



# OPEN*workshop*<sup>TM</sup>

## An Overview

OPEN*workshop* is the Object-oriented development environment from Thoroughbred Software International. It features new concepts that dramatically change the way developers build business software.

OPEN*workshop* offers:

- Much lower development and maintenance costs,
- A much more flexible system for users,
- The choice of both Character and Graphical user interfaces.

OPEN*workshop* is based on simple concepts:

- Objects and attributes
- Wholes and parts
- Classes and members

...



Published by:  
Thoroughbred Software International, Inc.  
19 Schoolhouse Road  
PO Box 6712  
Somerset, New Jersey 08875-6712

Copyright © 1999 by Thoroughbred Software International, Inc.

All rights reserved. No part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

Document Number: WP001

The Thoroughbred logo, Swash logo, and *Solution-IV* Accounting logo, THOROUGHbred, IDOL, OPEN WORKSHOP, AND VIP VISUAL IMAGE PRESENTATION are registered trademarks of Thoroughbred Software International, Inc.

Thoroughbred *Basic*, *Thoroughbred Environment*, *OPENworkshop*, IDOL-IV, *Dictionary-IV*, *Script-IV*, *Report-IV*, *Query-IV*, *Source-IV*, TS Network DataServer, TS ODBC DataServer, TS ORACLE DataServer, VIP (*Visual Image Presentation*), VIP4, GWW (*Gateway for Windows<sub>NT</sub>*), *GWWImage*, TS ChartServer, TS ReportServer, TbredComm, WorkStation Manager, and *Solution-IV* are trademarks of Thoroughbred Software International, Inc.

Thoroughbred Software International, Incorporated licenses the U/SQL Client-Server product from Transoft Limited. Thoroughbred Software International, Incorporated incurs sole responsibility for support and maintenance of the Thoroughbred U/SQL Client-Server product. Transoft, AIM, OEO, UBB, U/BL, U/FOS, U/Gi, and U/SQL are trademarks of Transoft Limited.

MS-DOS, Xenix, Windows, Windows for Workgroups, Microsoft Windows 95, and Windows NT are trademarks of Microsoft Corp. IBM, IBM PC, OS/2, PS/2, and PC-DOS are trademarks of International Business Machines Corp. DEC, OPEN VMS, and ULTRIX are trademarks of Digital Equipment Corp. UNIX is a trademark licensed exclusively through X/Open Company, LTD. Novell is a registered trademark of Novell, Inc. Oracle is a registered trademark of Oracle Systems Corporation. InstallShield is a registered trademark of Stirling Technologies, Inc.

Other names, products and services mentioned are the trademarks or registered trademarks of their respective vendors or organizations.

# BACKGROUND

## Object Technology

Object Technology began development in the early 1980's. Founded on research designed to maximize the re-usability of software, Object Technology (OT) creates Objects that reflect real-world entities, and Methods that implement the way Objects behave.

As is often the case, this new technology has taken a long time to get to market. Two fundamental problems impeded its deployment in practical business applications: hardware costs and the lack of suitable software infrastructures.

Object Technology demands more memory and processing power than traditional software development environments. The cost differential has been large enough, until recently, for end-users of business applications to avoid OT-based solutions. Today, much better price/performance values are available from hardware suppliers.

True Object Technology is rigid and inflexible when developing Transaction types of applications. For this reason Thoroughbred's *OPENworkshop* is defined as "Object Oriented". Object Oriented provides the flexibility in developing applications for "real world" business applications, by allowing the Object Rules to be modified and enhanced when needed to satisfy application requirements. Most available OT-based systems are usually Object Oriented because of this flexibility.

Enormous development efforts have been put into operating systems and utilities over the history of commercial computing. They have combined to create a wealth of resources available for application developers to use in their solutions. Early attempts to deliver OT-based development environments ignored these resources in a drive to overturn traditional architectures. Now, mainstream environment developers have found ways to deliver Object Technology while leveraging on the available infrastructures.

