LISTING 8 | LISTING 9 | LISTING 10 | LISTING 11

```
echo 18 > /sys/class/gpio/unexport
```

Download all listings in this issue as text

The Device I/O API: Beyond Pins

The Device I/O API supports much more than GPIO pins; most of the popular device communication standards are supported. **Listing 11** shows the code for reading a GPS sensor using UART communication through Raspberry Pi pins 8 (RX) and 10 (TX).

The Device I/O API is very complete and covers most low-level devices, from PWMs to DACs to ADCs. The way you establish a connection with various devices is quite similar. For each type of device, you need to use the specific interface shown in **Table 1**.

The API, DIY, and the Industry

We are living in an exciting time for the embedded space as a result of

INTERFACE	DESCRIPTION
jdk.dio.adc.ADCChannel	ANALOG-TO-DIGITAL CONVERTER SUPPORT
jdk.dio.atcmd.ATDevice	MODEM/CELLULAR SUPPORT
jdk.dio.counter.PulseCounter	PULSE COUNTER SUPPORT
jdk.dio.dac.DACChannel	DIGITAL-TO-ANALOG CONVERTER SUPPORT
jdk.dio.gpio.GPIOPin	DIGITAL PIN SUPPORT
jdk.dio.i2cbus.I2CDevice	I^2C BUS SUPPORT
jdk.dio.power.PowerManaged	POWER MANAGEMENT SUPPORT
jdk.dio.pwm.PWMChannel	PULSE-WIDTH MODULATION SUPPORT
jdk.dio.spibus.SPIDevice	SPI SUPPORT
jdk.dio.uart.UART	UART RX AND TX COMMUNICATION SUPPORT
jdk.dio.watchdog.WatchdogTimer	SUPPORT FOR WATCHDOG TIMER THAT FORCES RESTART

Table 1

decreases in hardware prices, the availability of open source hardware, the Internet of Things (IoT), startups, a resurgence of "do it yourself" (DIY) projects, and the Maker Movement. There is definitely an explosion of possibilities that goes much further than the mobile phone market. You can create a gadget to solve a particular problem using sensors, motors, lights, and so on, and doing that is becoming popular. As Arduino's Mássimo Banzi said:

". . . hardware becomes like a piece of culture that you share and you build upon, like it was a song or a poem."

The hardware industry now has many small, midsize, and startup companies; it is not like in the past when you could count on your fingers the number of hardware companies. It's predictable that many successful products will comprise custom hardware, code from the community, and the cloud.

Java EE is very reliable and popular for server-side development, big data, the web, cloud, and service-oriented architecture (SOA), and there are now many standards in use. Now it's time to work with standards and implementations for embedded devices. The Device I/O API is definitely an important step for the future of embedded Java, similar to what JDBC was for the Java platform.

Conclusion

This is just the beginning for embedded Java. Much more will come from the community, companies, and universities. If we look again to the Java EE platform, after JDBC many persistence frameworks were created and now we have a standard for persistence frameworks: Java Persistence API. The same might happen in the embedded space; once the standard API for Device I/O control is done, we might see more and more frameworks for devices in the embedded space, making it even easier to work with hardware, boards, devices, and embedded Java.

Now—what will you create with Oracle Java SE Embedded and the Device I/O API? `</article>`

MORE ON TOPIC:

Internet of Things

LEARN MORE

- OpenJDK page for the Device I/O project
- Video: *Accessing HW Devices Using Java ME 8 Device I/O API*
- Download the code shown in the video demo

JOSÉ PEREDA

BIO

Building Castles in the Sky

Use JavaFX 3D to model historical treasures and more.

In the last few years, reconstruction algorithms for 3-D real objects have received significant attention, not only for artificial intelligence but also as tools for applications in fields such as medicine, manufacturing, robotics, and archeology that require 3-D modeling of real environments.

When geomatic engineering is applied to the reconstruction of cities, historic buildings, or museums, digital documentation—such as an inventory of the buildings and sites—is generated. Then, with the use of visualization tools, georeferenced and scaled virtual tours can be created, allowing you to "walk" through the areas while sitting in front of your computer.

This article briefly presents a JavaFX project that uses 3-D models provided by Óscar Cosido Cobos and his team at the Training and Employment Center (CEFEM) for the City Council of Santander in Cantabria, Spain. Cosido and his team were able to generate all the digital documentation, including the main 3-D objects, for the Menéndez Pelayo Library, a historical building opened to the public in 1923. They did this using reverse engineering and a hybridization of technologies and techniques such as geographic information systems (GIS), artificial vision, terrestrial photogrammetry, and flight with microdrones. Then they postprocessed the data for point cloud optimization, orthorectified the images, and texturized the 3-D model.

GET A VISUAL
JavaFX 3D can be used as the **visualization framework** for displaying a 3-D model and navigating through it.

This article focuses on the visualization tool, JavaFX 3D, which was used as the visualization framework for displaying the 3-D model and navigating through it. By using the JavaFX 3D API, you can develop your own applications for targeting and customizing some of the usual capabilities of commercial software.

Reverse Engineering Techniques

Neither digital images nor analog images are metric. In other words, we cannot derive measurements of the objects that are displayed nor can we derive the distances between them. We can visualize, interpret, and analyze, but we cannot exploit the objects metrically. Computer stereo vision is the extraction of 3-D information from digital images. Through multiple images of a scene taken from different viewpoints, the 3-D characteristics of the scene can be extracted.

Photogrammetry is a technique for obtaining two- or three-dimensional metric information from photographic images. Photogrammetry is, therefore, a technology that can provide dimensional aspects when using images to reconstruct a given scene. By using oblique processing techniques for 3-D reconstruction, an object is reconstructed by identifying (in at least two images, and preferably three) the points and lines that form the physiognomy of the object.

Usually, a 3-D modeling project starts with the topographic work, in which total stations are employed to set several main control points and form a network. To create the graphic and photographic documentation of an object, an essential step is the calibration of the camera. For

PHOTOGRAPH BY BOB ADLER/ GETTY IMAGES

that, a template such as the one shown in the left side of **Figure 1** is used. This process involves calculating the transformation that relates the instrumental coordinates with the photocoordinates. The right side of **Figure 1** shows a shot from the camera.

Once everything is in place, it's time to shoot photos. This requires the object to be divided into smaller parts to shoot. In addition to the images taken around the object at ground level, oblique and zenith aerial images taken from flying a drone over the object can be used.

Finally, once the characteristic points of the images are obtained, a triangulation has to be performed. The result will be a series of 3-D points corresponding to the surfaces of the object. For this phase, a combination of Voronoi diagrams and Delaunay triangulation techniques is used.

