**The World of Objects**



**Your Dog on .Java. Any Questions?**

**BY ROGER SESSIONS**

t's hard not to like dogs. You come home after a hard day. You turn on your computer. You type



I

"dog". You always get the cutest little

"woof woof".

The only problem I ever have with

my dog is when I'm forced to use some­ body else's computer. Then when I type "dog", instead of seeing my expected greeting I get:

SYS1041: The name specified is not recognized as an internal or external command,oper­ able program or batch file.

I have thought of placing my *dog.exe* on my personal Web page. I would know then, that, wherever I am, on whatever computer I am using, and whenever I am feeling stressed out I can start up a Web server, download *dog.exe,* and get my reassuring little "woof woof".

This setup would work in those

lucky situations that contain the fol­

lowing conditions:

• I am using an IBM compatible machine.

• I am using an OS/2 system instead of a you-know-what system.

• All of the DLLs my dog depends on have been installed.

Wouldn't it be nice if I could down­ load my dog to *any* machine, while running *any* operating system, and know that I could count on my dog doing its little doggy thing?

For that matter, as a dog developer,

who has invested thousands of person hours in developing my dog applica­ tion, wouldn't it be nice to know I could sell it to dog lovers everywhere?

**46 OS/2 M A G A Z I N E** J U N E 1 9 9 6

And that I wouldn't have to have one version for OS/2, another version for MVS, another version for Macintosh, and soon?

**Smell the coffee**

The ability to create a single executable

and run it on any machine is the most

important feature of a hot new pro­ gramming language called Java. Like any self-respecting program developed in the last 10 years, Java is object-ori­ ented, which makes it fair game for this column.'Unlike other recently devel­ oped programming languages,Java has received unprecedented interest with­ in the programming community, even before it was released as an official product. If you start getting interested in Java, there are a few web sites you might want to check out. These sites are described in Table 1.

Most of Java's publicity has focused on its ability to run programs within Internet Wb browsers; it is within this context that most Java code is current­ ly bt'!ing written. However, over the long haul, it's the system independence of the program executable that may well prove its most historic feature.

On December 6th of last year, IBM announced that it had licensed Java, and that it"...intends to port Java tech­

nology to its OS/2 and AIX operating



systems, as well as Microsoft Windows

3.1, and it will make those ports avail­

able over the World Wide Web..."

IBM has already fulfilled at least part of this availability promise. At press time for this column, you can set your

Web pointer to [*http://ncc.hursley.ibm.*](http://ncc.hursley.ibm/) *com/javainfo/,* register as a Java devel­ oper, and download the entire beta

Java system.

If you do, make sure you installJava

in an HPFS partition.Java files use very long path names, and have no option for standard DOS-style 8.3 names.

Let's take a look at some actual]ava code and get a feeling for the language.

We will start with an animal inter­ face that defines the **eat** and **setFood** methods. From it, we derive a dog inter­

face that adds on the l:>ark method. The animal interface is defined in the file *animal.java.* The dog interface is defined in *dog.java* (Listing 1).

An interface describes what meth­ ods an object supports, but not how those behaviors are implemented. In Listing **1,** the first line of both *ani­ mal.java* and *dog.java* uses the keyword **interface** to indicate that we are defin­ ing interfaces, not implementations. These irtterface files are used by clients to understand what behaviors are sup­ ported by their objects.

An interface can be derived from

**The World of Objects**

**Listing 1: Interface files.**

another interface, as is our dog from our animal in Listing 1. This fact is indi­ cated by use of the keyword extends. We say dog extends animal, meaning that dog will support all the animal behaviors in addition to its own.

In Java, a class is one of many possi­

this is unfortunate. The con­ cept of interface descrip­ tions, independent of im­ plementation, is a useful concept. Describing all pub­ lic classes with well-docu­ mented interfaces and dis­ tributing these as source files, while distributing implementations only as binary files, would be very natural.

Our client code is shown in Listing 3. In C++, client code is not typically a class.



In Java, everything is a class (except for interfaces, which aren't really things at all, but only definitions of things). You run a class.

As you can see in Listing 3, the class test defines one public static method

called main. It's defined as public so that it can be invoked publicly. It's defined as static, because it can be invoked without an actual object of the class being instantiated. The name of the method is main, which is the method that is run by default when running a class.

