

Made with Jade

If you ever found yourself wondering what happened to 4GL development environments, you'll find the modern equivalent in JADE from New Zealand-based Jade Software Corporation. **Ian Yates** goes in search of new gems.

JADE is like FileMaker on steroids, implementing a self-contained database and coding package that you can download for no charge.

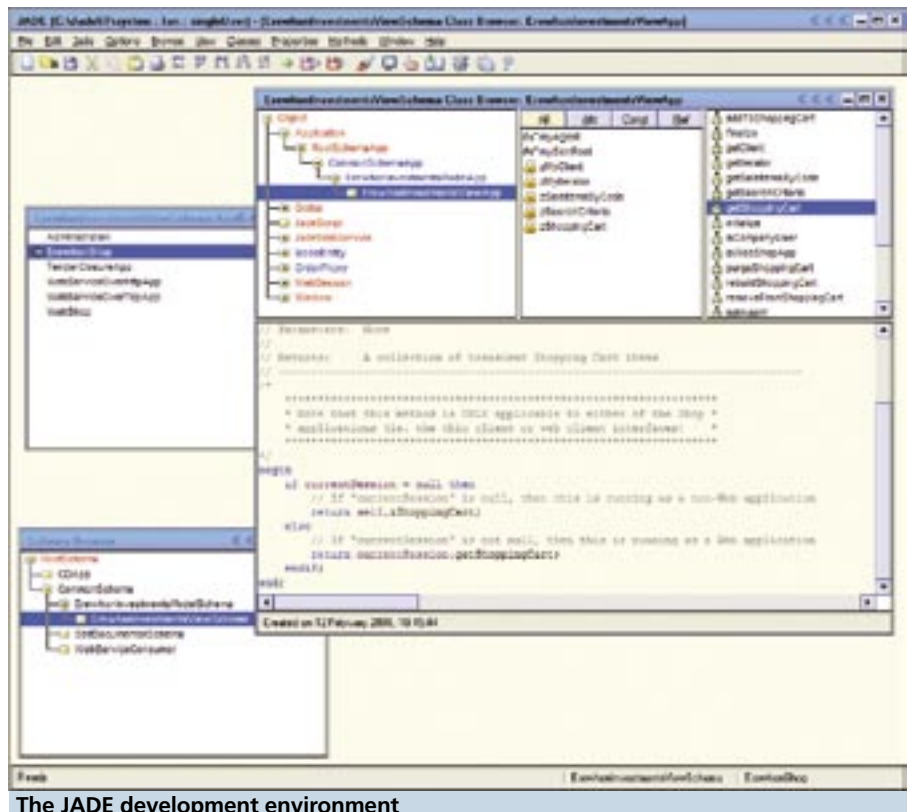
You have to pay money to be able to use any applications you develop, but you can play with the product and try out its features without getting budget approval.

Elizabeth Post at the Lincoln University in Canterbury, NZ describes JADE as, "Totally object-oriented, making it ideal to teach object-oriented concepts, such as encapsulation, inheritance and polymorphism and it's simple to use compared to other object-oriented languages. There is no confusion with alternative ways of programming or non-standard interpretations of object-oriented concepts as in some other so-called object-oriented languages. Its integrated language, database and development environment also make it an ideal introduction to an object-oriented database."

Concentrate on the programming

"For anyone wanting to get to grips with the object-oriented paradigm, JADE is perhaps the easiest way of learning object-oriented programming. With its visual class browser it is easy to understand such concepts as classes and objects, properties and methods, relationships between objects, encapsulation, inheritance and polymorphism and also to gain insight into how to write reusable code, as is done in JADE itself. Also, JADE handles such issues as persistence of data, distributed computing and multiple users, and concurrency in such a way that the programmer need not be concerned with much of the detail, but can concentrate on programming."

For anyone wanting to get to grips with the object-oriented paradigm, JADE is perhaps the easiest way of learning object-oriented programming



The JADE development environment

While that description is certainly true, JADE is not just designed to teach the kiddies all about object-oriented programming. The product has been used extensively by some major corporate and government clients to build complete software solutions. Jade Software Corporation also accepts contracts to write bespoke systems itself for clients that don't want to cut their own code and offers several 'off-the-shelf' packaged

solutions that can be readily customised to suit client requirements in the areas of health care, education administration, logistics and human resources.