The last step for obtaining the 3-D model of an object is to obtain the corresponding points in the rest of the images. By knowing the points of the base image, it is possible to find the epipolar lines in other images using the fundamental matrix and a correlation between the intensity of a point in the image and its surroundings. **Figure 2** shows the white model (a model in the early stages, which is not fully colored or texturized yet, but already provides a clear view of the modeled object) of the inside of the Menéndez Pelayo Library building.

3-D Modeling with JavaFX

The JavaFX 3D graphics APIs provide a general-purpose 3-D graphics library for the JavaFX platform. Since the launch of Java SE 8, a true 3-D scene is available with lights, cameras, materials, and basic 3-D objects. Access to dedicated graphics processing unit (GPU) hardware allows 3-D rendering with great performance.

Besides the official documentation, there are many good resources out there for learning about JavaFX 3D, including the book I coauthored (*JavaFX 8: Introduction by Example*) and the new *Pro JavaFX 8*, as well as many blogs and presentations from conferences. In addition, there is the 3DViewer application, which you can find in the OpenJFX project, as well as other experimental material under the project's Toys folder.

But the best way to learn is to get your hands dirty hacking some code. So now we'll explore a simple application intended to be a multimodel 3-D viewer and manipulator. You can find all the code and some simple 3-D models that were donated by Cosido and his team in my repository. At the end of the article, we'll use their real models to briefly show what can be achieved.

Step 1. Importing the model.
The JavaFX 3D graphics APIs provide nodes that represent 3-D objects from a common base class, Shape3D, and four concrete subclasses: Sphere, Box, and Cylinder, which are predefined 3-D primitive shapes, and MeshView, a user-defined shape that defines a surface with the specified 3-D mesh data.

The Mesh abstract class and its TriangleMesh concrete subclass are where the geometric information of a 3-D shape is stored, including the 3-D coordinates of all the vertices, two-dimensional texture coordinates, faces, and optional face smoothing groups.

Material is an abstract class with one concrete class called PhongMaterial that controls the surface material. The drawMode enum has two constants: LINE for a wireframe drawing or FILL for a solid drawing. cullFace determines which side of the face is not rendered (BACK or FRONT), while NONE allows

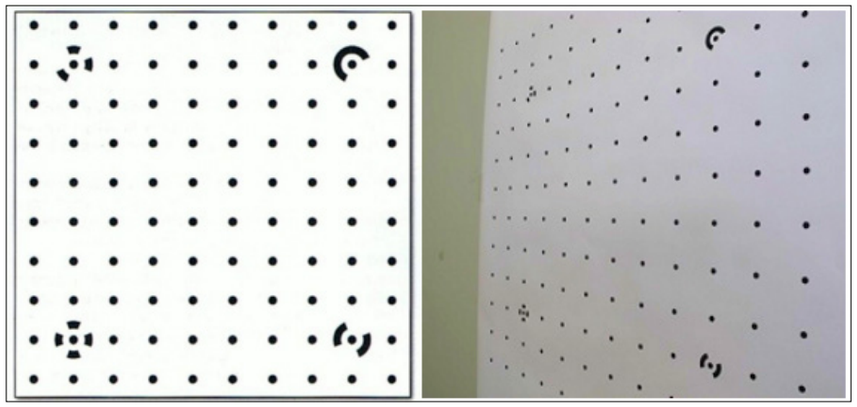


Figure 1

Figure 2

both sides to be rendered. Note that the same material is shared among multiple Shape3D nodes.

Complex 3-D models—such as those shown in **Figure 3**, which shows, from left to right, a wire-framed model, a solid model, and a solid model with texture—can't be generated easily with just 3-D primitives, because they are made of complex meshes. In order to display these models in a JavaFX scene (or SubScene), we need a way to import all the information and reconstruct the JavaFX model. Usually this is accomplished by loading the meshes that define those models.

There are no loaders for any of the common 3-D file formats in the JavaFX 3D APIs, but some of them (OBJ, Maya, 3D Studio Max, Collada, and FXML) can be imported into a JavaFX scene by the importers available with the 3DViewer application.

We'll focus on the common open format OBJ and its MTL material format. **Listing 1** shows an excerpt of the chair.obj file for the model shown in Figure 3, where a chair from the library is shown in a JavaFX subscene after the file is imported and different options are selected.

In **Listing 2**, you can see how an .obj 3-D model can be imported using ObjImporter from the 3DViewer application to create objects based on triangular meshes. Note that you can also select PolyObjImporter, in case you have polygonal meshes with

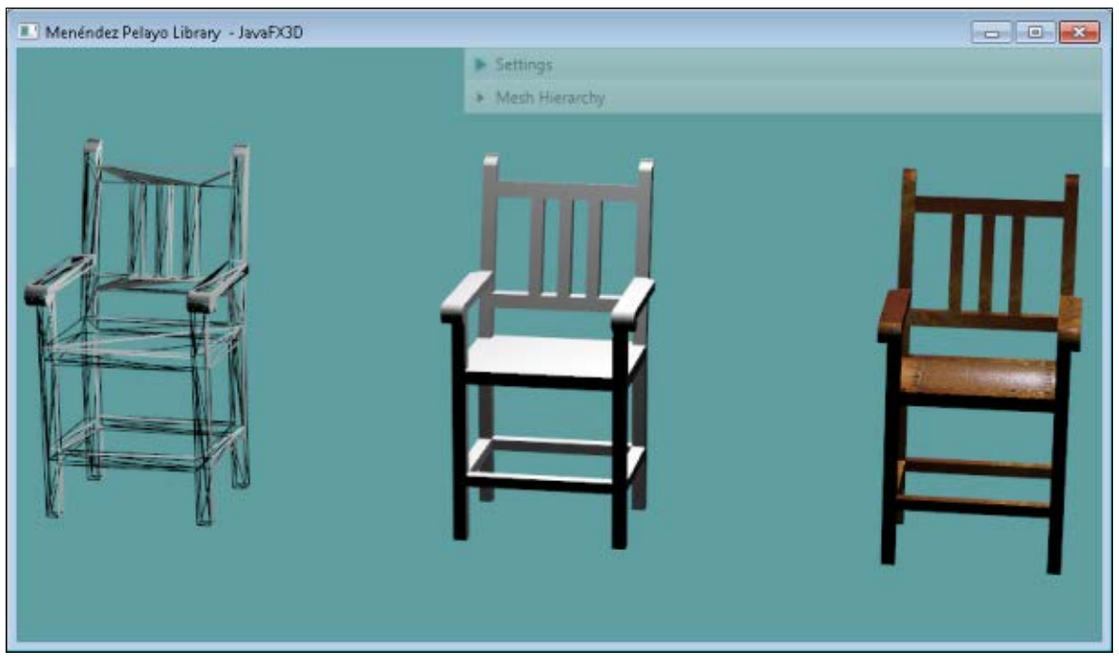


Figure 3

LISTING 1 LISTING 2

```
mtllib chair.mtl

v  -2.0226 -0.3300 -0.3345
v  -2.0226 0.0000 -0.3345
...
vn -1.0000 0.0000 -0.0000
vn 1.0000 0.0000 -0.0000
...
vt 1.0000 1.0000 0.0000
vt 0.0000 1.0000 0.0000

g Chair01
usemtl Material__64
s off
f 1/1/1 2/2/1 3/3/1 4/4/1
usemtl FrontCol
f 5/5/2 4/6/2 3/7/2
f 5/8/2 6/9/2 7/10/2
...
```