The method test::main declares four objects. Pooh is an animal, while

Snoopy, Lassie, and Toto are all dogs. Keep in mind that animals are just interfaces. We won't discover what actual implementations will be used until object instantiation time

The instantiations use the new directive, as shown in the client code. Through use of this directive, Pooh becomes an animal\_implementation, a class which supports the animal inter­ face. Snoopy, Lassie, and Toto become

ble implementations of an interface. Listing 2 gives the animal\_implemen­ tation class, an implementation of the animal interface, and dog\_implemen­ tation, littleDog\_implementation, and bigDog\_implementation class­ es-three implementations of the dog interface. We use the keyword class to indicate we are showing an implemen­ tation, not an interface.

Implementing an interface means to provide implementations of each of the methods defined in a particular inter­ face. Our dog\_implementation imple­ ments our dog interface, which, remember, includes the animal inter­ face. We use the keyword, implements, to indicate that a particular class is implementing a particular interface.

Extending a class means deriving a new class from an existing class. The derived class can either add new meth­ ods or override existing ones. As shown in Listing 2, our dog\_implementation extends the animal\_implementation by adding a bark implementation to the setFood and eat implementations

provided by animal\_implementation.

Our littleDog\_implementation and bigDog\_implementation extends the dog\_implementation by overriding

the bark method.

The use of interfaces is optional in Java. Most writers either don't describe it at all or describe it as a poor man's version of multiple inheritance. I think

**Listing 2: Implementation program.**



**OS/2 M A G A *Z* I N E** J U N E l 9 9 6 **47**



dog\_implementation, big­ Dog\_implementation, and lit­ tleDog\_implementations, respectively-all classes that support the dog interface.



java has simplified the cod­ ing process with two decisions. The first is lack of support for pointers. The second is auto­ matic memory management.

You can see the impact of both of these decisions in the implementation of the setFood method shown in Listing 2. This implementation requires only a single line of code:

food = myfood;

In C++, this code looks like:

if (!food) free(food);

food= <string)

malloc(strlen

: i (myfood)+1);

strcpy(food,myfood);

Two out of the three lines of the C++ version are dealing · with memory management. Not only is most of the code doing memory management, but it's by far the most compli­ cated code. The ability of java to automatically manage memory

is a considerable simplification.

Not only is the java code simpler, it's less fragile. The java version will work fine even if the food variable had never been initialized. The C++ code will fail on any system where

food had not been initialized to NULL. java is heavily influenced byC++. The main syatax of declaring vari­ ables, using operations, and invoking methods is the same in both lan­ guages. The major differences, many of which we have already discussed,

include the following:

• addition of the concept of interface

• addition of automatic memory management

• lack of support for pointers

• lack of support for multiple inheri­ tance in class definitions, though it is supported in interface definitions

• lack of support for structures, which were considered redundant with classes.

**48 05/2 M A G A *Z* I N E** J U N E 1 *9 9* 6

milar to C++,Java has full support for polymorphism. (If you are unfamil­ iar with polymorphism, see "littleDogs, Polymorphism, and Frameworks," Feb­ ruary 1996, p. 46.) You can see the polymorphic resolution of the bark method in the client code output shown in Listing 3, where the three dif­ ferent dogs bark differently based on their implementation of bark.

java is heavily influenced by the Internet. The mindset of]ava is to de­ velop programs that, intheir executable form, can be downloaded and run on any machine.Inother words, java seeks

to create a universal executable.

java accomplishes this universal

executable by running all Java exe­

cutables through an interpreter. The

executables are, in fact, com­ piled versions of the source code, but they are compiled for the interpreter.

This concept is different than our traditional notion of interpreters. Most interpreters run source code directly. This method achieves universality but at a considerable perfor­ mance penalty. The java inter­ preter runs binary code that has been heavily optimized by the java compiler.

Unlike most binary code, the binary code produced by the java compiler is universal­ ly standardized. It's exactly the same whether compiled on OS/2 or on Windows 95, be­ cause it's produced according to the standards set by the Java Interpreter, not the standards set by an operating system. It's the java Interpreter that is responsible for getting the

binary to run on a particular operating system.

Thus, we get both universal­

ity and performance. We get universality because the java

interpreter is responsible for running a program, and any machine with a java inter­ preter can run any Java pro­ gram. We get performance, because the interpreter is run­ ning pre-compiled code rather than source code.