As a result, practical OT-based solutions are now deliverable, and we can expect to see significant market growth as the "early adopters" consolidate their positions.

## Thoroughbred Mission

Thoroughbred's formal Mission is *to provide the most highly portable language development environments for developers creating end-user business applications.*

The word "Portable" has been used widely in the software industry, and people attribute many different meanings to it. Thoroughbred itself has two distinct developer requirements in mind when committing to portable language development environments.

Firstly, developers wish to be able to design an application for their chosen markets once, and then implement it on many different hardware and software environments, depending on their own customer's needs and preferences. Thoroughbred's mission is to deliver this portability, allowing the developer to maintain only a single application code set.

Secondly, developers have invested heavily, designing applications for the end-user business market, and need to be able to conserve that past investment. Changes in technology, costs, and the competitive environment will drive them to adapt and enhance their applications, but *Evolution, rather than Revolution*, is the preferred strategy. Thoroughbred takes responsibility for enabling developers to choose this strategy. Ten, or even five years ago, it was possible to deliver the *Evolution* capability by guaranteeing upward portability in the programming language.

More recently, demands for evolution have become more complex. Evolution requirements include the enhancement of the user interface of their application to support graphical presentation (GUI), or the data environment to encompass Client/Server architectures. Thoroughbred's solution to this level of portability is based on the *Dictionary-IV*<sup>TM</sup> system dictionary. Thoroughbred guarantees that all applications based on *Dictionary-IV* will operate **without change** in all environments supported by Thoroughbred.

### Positioning OPENworkshop

OPENworkshop is an Object-oriented development and run-time environment encompassing Thoroughbred's three-tier environment.

#### OPENworkshop Features:

- Data controls the application
- Applications can be developed incrementally
- Relational Integrity is maintained and insured
- Can modify Object Rules
- Global Definition-based
- GUI and Character Applications with same Program Code
- Client/Server or Local Host-based Applications
- Internet Transaction Processing
- Windows/95/NT, UNIX or DEC VMS is supported
- Thoroughbred, ORACLE, VMS, and a variety of ODBC databases are supported

OPENworkshop provides the tools to develop mission critical applications for a variety of business requirements and system configurations.

Most of the major relational database vendors have developed or are developing Object-oriented products. In addition to these suppliers a new set of ventures are attempting to develop new Object-Oriented products from scratch, using entirely new architectures. OPEN*workshop* is an evolution of Thoroughbred's *Dictionary-IV*, and builds on the portability of previous Thoroughbred development products.

OPEN*workshop* PROVIDES support for enterprise-wide, distributed Object databases as well as local, host based systems. Thoroughbred has always had a policy to make its development environments open for its developers. As these new products become established, Thoroughbred will enable developers to take advantage of them.

OPEN*workshop* SUPPORTS an Object-Oriented Programming environment. Built to support Windows/95/NT clients and Windows/NT and UNIX Servers running either Thoroughbred's robust database or Relational Databases including ORACLE and most ODBC compliant databases including Informix, Sybase, and SQLserver. OPEN*workshop* enables the application designer to define and use the most appropriate database for the application rather than being limited to a proprietary database or limited third party databases. Each Format definition can have its own database support.

OPEN*workshop* PROVIDES support for developers wishing to use a graphical user interface (GUI) on a client workstation. Thoroughbred's VIP provides a Graphical User Interface to the application without requiring changes to the application's code.