Download all listings in this issue as text

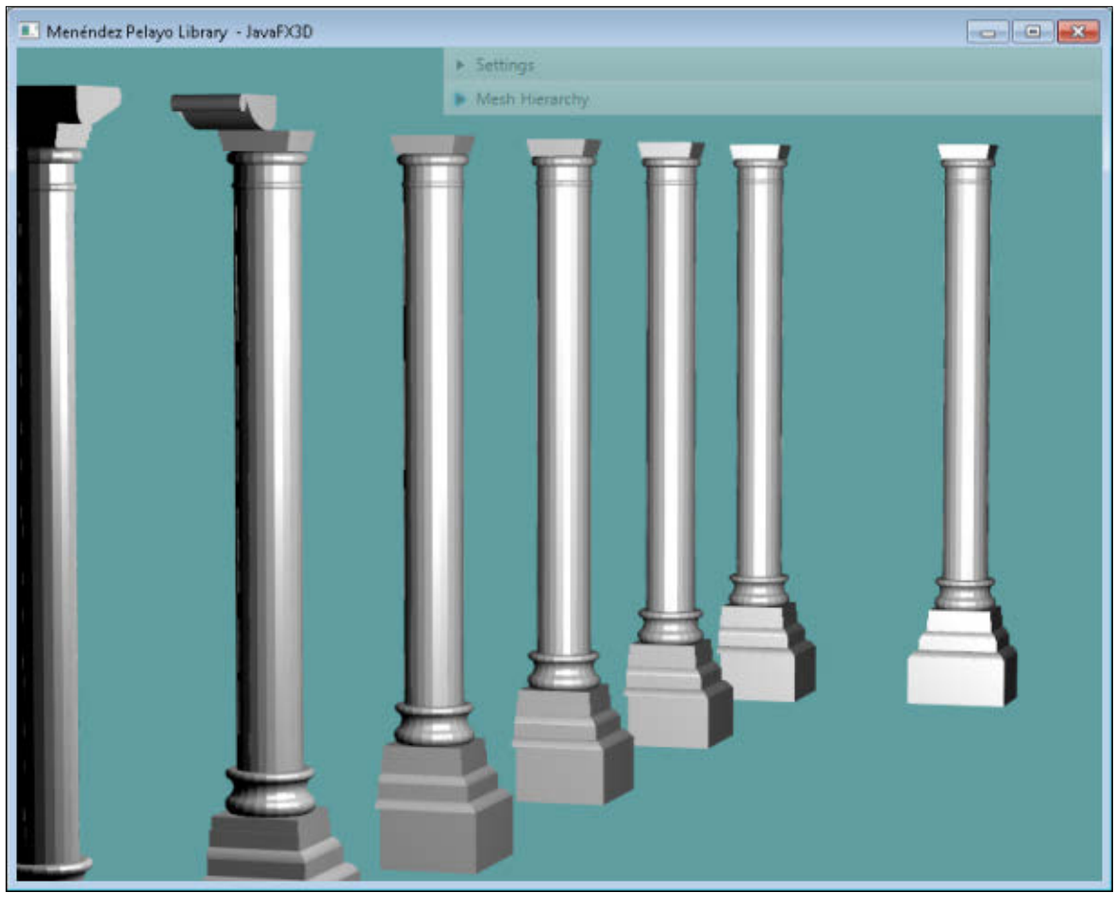


Figure 4

faces consisting of more than four sides. This is the case in **Figure 4**, which shows several columns of the library, which are imported with PolygonMeshView after the file lib_Columns.obj is imported.

Step 2. Manipulating the models. When adding different models to a multimodel application, some of the models might require scaling, translating, or rotating to locate them in the correct position. For this, an Affine class is provided for every model, as shown in **Listing 3**, in which the chair model is scaled down and translated to its final location.

Using the prepend methods causes the transformations to take place in the provided order because, from a mathematical point of view, operations are performed on the left of the transformation matrix (this is called *premultiplication*) and referred always to the global coordinate system. In **Listing 3**, the chair is first translated using its actual dimensions (because the original height of the chair's model is 8.43 m and its minimum Y coordinate is -4.13 m, it's moved up 4.13 m), and then it is scaled down with a 0.14 factor to a final height of 1.20 m.

Note that by using append methods instead, operations would be performed on the right of the matrix, the so-called *postmultiplication*. This means that new transformations are performed relative to the current object's coordinate origin. While this is the preferred technique in JavaFX, both multiplication techniques are implemented in the JavaFX 3D APIs, so you can use what is best suited for your problem.

LISTING 3 **LISTING 4**

```
public class Library {
...
   private final ObservableList<Model3D> models=
         FXCollections.observableArrayList();

   public Library(){
      Model3D modelBuildingOut =
            new Model3D("lib_Walls.obj","Library Walls");
      modelBuildingOut.importObj();
      models.add(modelBuildingOut);

      Model3D modelChair1=new Model3D("chair.obj","Chair - 1");
      Affine affineChair=new Affine();
      affineChair.prependTranslation(2,4.13,-12);
      affineChair.prependScale(0.14,0.14,0.14);
      modelChair1.setAffine(affineChair);
      modelChair1.importObj();
      modelChair1.setPickable(true);
      models.add(modelChair1);
      ...
   }
}
```