We run the java compiler

by typing "javac" and the name of the source file, which

must include the extension *.java.* For example, to compile our test class, we type "javac test.java."

The compiler will automatically compile' any other files on which this file is dependent. So for our test pro­ gram, we will automatically compile *animal.java, dog.java, animal\_imple­ mentation.java,* and so on.

The compiler generates files with the extension *.class.* In our case, the compile of *test.java* will result in *test.class, dog.class,* and so forth.

We can run any of these classes by invoking the java interpreter and also the name of the class without any extensions. The only class that is meaningful to run is our test class, because it is the only class with a

**The World of Objects**

main. We run the test class by typing

"java test."

Java is· designed to be virus un­ friendly.Java programs are intended to be downloaded over the Internet. We

are all aware of the danger of down­ loading viruses. The Java developers wanted to make Java an unlikely carri­ er for viruses, so two mechanisms are used to protect Java programs.

The first protection mechanism is lack of support for pointers. We have

already discussed this in the context

of program simplification. Without pointers, writing a program that will wander outside of its address space, and, for example, starts usurping func­ tions of the operating system is very difficult.

The second protection mechanism

is through the Java interpreter. Because

. all Java programs run under control of the interpreter, Java has an opportuni­ ty to pre-screen programs before they

are allowed to run. Considerable intel­

ligence is being built into the inter­

preter to ensure that programs don't do things outside the bounds of stan-

dard operations, such as update system files.

In summary, lets recap the major Java features that make it attractive to online developers.

II Java is a full object-oriented pro­ gramming language, supporting the three must-haves of any object-ori­ ented language: encapsulation, inheritance, and polymorphism.

II Java supports separation of interface and implementation, allowing dis­ tribution of interface source with­ out distribution of implementation source.

II Java is much easier to program than C++ due to its automatic memory management and lack of support for pointers.

II Java executables are machine-inde­ pendent and virus-resistant, allow­ ing them to be easily and safely dis­ tributed over networks.

So this is your dog on}ava: simple, safe, and a dog that will follow you anywhere.

***Acknowledgements***

*The sample code* in *this article* is *derived from some code I developed with Simon Nash at IBM. I appreciate his willingness to share his expertise and to"zetme use this code.* !llifil

*Roger Sessions is presidentofObjectWatch Inc., a company specializing in training and consulting in the use of COREA tech­ nologies on IBM platforms.He has spoken at over 30 conferences and has written extensively.His books include* Object Per­ sistence: Beyond Object-Oriented Databases, Class Construction in C and C++; Object-Oriented Fundamentals, *and* Reusable Data Structures for C. *Roger also publishes an Internet newslet­ ter called* ObjectWatch on SOM *and can be contacted via e-mail at* [*roger@fc.net.*](mailto:roger@fc.net)

***Schedule Programs*** *&*

***Reminders Automatically...***

***Chron v4.0***



*Scheduler for OS/2*

**Only $99**

Site Licensing Available

Uses:

• Schedule backups and database maintenance

• Remind yourself of recurring meetings

• Schedule long-running or resource

**Can Your Test Tool Tackle The Entire Testing Process?**

Business worf<flqvy<.;j3egr§esion ... Load ... Concurrency ... MdlticiJser...Str§SS ... Inte­ gration ..: are just a few ofUJesj(yations you need to deal With when test' terprise Systems> And, if you don't L . ..r system the wayit'pgqing to be usedlfl ()tual oper­ ation, the test results may J:m!.JJJ[¢1iable.

Softbridge's AutomateqC · '"'acility has a unique architecture deslg,,ed to perform rigorous tests that PC!r§I!Jel actual operating environments- environments where end-users spread across an organization enter, access, and act on informC1ti()n collaboratively.

consumptive tasks for after hours ATF tests applications under Windows, Windows NT, Windows 95 and all flavors of OS/2.



Hilbert Computing



1022 N. Cooper

Olathe, KS 66061

Voice: (913) 780-5051

BBS/Fax: (913) 829-2450

CIS: 73457,365

Reader Service No. 21

For information On ATF contact: Softb(idg, Inc.

125 Carl1bridgePark Dr. Cambridge, MA 02140

Phone: (617) 576-2257

Fax: (617) 864-7747

E-mail: [market@sbridge.com](mailto:market@sbridge.com)

Reader Service No. 22



**05/2 M A G A Z I N E** J U N E 1 9 9 6 **49**

.................. . ---