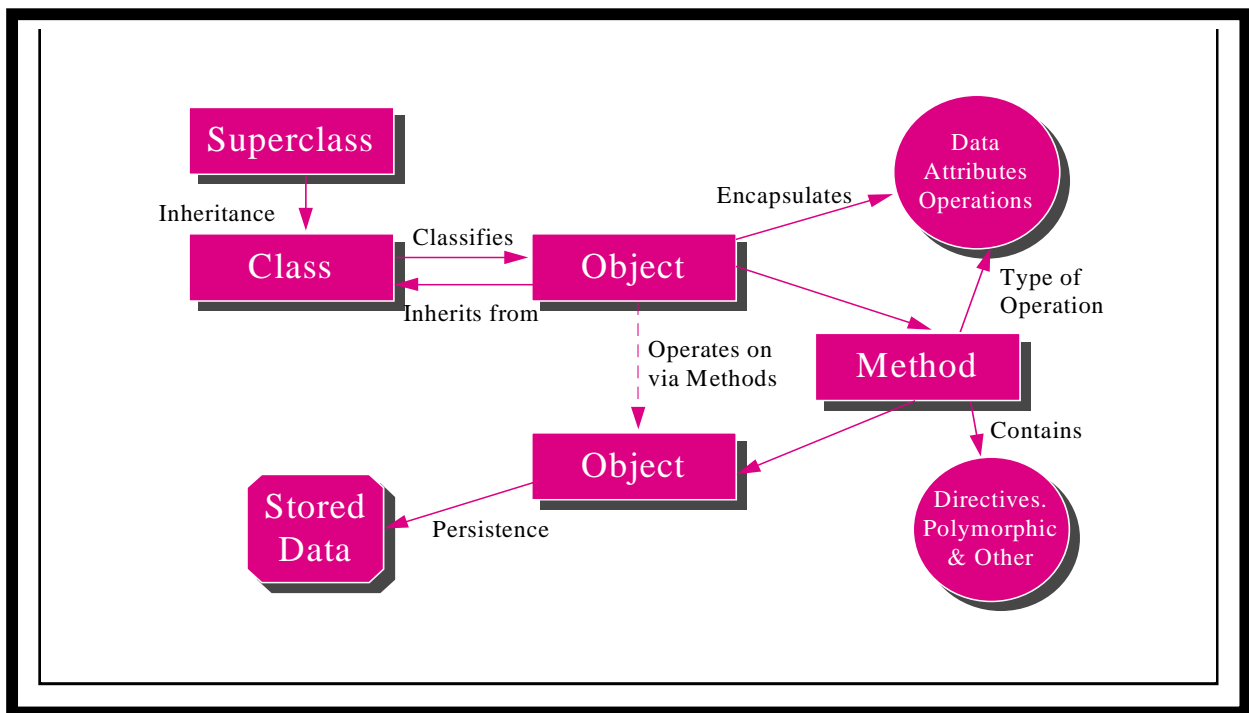
OPEN*workshop* is a highly portable Object-oriented development environment for developers creating end-user business applications. It embodies Object Technology and delivers the benefits. It is based on a time proven system engine. It uses and supports components that Thoroughbred has developed and refined for many years, while at the same time introducing many new features.

Deliberately, OPEN*workshop* is not a break with the past, but a path for *evolution* to the future.



# OBJECT TECHNOLOGY CONCEPTS

Object Technology embodies key concepts that combine to deliver its benefits. Extensive technical literature<sup>1</sup> is available on the subject, but the following provides a summary of the concepts and an introduction to how OPENworkshop delivers them.



## Object

Is any thing, real or abstract, about which we store data ad those operations that manipulate the data.

*(James Martin)*

OPENworkshop allows the developer to store information that reflects the properties of real world items in Data-names. The operations that may manipulate the data are defined in Methods and are associated with the Data-name.

## Class

A collection of objects which share common attributes and methods.

<sup>1</sup> See bibliography for a small sample

*(Ian Graham)*

In OPENworkshop Classes define a collection of Objects. Classes themselves are collected into sets with common structures or behavior. For example, all Views in OPENworkshop have similar behavior.

## **Method**

An implementation of an operation. Code that may be executed to perform a requested service. Methods associated with an Object may be structured into one or more programs.