Download all listings in this issue as text

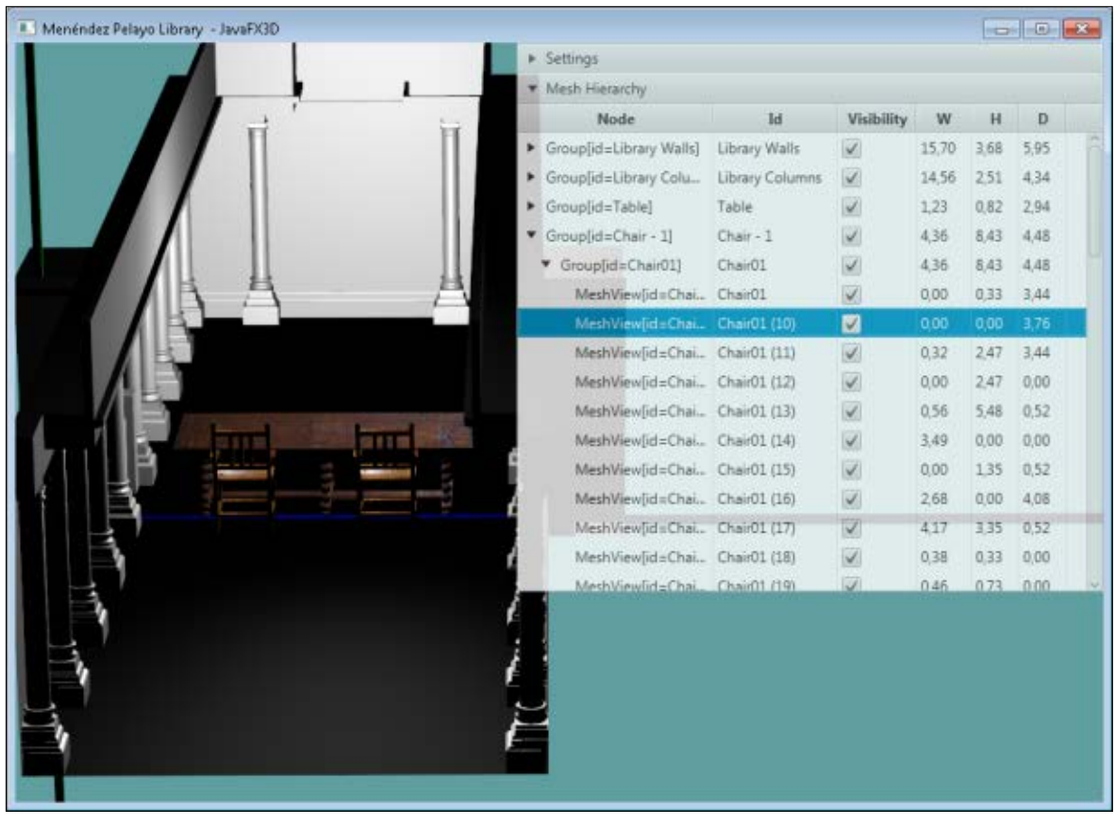


Figure 5

In the ContentModel class shown in **Listing 4**, a subscene is created containing a main group, root3D, with lights and a second group named group3D. This one contains the axes and a third group with as many groups as models, each of them containing their meshes loaded. Note that Java SE 8 streams are used to create these groups based on a sequential stream of the 3DModel collection, and note also that a Comparator can be used to sort the way the meshes are added.

A TreeTableView is used to display the hierarchy of groups and meshes of the final model, as shown in **Figure 5**.

Step 3. Moving the models. Several listeners are added to the subscene, so it becomes responsive to user input, mainly input provided through a mouse. The main responses consist of camera movements. For that, MOUSE_PRESSED, MOUSE_DRAGGED, and MOUSE_RELEASED mouse events are listened to in order to trigger camera rotations (left mouse button pressed), zooming (right mouse button pressed), or panning (mouse wheel pressed), with some modifications of that behavior, such as slowing down (CTRL key pressed) or speeding up (ALT key pressed).

MOUSED_PRESSED events can also be used to select a model from the scene and move it around. For that, event.getPickResult() is the key for obtaining the selected node on the scene. The PickResult class, which is part of the javafx.scene.input package, allows mouse, gesture, or touch events to return information about

LISTING 5 **LISTING 6**

```
private final EventHandler<MouseEvent> mouseEventHandler =
    event -> {
    if (event.getEventType() == MouseEvent.MOUSE_PRESSED) {
      String pickedModel=getModelId(
            event.getPickResult().getIntersectedNode());
      if(!pickedModel.isEmpty()){
        if(models.stream()
            .filter(Model3D::isPickable)
            .filter(m->m.getId().equals(pickedModel))
            .anyMatch(m->{
              modelMeshes=m.getMeshesList();
              return true;
            })){
            cursor.set(Cursor.CLOSED_HAND);
        }
      }
    } else if (event.getEventType() ==
        MouseEvent.MOUSE_DRAGGED) {
      mousePosX = event.getSceneX();
      mousePosY = event.getSceneY();
...
      // update model location
...
    } else if (event.getEventType() ==
        MouseEvent.MOUSE_RELEASED) {
        cursor.set(Cursor.DEFAULT);
    }
  };
```

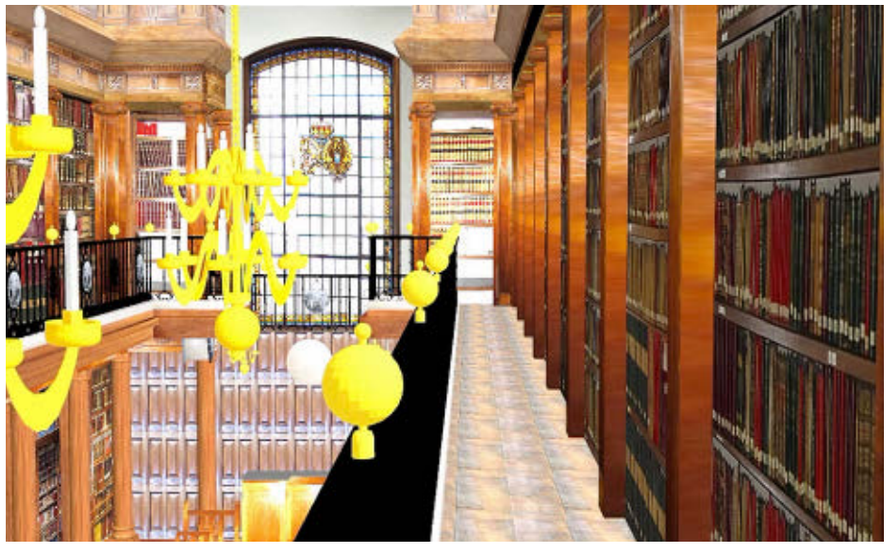
Download all listings in this issue as text

the user's pick result, such as the intersected node, the intersected point in the local coordinate system of the picked node, or the distance from the camera to the intersection. **Listing 5** shows how the pick result can be used to move models around the scene. Note that models are moved freely without checking possible intersections with other models.

In the video below, you can see a demo of the application, including several implemented features that are worth mentioning:

- Picking can be used to identify the node and select its tree item in the TreeTableView.
- Selecting a mesh tree item on the TreeTableView highlights the real mesh.
- In the settings titled pane, models can be marked as pickable or not pickable, so they can be moved around the scene or prevented from being moved.
- While moving the camera, a balance is required between providing detailed visualization and ensuring usability, especially when dealing with big models. For that reason, textures are unloaded while rotating, panning, or zooming the scene, as shown in **Listing 6**.

Missing features in the application, for example, light or camera manipulation, can be added easily. At this point, you are invited to contribute creating pull requests with more-advanced features.



José Pereda gives an overview of his article and demos the JavaFX 3D tool.

