Return to Module Overview Page (https://onlinelearning.berkeley.edu/courses/665040/pages/module-one)

1.2: Enter Oak

None of these scenarios are unfamiliar to most computer professionals. The first three problems portability, maintainability, and safety have been the bugbears of software engineering since shortly after the invention of the computer. The security problem is somewhat newer, appearing with the personal computer revolution in the late seventies and really blossoming in the age of the Internet.

Over the years, there have been many attempts to deal with these problems, and huge advances have been made. Oracle/Sun Microsystems' Java programming language represents the latest and greatest advance in this long evolution. It adds other innovations in the bargain, but in the end it is Java's combination of solutions to the above-mentioned Big Four problems that makes it so exciting.

Where did Java come from?

Legend has it that James Gosling and his research group, working at Oracle/Sun's Sun Laboratories division, were investigating "small, reliable, safe, long-lived, portable, real-time, distributed systems." In fewer words, they were thinking about a kind of Internet, a network of computer-enhanced consumer appliances in your house that could coordinate everything from climate control to interactive entertainment. Although this home Internet did not materialize, one offshoot of this research would be a new computer language, originally named Oak, in which the devices were to be programmed.

The language being sought needed the four characteristics we have already noted as often lacking:

Portability. Small programs written in this language would be sent around the network from device to device. Therefore, Oak programs needed to be literally portable: it had to be possible to run the same executable code on many different pieces of hardware. It would also help if the computer language could easily handle human languages other than English. (C++'s dependence on the 8-bit ASCII character set causes problems even in Europe, let alone in Asia.)

Maintainability. The language had to be small and the code easy to read, so that software could easily be shared among device manufacturers. Furthermore, it had to be easy to determine the relationships between different chunks of code: how else would you know which chunks needed to travel together over the network? Still, it needed to be both powerful and familiar. Ideally, it would be very similar to an established language such as C or C++.

Safety. Erroneous code running on the Internet couldn't be allowed to interrupt the normal functioning of any devices. If the furnace crashed, or the pay-per-view system went down,

serious economic or other consequences could ensue. Devices would need protection against stray pointer damage.

Security. Internet viruses could spell disaster. Perhaps more importantly, they could steal cable TV! As a result, there had to be strong security controls on what a particular section of Oak code would be allowed to do.

Oak answered each of these requirements directly. First of all, it was decided that Oak code would be compiled not into hardware-specific instructions for any real computer, but into instructions for a virtual machine. The Oak virtual machine is a simulated computer, not a real one. If you write programs to simulate the Oak virtual machine on every one of the world's real computers, then compiled Oak code could run on every one of them by running inside the simulation! The simulation hides the differences between the real computers it might be running on, making all machines look the same from the perspective of the Oak program. This makes Oak code completely portable.

Oak source code, as well as text processed by the Oak input/output libraries, is represented in 16-bit Unicode. Unicode is a character set with enough bits to represent almost all of the world's human languages.

The Oak language is designed for ease of code maintenance. First of all, a given variable can hold only one kind of information, so there is less confusion (this is called strong typing, and it will be one of the central themes of this course.) Secondly, functions don't do arbitrary, inappropriate tasks. This is achieved through encapsulation, another important theme of our course. With encapsulation, functions work only within the data defined in an object. This leads to more predictability and less side effects. Oak is very similar to C++, which makes it easy for many practicing C and C++ programmers to learn. (James Gosling has been quoted as saying that "Java is C++ without guns or knives," referring to its resemblance to C++ but with all the parts that can hurt you removed.)

Oak is safe primarily because the dangerous notion of pointers does not exist. There is no use of memory addresses, so it is impossible to refer to a random address! Oak also removes a number of C++'s other dangerous elements, most importantly, many ambiguities in the language definition. In Oak, nothing is "implementation dependent" or "undefined"; in the C and C++ language standards, by contrast, these phrases appear with great frequency.

And Oak is extremely secure, because the Oak virtual machine can be controlled in a way no physical computer can. Want to forbid all Oak programs from writing to your hard disk? Just replace the disk access code in the Oak virtual machine with code to post an error message. Want to prohibit Oak programs from using your computer's network card? Insert similar vetoing code into the Oak virtual machine. Want to allow access for some Oak programs and not for others? Add a dialog box and let the user configure access permissions on a case-by-case basis.

Furthermore, the Oak virtual machine is very picky about the code it runs. Before any code is executed, the virtual machine performs a number of checks on it to make sure that, for example, all local variables are initialized before use, and the machine stack can never be overrun. Then at run time, the virtual machine continually checks for such things as out-of-bounds array access.

While the envisioned market for the home Internet technology did not materialize, Oracle/Sun soon deployed the Oak language in another context: in a World Wide Web browser called HotJava.