*(Object Management Group)*

OPENworkshop Methods perform operations on Data-names. Some specific types of operation are identified for particular support by OPENworkshop, including:

- Pre- and Post-Processing Methods prepare an Object for modification by the end-user and validate the data afterwards.
- Insert Methods assist in creating a new Object.
- I/O triggers validate the database and ensure that updates are performed consistently.

## **Inheritance**

The construction of a definition by incremental modification of other definitions.

*(Object Management Group)*

In OPENworkshop all Classes are created and modified incrementally, and any definition created is interpreted consistently wherever it is referenced. OPENworkshop Formats and Links are used to collect these definitions together.

## **Encapsulation**

Is the results (or act) of hiding the implementation details of an Object from its user.

*(James Martin)*

An Object may CONNECT to another OPENworkshop Object, and allow it to perform the operations it needs. When it has finished, it may return a value.

It is the combination of these concepts that allows developers to reduce application development and maintenance costs.

## **Polymorphism**

The ability to use the same expression to denote different operations.

(Ian Graham)

An expression or message can operate on Objects of different Classes. This type of re-useable code greatly reduces development and maintenance costs.

### **Recursive**

Objects can send messages to their own Methods recursively or send messages to themselves.

(Ian Graham)

The ability to interrupt an action to undertake another action or subroutine, and then to interrupt this again with the same subroutine and so on. For the Thoroughbred Environment this is equivalent to Public Programs.

### **Persistence**

The property of an object by which its existence transcends time.

(Object Management Group)

The value of data remains after the Class or Method that created it no longer exists. An example is Data Objects that can be stored in files, which is the ultimate form of persistence.



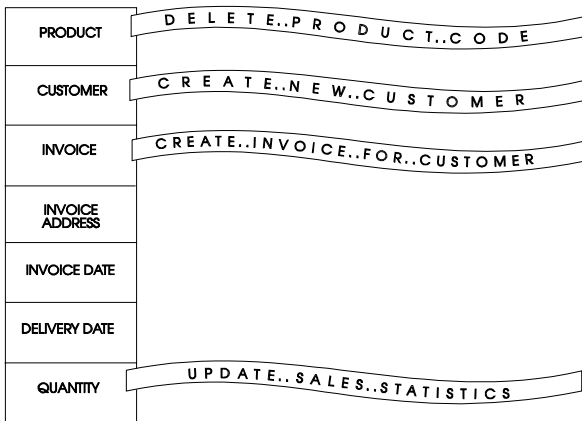
# OPENWORKSHOP SYSTEM CONCEPTS

Traditional software architectures are hierarchical in design, driven by a number of programs. Typically, users can select a program from a set of choices on a menu. Programs engage in a dialog with the user through data input screens, and may display data in lists and tables. The application program controls both the dialog with the user and the maintenance and updating of data items.

OPENworkshop re-defines the relationships between data and program code, and between application designer and application user. Developers using OPENworkshop must design their Data Objects first, and then associate “Methods”, which are independent modules of program code, with data items.

Because individual data elements control their own code, OPENworkshop eliminates any concept of a MAIN program. In place of a control structure that is imposed on the user through menus and escape keys, (all of which must be designed and planned for by the developer), OPENworkshop provides CONNECT directives, pop-up menus and user interface support that allows the user to control the dialogue with the system.

In OPENworkshop the data controls the application code, not the other way around. The system architecture enables the developer to partition the application code into Methods associated with Objects.



Because Objects are then packaged with their Methods, they are easy to re-use. Previous work is immediately available to use when creating other Objects with similar properties. Future modifications are automatically reflected in all places where the Object is used.

These benefits are important during initial application design, but even more important during the much longer maintenance life of an application.



# THE USER INTERFACE

OPENworkshop Presentation Classes (View, Screen, Menu, Report, Query, and Help) are used by developers to implement the user interface for the application. These are described in more detail later, but the OPENworkshop user interface contains some important features that are highlighted below.