Loading the Real Model

Finally, we'll use the developed application to load and visualize the complex model of the Menéndez Pelayo Library created by Óscar Cosido and his team.

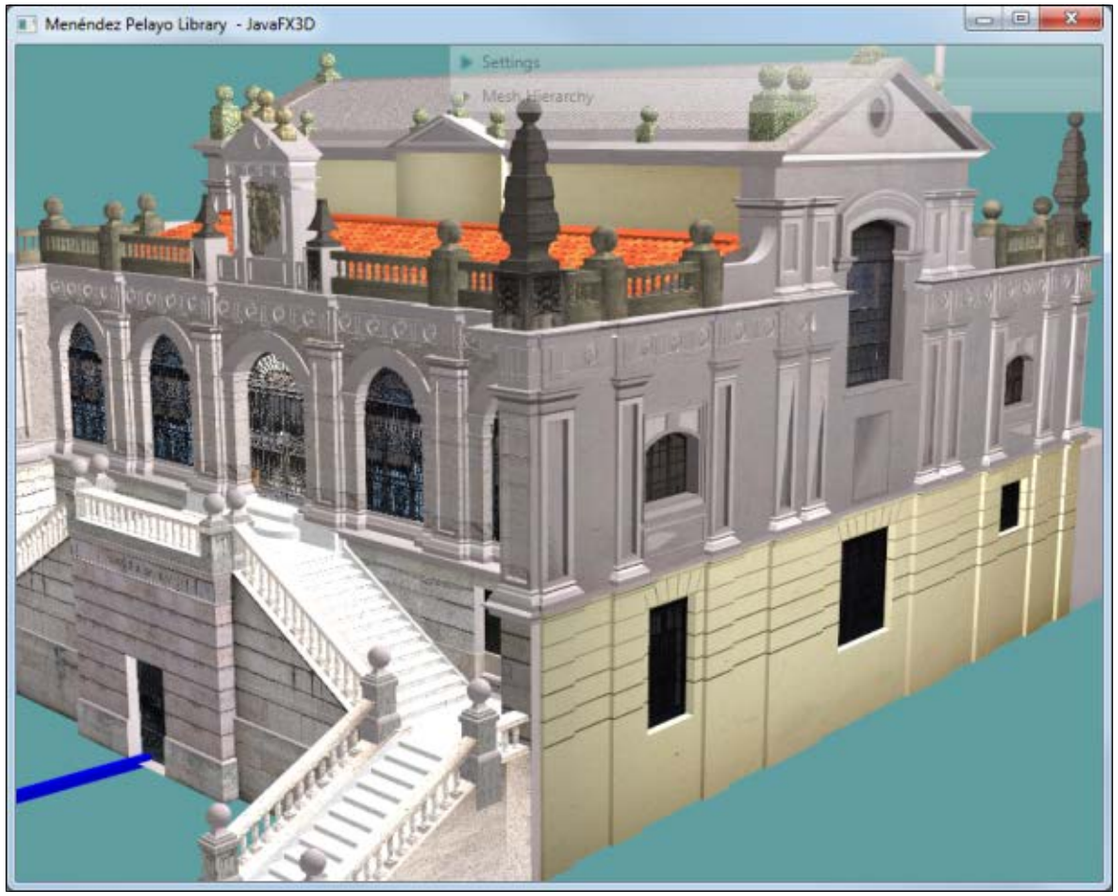


Figure 6

Figure 6 shows the outside view of the building, with 10,556 meshes and 1.5 million triangles, which results in a 132 MB .obj file and 32 MB in texture images. **Figure 7** shows the interior of the library, with 2,841 meshes and 2.4 million triangles, resulting in a 143 MB .obj file and 90 MB in texture images.

HELP IS HERE

New capabilities in Java SE 8 (mainly lambdas and streams) **help to deal with** big hierarchical models.

Conclusion

JavaFX 3D technology is mature enough to enable the development of tools for the geometric processing of complex models obtained by reverse engineering. It provides a way to navigate a virtual model and establishes the basis for the future development of tools for

Figure 7

managing 3-D objects in a virtual world as part of a 3-D information system.

New capabilities in Java SE 8 (mainly lambdas and streams) really help to deal with big hierarchical models. Still, work has to be done to provide JavaFX 3D APIs for building more-powerful tools for rendering, lighting, shadowing, model movement, and the usual features one can find in 3-D modeling software. `</article>`

Acknowledgements

The author would like to thank Óscar Cosido and the students who attended the "GIS and 3D Heritage Modeling" workshop organized by the City Council of Santander. Also, special thanks goes to Esteban Sainz Vidal, director of the CEFEM, for the support and help he offered.

LEARN MORE

- OpenJFX project
- Menéndez Pelayo Library

DAVID GILBERT

BIO

A Bridge from Java 2D to JavaFX

Profit from the easy migration path provided by FXGraphics2D.

The JavaFX Canvas is an image node that can be added to a scene to display high-quality text and graphics. It has an "immediate mode" drawing API that resembles the HTML5 Canvas API, and it provides a feature set that is comparable to that provided by the Java 2D API, the standard vector graphics API in Java SE since JDK 1.2. The Canvas node provides a powerful additional capability for developers creating JavaFX user interfaces.

As developers migrate their applications from Swing to JavaFX, though, they might encounter a problem. The JavaFX Canvas API is different than the Java 2D API and does not provide a compatibility layer. Consequently, existing code based on the Java 2D API can require significant rework.

My company, Object Refinery Limited, faced this exact problem with our library, JFreeChart, which uses the Java 2D API exclusively for rendering. To solve this problem we created the FXGraphics2D project, an open source "bridge" from the Java 2D API to JavaFX. It provides an easy migration path for applications that are based on the Java 2D API, and it avoids the need to rewrite already working code.

This article provides an overview of FXGraphics2D, including a demo and some performance benchmarking. **Figure 1** shows the JFreeChart library being used to render a chart on a JavaFX Canvas via FXGraphics2D.