Getting started with JADE is best achieved by following the vendor's online tutorials, which introduce the concept of a 'schema' to describe the database, which should be familiar to many developers. What might not be so familiar is the realisation that everything lives inside the schema, not just the data definitions but the forms and the code that allow users to interact with the data. From within the JADE environment you can inspect the raw contents of the database as well as the definitions of the database. Once you get comfortable with this all-in-one packaging approach and stop looking for the separate pieces, it all starts to make sense.

Re-usable code philosophy

Any new schema gets automatically populated with JADE's RootSchema which includes most of the code you would ever need to build

your application. Because the schemas are inheritance-based, anything added to a lower level can make use of any code parked in a higher level. This means that each time you conjure up a snippet of code you can elevate it to a higher level in the schema and make use of the code in some other part of the application, bringing the re-usable object-oriented philosophy to the forefront.

Objects 101

Objects represent the physical and conceptual things that exist in the world around you. You might consider yourself as an object with properties such as eye colour, weight, age and height. You are capable of certain types of behaviour such as walking, talking, sleeping and running. You also have your own unique identity that sets you apart from other people, or objects of the same type. Finally you interact with other objects to accomplish certain tasks. For example, in the context of an airline reservation system you might make a booking to travel. In doing so the object representing you would interact with a flight object, an aircraft object, a booking object and a seat object.

JADE combines all of these aspects into a single solution. It closely follows OO concepts and implements them without compromise. This is the main reason why so many tertiary organisations have adopted JADE as the prime tool to teach OO concepts and reinforce them by direct practical application.

For the purposes of modelling objects are thought of as having state, behaviour and identity.

The state you're in

The state of an object encompasses all its properties and their current values. State is something that you can measure and assign a value to such as weight and number of seats. State also covers relationships with other objects such as your flight, aircraft or pilot. These values, called references, allow you to navigate from one object to a related object. Without these references, you could not communicate with other objects.

JADE suggests following a common convention when naming properties such as prefixing the name of a reference with "my", for example myFlight and myAircraft. Sometimes a reference is to a collection of objects, such as

The main task for a JADE programmer is to write code to enable objects to carry out behaviour that is associated with them

all my flights or all my bookings. In this case, the prefix "all" is used: for example, allFlights and allBookings. Relationships may be one-to-one, one-to-many or many-to-many.

Appropriate behaviour

Behaviour covers what an object can do, for example, in the case of a plane, refuel, service, depart and land. The main task for a JADE programmer is to write code to enable objects to carry out behaviour that is associated with them. In JADE, behaviour is implemented using methods. The names given to methods are generally verbs.

To make an object do something you must identify both the object and the desired action. For example, to make John sit down you would not walk over, move his legs and arms and push him on the chair. Instead you would ask the "John" object to invoke the "sit down" method. Similarly to book a flight on an aircraft you ask the person behind the desk or at the end of the phone to do the booking on your behalf. You just supply the parameters such as destination, date, preferred price range and so on.

This is where object programming differs from procedural programming where a large program manipulates the data. In an object-oriented system objects manipulate their own data using their own methods.

Every object has a unique Identity that is separate to the values of any properties associated with the object. This allows you to differentiate one object from all other objects even if the value of all their properties are the same. In JADE, this unique identifier is called the object id. The object id is assigned when an object is created and does not change for the

lifetime of the object. The object's state may change but its object id never changes.

A unique object id allows for quick access to any object. Once you have an object id, also called a reference to an object, you can send a message to the object to invoke one of its methods. This is what happens in all object-oriented applications. Objects send messages to each other that ask them to carry out tasks.

A touch of class

A class is a template that is used when an object is created. A JADE developer creates classes in the JADE development environment and adds properties and methods to describe and model the desired behaviour (see Figure 1).

An object is an instance of a class. At run time the application finds the appropriate class and uses it as a blueprint when creating objects. The diagram above shows the customer class and two customer objects based on the class definition. In fact, classes are really a special category of objects. In JADE they are instances of the Class class and as a developer you can access class definitions from within your code as you would any other object.

Collections

A collection is an object that groups several other objects into a single unit. A collection may sometimes be called a 'container' and stores references to the objects that are part of the collection. In JADE there are three types of collection as shown in Figure 2.