## Mix both Character Terminals and Graphical Workstations

OPENworkshop offers a unique choice for the physical presentation of these Classes. The user may be equipped with a character terminal or a graphical workstation. Graphical workstations are PC-based workstations running a Microsoft operating system.

What is more, all of these options can be mixed in a single application site, with a single set of code. OPENworkshop manages the communication with the specified user interface device as appropriate.

It would not be uncommon to implement a business system on OPENworkshop with a mixture of character terminals and graphical workstations.

## Views

A View presents information in rows and columns. Like a spreadsheet, it is a form of presentation that is dense in information, and highly flexible. People naturally relate to this layout, once they see how flexible it is.

The Views shown here give information about customers and invoices.

The Views allow the user to explore information throughout the system. As the user moves around, options for action are presented. If the user is operating a character terminal, options are associated with Function

Keys, and the options available are described on the top line of the display. On a graphical workstation, options are displayed as a list, and may be selected using the mouse.

Cust Code	Cust Sales	Customer Contact	Customer Name	Sr Cd	Customer City
100100	15843.00	Tex Rogers	Toot-Your-Horn	AF	Port Lavaca
100102	16253.36	David Kelly	Fix-M-Up	JJ	Madison
100103	16628.04	Sue Thompson	Computer Inc.	JS	Bridgewater NJ

Inv Number	P F	Invoice Date	Cust Code	Sr Cd	Invoice Tot Amount	C M
000001		09/03/95	100100	AF	241.93	
000012		09/14/95	100100	AF	2903.13	
000023		09/25/95	100100	AF	5564.32	

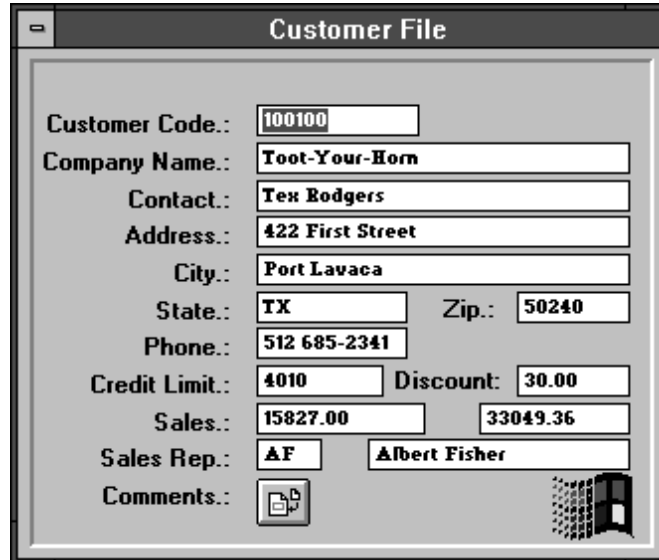
Inv Number	Ln Nm	Cust Code	Sr Cd	Item Code	Item	Price	Tax	Dscent %	U P
000012	01	100100	AF	10-100-000	1.40	16.80	1.18	30.00	
000012	02	100100	AF	10-100-010	3.50	42.00	2.94	30.00	
000012	03	100100	AF	10-100-020	4.20	50.40	3.53	30.00	
000012	04	100100	AF	10-100-030	5.60	67.20	4.70	30.00	
000012	05	100100	AF	10-100-000	5.60	67.20	4.70	30.00	

OPENworkshop allows the developer to specify a different list of options for each column. Thus the options can relate directly to the Data Object currently highlighted.

## Screens

In contrast to a View, an OPENworkshop Screen presents information for one instance of a data set at a time, set out as a form. This layout is used to collect or display detailed information where numerous collections of data are shown simultaneously.

A Screen supports the same Pre- and Post-processing options as a View, and therefore allows the user identical flexibility to move to other Classes, modify or lookup data, and return.