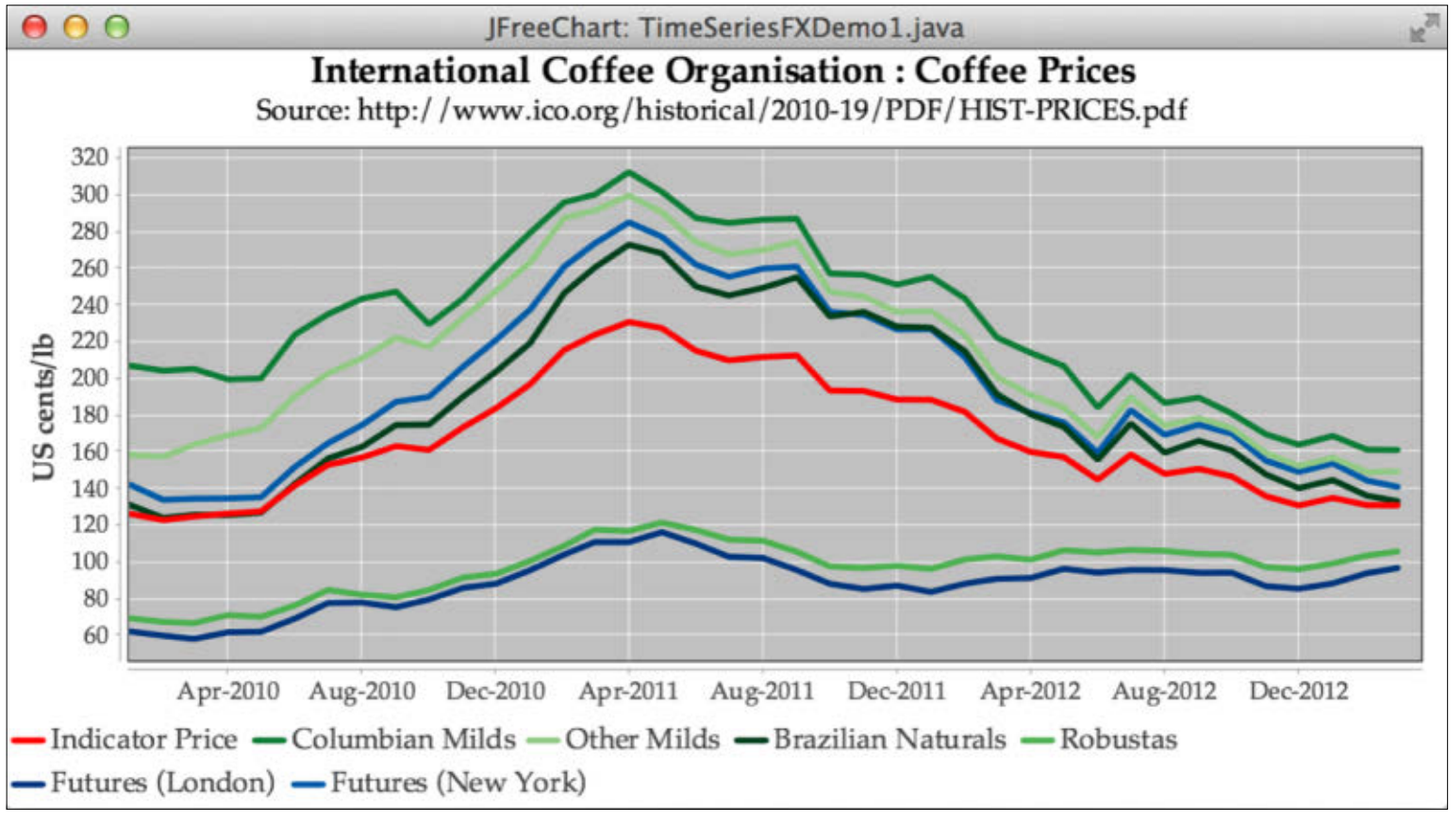


Figure 1

About the FXGraphics2D Project

The FXGraphics2D project is small and open source. The library consists of a single class (FXGraphics2D.java)

that extends the Java 2D API's Graphics2D class and is licensed using a very liberal BSD-style license. We chose this license to ensure that there are no obstacles for anyone who wants to use this code. The source code is hosted at GitHub. The download includes additional files, including some demo applications.

About the Java 2D API

The Java 2D API is the vector graphics API that has been part of the Java standard library since JDK 1.2 was released in 1999. Among other things, it is the graphics engine that powers Swing. The Java 2D API is a transparent, feature-complete, and fast API that is available to all Java application developers, on both the client side and the server side, and on all Java-supported platforms.

Graphics that are compliant with the Java 2D API can be easily exported directly as images in Portable Network Graphics (PNG), Joint Photographic Experts Group (JPEG), and other formats using the Java Image I/O API. With appropriate third-party libraries—for example, Batik, iText, and JFreeSVG—it is also possible to export to Scalable Vector Graphics (SVG), Portable Document Format (PDF), and other formats easily.

KEEP YOUR CODE

Migrating to JavaFX can be a lot easier if you don't need to throw away a lot of existing code. This article demonstrates that **FXGraphics2D is a robust solution** that allows you to keep your tried-and-tested Java 2D code and incorporate it into JavaFX applications.

During the last decade and a half, a lot of code has been written to use the Java 2D API—open source examples include JFreeChart, JGraphX, and JLaTeXMath, and there are many more.

Retained Mode Versus Immediate Mode

In JavaFX, the graphics model is scene-based, which means objects are added to a scene and rendering is managed by the JavaFX runtime. Objects can be updated at any time and, because JavaFX retains state information, the scene can be automatically rendered again without programmer intervention.

JavaFX is sometimes referred to as being a *retained mode* API. Java 2D, on the other hand, is an *immediate mode* graphics API; it does not retain any state information about shapes or their properties (color, line style, and so on) after they have been rendered. Immediate mode rendering requires less memory space when rendering complex graphics that have many elements. The JavaFX Canvas node introduces an immediate mode rendering option for JavaFX applications that require it.

A Demonstration

To demonstrate FXGraphics2D, we will use the open source JLaTeXMath library, whose stated main purpose is "to display mathematical formulas written in LaTeX." JLaTeXMath renders its output via the Java 2D API, making it a perfect candidate for use with FXGraphics2D. Our simple demo application will produce the output shown in **Figure 2**.

You can download JLaTeXMath here. For this demo, we are using jlatex-1.0.3.jar. In **Listing 1** you can see the code for our JavaFX demo application. There are several important things to note in the code.

In the start() method, the first thing we do is preload the fonts that are required by JLaTeXMath. These fonts are included in the jlatex-1.0.3.jar file, but they are not available to JavaFX until we load them using Font.loadFont().

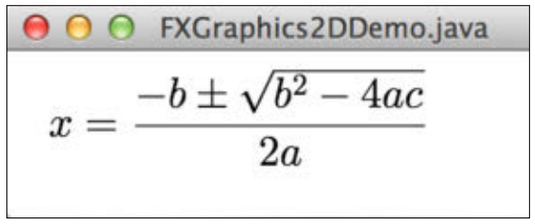


Figure 2

We create an instance of TeXFormula using the following string to define the formula that we want JLaTeXMath to render:

```
"x=\\frac{-b \\pm \\sqrt
{b^2-4ac}}{2a}"
```

If you are familiar with LaTeX, you will understand what this somewhat-cryptic string of characters specifies. If not, but you are curious about LaTeX, here is a good starting point.

Next, we create an instance of FormulaCanvas (a subclass of the JavaFX Canvas class; see **Listing 2**) that handles the creation of an FXGraphics2D instance and manages the rendering of a TeXFormula.