A collection may be used to implement an index, for example all customers in name order. Collections can also be used to implement one-to-many and many-to-many relationships

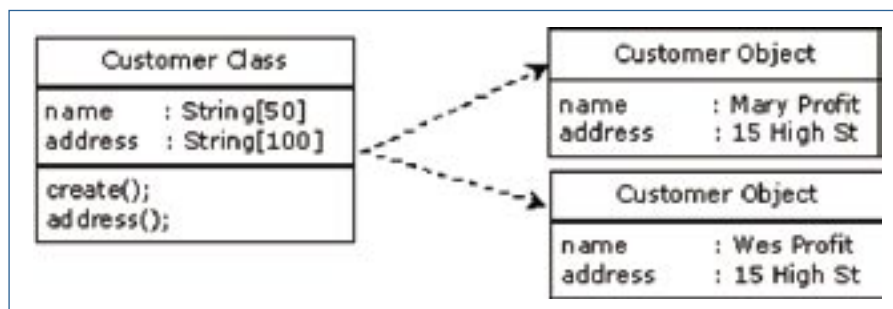


Figure 1 Objects created as instances of a class

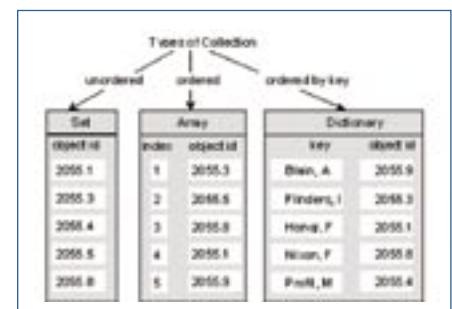


Figure 2 JADE Collections

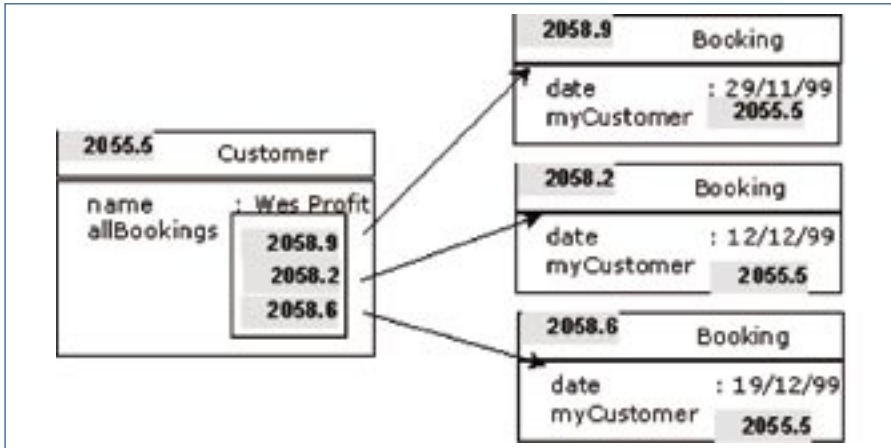


Figure 3 One-to-many relationship in JADE

between objects. In this latter case each object can 'own' a collection of related objects, for example all bookings for a given customer or all seats on a given plane. Since the collection just stores references to the member objects they are generally very small (see Figure 3).

In the diagram above a customer object has a collection of all its bookings, called allBookings. Each booking object has a property called myCustomer that stores a reference to the customer associated with the booking.

In JADE these properties, allBookings and myCustomers, can be defined as inverse references, or end-points in a bi-directional relationship. The effect of this is that should you update one property in your code, JADE will automatically update the other end of the inverse relationship. For example, when a booking is created, setting the value of the myCustomer property for a Booking object will automatically update the allBookings collection for the appropriate Customer object.

Encapsulation

Encapsulation, in an object-oriented context, refers to the practice of packaging data (properties) and code (methods) together as objects and in such a way that no other object need be aware of the internal structure. When you define properties and methods you specify whether they are part of the public interface or whether they are private and therefore part of the internal structure of the class. Objects can only be accessed through their public interface. This distinction between what is public and what is private is often referred to as Information Hiding.

In JADE, properties may be public, protected, or read-only. Methods may be public or protected. A public property may be read or updated by other objects while a read-only property may be read but not updated. A protected property is not visible to other objects at all and is not part of the public interface. Public methods form part of the public interface while protected methods are not accessible by other objects. An extreme approach is to make all properties protected and provide methods to get and set property values. In practice,

the read-only option is often used to provide read-access to a property while preventing unauthorised updates to the property.