For this rendering, we create a TeXIcon from the TeXFormula that was passed to the constructor. This icon object can be used easily in Swing because it implements the Icon interface, but for our (JavaFX) purposes, a more interesting item is the Box object that is returned by the TeXIcon getBox() method. Box is an abstract class in JLaTeXMath that declares the fol-

lowing draw() method:

```
public abstract void draw(
Graphics2D g2, float x,
float y);
```

This method renders the content of the Box—the graphical output derived from the TeXFormula—at a specific (x, y) location. Now, because this content is drawn to an arbitrary Graphics2D instance, we have an opportunity to render the formula to a JavaFX Canvas simply by supplying an instance of FXGraphics2D to this draw() method.

The FormulaCanvas class also takes care of creating the FXGraphics2D instance and connecting it to its own GraphicsContext. This wiring needs to be done only once, in the constructor.

Canvas drawing is triggered when the width or height properties are modified, because we based our FormulaCanvas on the resizable canvas example posted online by Dirk Lemmermann. For this demo, the resizing capability is not strictly required, because the output is static. However, the technique is useful to know about.

The actual canvas drawing is handled by the code shown in **Listing 3**. Two things happen here. First, a clearRect() call is performed for the whole of the canvas. This is very important, because in JavaFX, drawing operations on the canvas are not executed immediately but are queued for later execution. When the canvas surface is cleared in this way, anything that was previously in the queue can be discarded because, logically, it can no longer affect the output. Without the clearRect() call, the drawing queue continues to grow—you would want this to happen only if you were adding to the output rather than regenerating it.

Second, the code in **Listing 3** draws the content of the box (our formula in the demo) to the FXGraphics2D instance (g2). Here, JLaTeXMath will generate a sequence of Java 2D API calls that the FXGraphics2D instance will translate directly to the equivalents for the JavaFX Canvas (actually, the GraphicsContext for the Canvas). This is where FXGraphics2D does its job. If you didn't already run the demo, please try it now.

The demo requires that you have the two jar files, jlatexmath-1.0.3.jar and fxgraphics2d-1.1.jar, on your classpath. First arrange the files in a working directory as follows:

```
lib/jlatexmath-1.0.3.jar
lib/fxgraphics2d-1.1.jar
src/demo/FXGraphics2DDemo.java
src/demo/FormulaCanvas.java
```

LISTING 1 LISTING 2 / LISTING 3

```
1/**
2 * A demo using FXGraphics2D and JLaTeXMath. Requires jlatex-1.0.3.jar.
3 */
4public class FXGraphics2DDemo extends Application {
5
6   @Override
7   public void start(Stage stage) throws Exception {
8      Font.loadFont(getClass().getResourceAsStream(
9"/org/scilab/forge/jlatexmath/fonts/base/jlm_cmmi10.ttf"), 1);
10      Font.loadFont(getClass().getResourceAsStream(
11"/org/scilab/forge/jlatexmath/fonts/maths/jlm_cmsy10.ttf"), 1);
12      Font.loadFont(getClass().getResourceAsStream(
13"/org/scilab/forge/jlatexmath/fonts/latin/jlm_cmr10.ttf"), 1);
14
15      TeXFormula f = new TeXFormula(
16"x=\\frac{-b \\pm \\sqrt {b^2-4ac}}{2a}");
17      FormulaCanvas canvas = new FormulaCanvas(f);
18      StackPane stackPane = new StackPane();
19      stackPane.getChildren().add(canvas);
20       // Bind canvas size to stack pane size.
21      canvas.widthProperty().bind(stackPane.widthProperty());
22       canvas.heightProperty().bind(stackPane.heightProperty());
23       stage.setScene(new Scene(stackPane));
24       stage.setTitle("FXGraphics2DDemo.java");
25       stage.setWidth(300);
26       stage.setHeight(100);
27       stage.show();
28   }
29
30   public static void main(String[] args) {
31     launch(args);
32   }
33
34}
```

Download all listings in this issue as text

Create an empty directory named build alongside the lib and src directories, and then compile the demo using the following command:

```
javac -sourcepath src -cp
lib/jlatexmath-1.0.3.jar:
lib/fxgraphics2d-1.1.jar
-d build
src/demo/FXGraphics2DDemo.java
```

. . . and run the demo with

```
java -cp build:
lib/jlatexmath-1.0.3.jar
demo.FXGraphics2DDemo
```

Performance

A key question that you might already be asking is "how does FXGraphics2D perform?" There are two aspects to consider. First, there is the overhead of the on-the-fly translation between Java 2D API calls and the equivalent JavaFX Canvas API calls—surely there will be some cost here. Second, there is the impact of using the JavaFX rendering pipeline versus the Java 2D pipeline. Could this bring some gains?

It would take some careful benchmarking to separately measure these two aspects and, for now, I have not done that. The benchmark used here compares the performance of rendering directly to a Swing JPanel versus rendering to a JavaFX Canvas via FXGraphics2D—and so both the translation overhead and the graphics engine change are combined in one measure.

We would expect the translation overhead to be minimal in terms of CPU time—particularly as a percentage of overall rendering time—and small in terms of memory space. The impact on memory usage is likely to be the greater of the two overheads, since the calling code will create Java 2D objects (shapes, paths, colors, strokes, fonts, and so on) that will, in many cases, be copied to equivalent JavaFX objects. For most applications, this should not be a big issue.

The switch from the Java 2D rendering pipeline to the JavaFX rendering pipeline is more interesting. I found surprisingly little information online regarding the relative performance of these graphics engines. You might expect the newer JavaFX code to be faster, but bear in mind that the Java 2D API has been the focus of a good deal of optimization over the years. Moreover, JavaFX is required to perform one or two tricks (for example, shadow effects) that the Java 2D API is not required to perform. This can restrict the available options for optimization in certain cases.

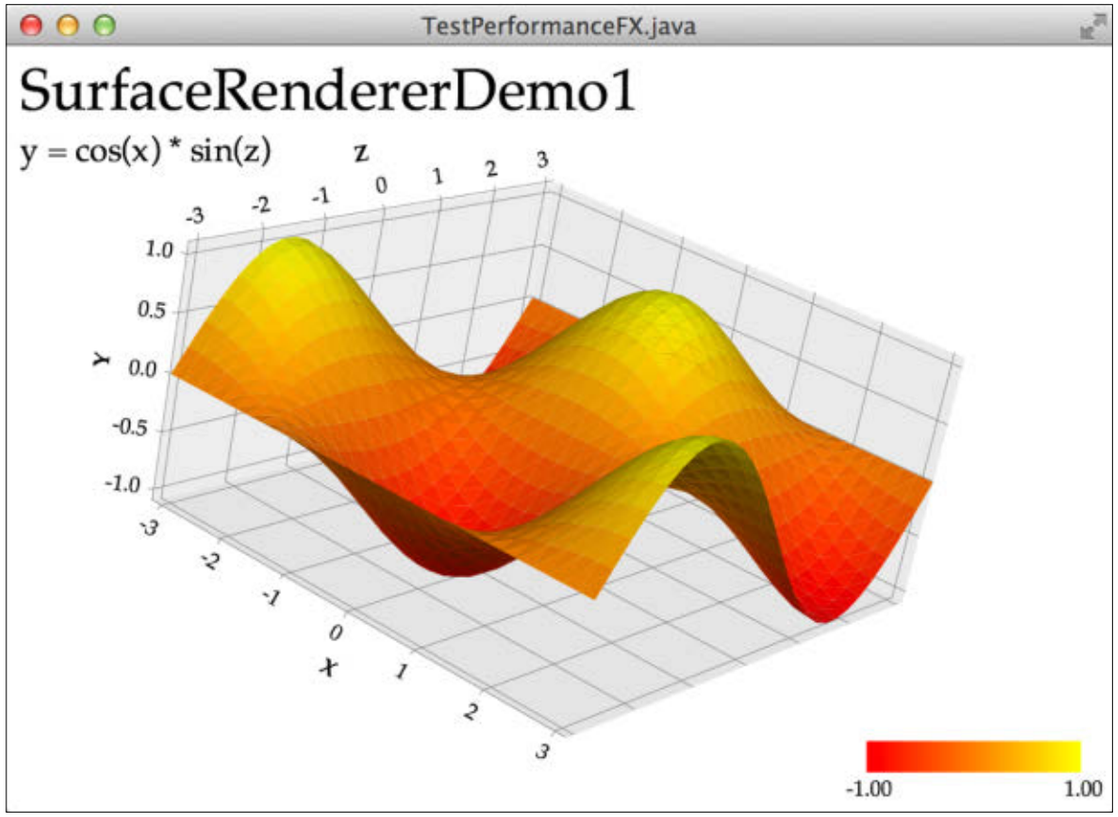


Figure 3

In fact, the one benchmark that I found online suggests that the two graphics engines have roughly equivalent performance, which is not surprising.

We wrote two benchmark programs, one for JavaFX (TestPerformanceFX.java) and an equivalent program for Swing (TestPerformanceSwing.java). You can download the source code for these programs from GitHub. The benchmark programs render 3-D charts using Orson Charts, a commercial library that is similar in style to JFreeChart (it has the same author: me) but is for 3-D charts rather than for 2-D charts.

To ensure that anyone can run the benchmarks locally, the benchmarks require only the free evaluation .jar file for Orson Charts, which you can download.

The benchmark programs create a frame, display a 3-D chart, and then rotate the chart for 1,000 frames. The first 500 frames are considered the warm-up phase to allow the just-in-time (JIT) compiler to work some magic, and the second 500 frames are timed. **Figure 3** shows a screenshot from the JavaFX version (itself a nice example of what can be done

RUN	JAVAFX WITH FXGRAPHICS2D	SWING WITH JAVA 2D
1	18,491 MILLISECONDS	17,623 MILLISECONDS
2	18,636 MILLISECONDS	17,342 MILLISECONDS
3	18,736 MILLISECONDS	17,429 MILLISECONDS
AVERAGE	18,621 MILLISECONDS	17,465 MILLISECONDS

Table 1

with FXGraphics2D).

I ran this benchmark on a recent (mid-2014) Apple MacBook Pro with a high-density display on JRE 1.8.0_11.

Table 1 shows the results. Note that you must run the JavaFX benchmark with the following Java Virtual Machine (JVM) option:

```
-Djavafx.animation.fullspeed=
true
```

Without that option, you'll get some impressive but misleading results, because the JavaFX rendering engine will populate the rendering queue more quickly than the capped frame rate, and some frames will be dropped, thanks to the clearRect() call described previously. For nonbenchmark use, this might be what you want.

As you can see in **Table 1**, the FXGraphics2D rendering is—for this particular configuration, at least—slower than the same code in Java 2D, but not by a large margin. Unless your application has very strict performance requirements, this margin is not likely to be an issue. You should, of course, run your own performance tests with a workload that is representative of your application and target platforms.

Conclusion

Migrating to JavaFX can be a lot easier if you don't need to throw away a lot of existing code. This article demonstrates that FXGraphics2D is a robust solution that allows you to keep your tried-and-tested Java 2D code and incorporate it into JavaFX applications.

The FXGraphics2D code is free to download, and the license is business friendly. There are no obstacles to using this solution for your own JavaFX applications. `</article>`

LEARN MORE

- FXGraphics2D project
- JFreeChart

//fix this /

In the September/October 2014 issue, Cyril Lapinte gave us code that was not making effective use of streams.

The correct answer is #3: Use the noneMatch() method instead of counting the number of filtered results from the inner stream. This will terminate the inner stream as soon as a match is found, meaning that the number is not prime. This will reduce the amount of work done in the inner stream pipeline considerably.

This issue's challenge comes from Abhishek Gupta, a senior identity and access engineer, who gives us a JAX-RS problem.

1 THE PROBLEM

The JAX-RS API (JSR 311) allows us to create resource implementations by inheriting from parent classes decorated with JAX-RS metadata annotations. This lets us enjoy all the benefits of inheritance and isolate the usage of JAX-RS annotations within a single parent class.

2 THE CODE

The ParentResource class is a simple POJO decorated with JAX-RS annotations. Because it is abstract, it has to be extended by a subclass in order to be utilized. ChildResource is the subclass that inherits from the ParentResource and provides an implementation for its one and only abstract method: greet.

Assuming that the REST service is accessible on http://localhost:8080/TestREST/demo, what do you think will happen when an HTTP POST request is sent to the service with a text (e.g. Duke) as the HTTP message body payload?

```
public abstract class ParentResource {
    @POST
    @Consumes(MediaType.TEXT_PLAIN)
    @Produces(MediaType.TEXT_PLAIN)
     public abstract String greet(String name);
}

@Path("/demo")
public class ChildResource extends ParentResource {
@Override
@Consumes({MediaType.TEXT_PLAIN, MediaType.TEXT_XML})
public String greet(String name) {
        return "Hello " + name + " !";
    }
}
```

Hint: The ChildResource class overrides @Consumes on the greet method.

3 WHAT'S THE FIX?

1) The invocation results in an HTTP 500 error and is fixed by putting the @Path("/demo") annotation on the ParentResource instead of the ChildResource class.
2) The client gets back "Hello Duke!" as the message along with an HTTP 200 response code.
3) The invocation results in an HTTP 404 error and is fixed by including the @POST annotation on the overridden greet method of the ChildResource class.
4) The application fails to deploy.

GOT THE ANSWER?

Look for the answer in the next issue. Or submit your own code challenge!

ART BY I-HUA CHEN