Inheritance

Inheritance, in the object-oriented context, is the means of defining one class in terms of another class. Classes are arranged in a hierarchy, with classes lower in the hierarchy inheriting properties and methods defined on classes higher in the hierarchy. In the diagram below, Concert, Flight and Game are all types of Event (see Figure 4)

JADE uses a single inheritance model. This

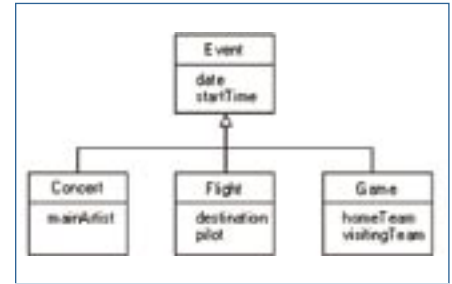


Figure 4 Class inheritance

means that each class can have only one parent or super-class. Sometimes classes higher in the hierarchy will be defined as abstract classes. An abstract class is one that can never be instantiated, but which implements properties and methods common to all sub-classes. In the example above, Event is an abstract class.

Classes inherit all super-class properties and methods. A sub-class may extend a super-class method and add behaviour specific to that sub-class. A sub-class may re-implement a super-class method to completely replace any super-class functionality. The ability to extend or re-implement super-class methods is the means by which polymorphism is supported in JADE. An abstract method is a method defined on an abstract class with no code. The implementation of the abstract method is deferred to the subclasses – in other words all subclasses must re-implement an abstract method.

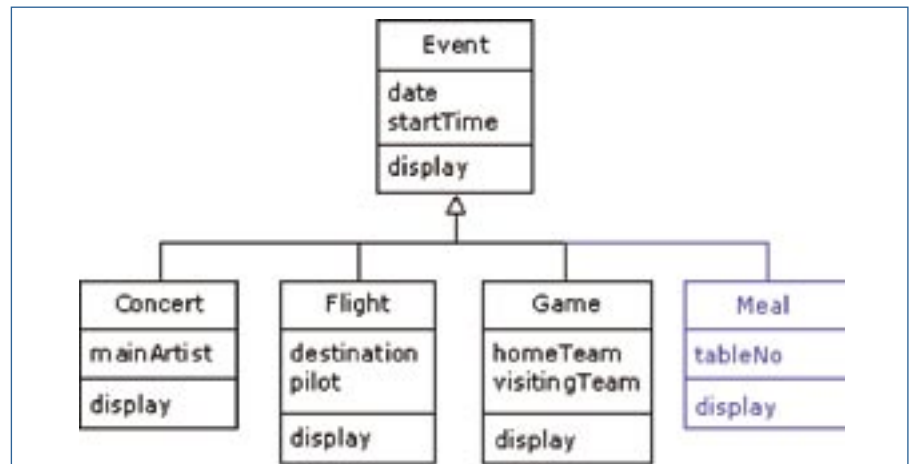


Figure 5 Extending a system by adding a class

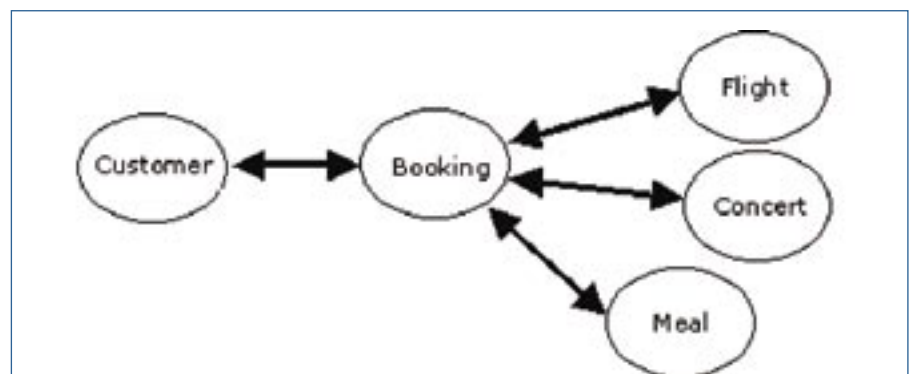


Figure 6 Messaging

Polymorphism

The word polymorphism is derived from the Greek word and means "assuming several forms". Polymorphism, in the object-oriented context, refers to the ability of objects to respond in different ways to the same request, depending on their type (or class). A message, or request to execute a method, can be sent to an object and the actual method executed will depend on the class of the object to which the message is sent.

In the sample booking system, the display method is implemented on the Event class and on each of its subclasses. This method displays details about the receiver event. You will see soon that the JADE inspector invokes this method when you inspect database objects. The actual method executed depends on the type of object receiving the message. Better still, if you add a new type of event, say Meal, and implement the display method, the new method will automatically be invoked for any Meal object being inspected (see Figure 5).

Another example of polymorphism in the sample Booking System is in the area of forms. Simple maintenance dialogs are arranged in a hierarchy with most of the functionality contained in higher-level classes. New maintenance forms can be added quite quickly as only a few methods need to be created and since the public interface is defined at the super-class level the new methods are invoked automatically.

JADE Object Manager

The JADE Object Manager (JOM), first used in 1991, was the first part of JADE to be built and has been tested underneath many complex applications. The JADE Object Manager administers the JADE distributed computing environment. It provides transient and persistent data storage, transaction management, cache management, concurrency control and dynamic binding. The JADE Object Manager manages the location of code execution. By default, all methods are executed on the client node and therefore the processing load is shared across all

Polymorphism, in the object-oriented context, refers to the ability of objects to respond in different ways to the same request, depending on their type or class

machines attached to the application.

This approach helps avoid bottlenecks associated with a fat-server environment. It is possible, using server execution methods, to transfer the processing from the client back to the server with one extra option on the method signature. The JADE Object Manager takes advantage of multiple processors by multithreading tasks.

The JADE Object Manager handles local caching of data on both client and server nodes. JADE has its own cache handling routine and if an overflow occurs, a temporary database is created. All settings for these options are contained in the Jade.ini initialisation file, which can be tuned to suit a particular application.

JADE Object Database

The JADE Object Manager includes an object-oriented database. The JADE Object Database has the power and resilience of the best relational databases and the added advantage of being integrated with a development environment. The JADE Object Database stores not only user data but data maintained by JADE itself, or system data. User data is not restricted to primitive values but includes relationships between objects. These relationships are

implemented using references and collections and enable objects to communicate with one another (see Figure 6).

System data includes class definitions, method code as well as data to support the JADE run-time environment. The JADE database also handles automatic synchronisation of updates to a primary database, with one or more secondary databases in different locations. The secondary database may be a hot standby server for disaster recovery purposes or could be used to provide read-only access to data.

Seamless interface

When writing JADE code, the reads, writes and calls to the database are integrated with the language and no database interface calls need to be written. These are encapsulated within the JADE Object Manager. There are no messy interfaces between definition, user interface and database (see Figure 7).

Deployment options

The JADE Object Manager also allows for multiple deployment options with its Smart Client Technology. This means the same repository can service a regular client/server application, HTML web-enabled application and JADE Smart Thin Client, at the same time. All these applications can share this same object database, so you can have LAN, WAN and Internet connected users all sharing the same object database (see Figure 8). ■■

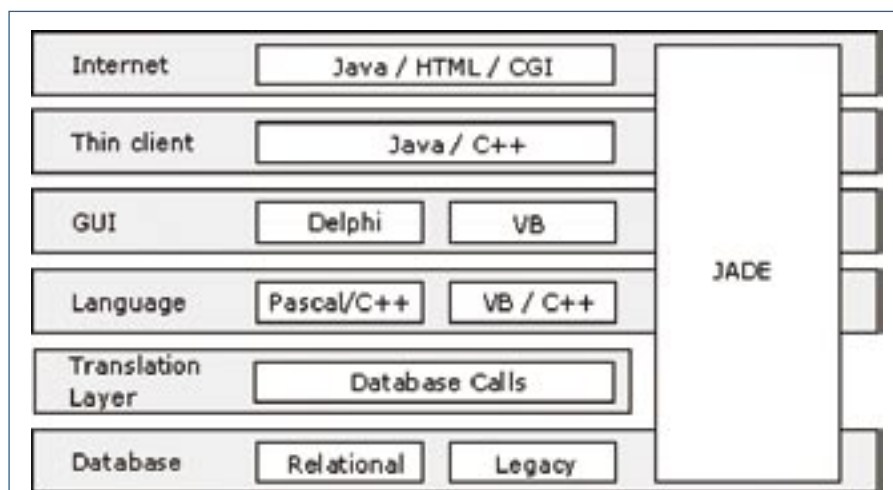


Figure 7 Seamless

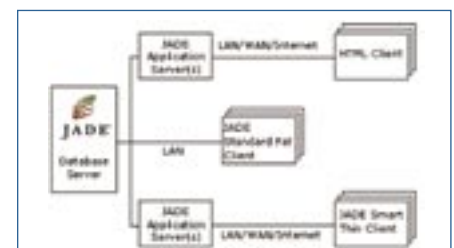


Figure 8 Local and remote access to a JADE system

URL: www.jadeworld.com