Customer Code.:	100100		
Company Name.:	Toot-Your-Horn		
Contact.:	Tex Rodgers		
Address.:	422 First Street		
City.:	Port Lavaca		
State.:	TX	Zip.:	50240
Phone.:	512 685-2341		
Credit Limit.:	4010	Discount:	30.00
Sales.:	15827.00		33049.36
Sales Rep.:	AF	Albert Fisher	
Comments.:	[Document Icon] [Grid Icon]		

## Menus

OPENworkshop offers the developer a number of menu types. The simplest option is the classical “list” of options from which to choose.

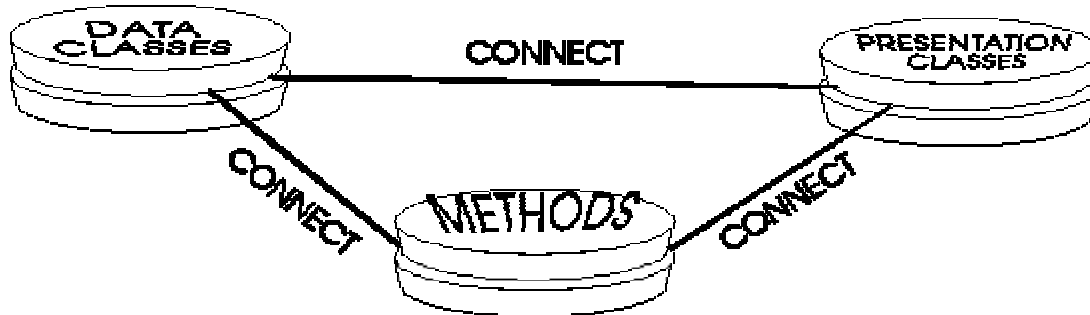
A Selection Menu allows the user to select values for parameters before selecting the option to be activated. The parameter values are passed to the selected Method.

A Matrix Menu displays options in a matrix. The user moves the highlight to the required cell in the matrix, and then selects it. This form of presentation can significantly reduce the number of layers of Menu required in many applications.

## Help

The Help system allows the developer to build Help into all levels of the user interface. At each step, Help can be displayed as a simple instruction, or a menu of levels. The Help system also supports a Subject Index, which the user can use to seek help by using keywords.

# OPENWORKSHOP CLASSES AND METHODS



An OPENworkshop application is built using Data Classes, Presentation Classes, and Methods. CONNECT directives are used to invoke instances of a Class.

## DATA CLASSES

Data Classes hold the application data. From the point of view of the application developer the most important Classes of Object are:

### Data-name

Defines an item of information, together with its attributes. A Data-name definition also specifies Methods to be used whenever the data item is created, displayed, or amended.

### Format

Collects together a set of Data-names that will be stored together in a single table in the OPENworkshop table or file system.

### Link

Specifies the physical files or tables that will be used to store a Format and associated key indexes. It also specifies default Presentation Classes (View and Screen) to use to display the data and allow it to be edited. Specifies the Trigger Method to be used when any data in the Link is updated to ensure that Referential Integrity is maintained throughout the system.

### Library

Collects together all the Classes relevant to an application or subsystem.



# PRESENTATION CLASSES

Presentation Classes display or print information for the application user to read or edit. They also allow the user to select subsequent actions.

## **View**

Displays data items, in spreadsheet form, as a table of rows and columns. Each column represents a different Data-name. Each row represents a record. Where there is more information than can fit in a window, vertical and horizontal scrolling is provided.

Views display information from a single Link, and from multiple joined Links combining data items. Views can also display calculated values.

Views allow the displayed data to be modified, and rows to be added to or deleted from the file or table being displayed. They also allow users to “drill down” or explore related information.

## **Screen**

Displays data items as a “form”. Allows the data to be modified and records to be added to or deleted from the file or table being displayed.

## **Message**

Displays a message and allows limited user input in response.

## **Report and Query**

Provides output to a printer or terminal as formatted reports containing application data and, if required, calculated values.

## **Menu**

Presents a set of options for the user to select. Menus can be presented as a simple list or as a matrix of options.

## **Help**

Displays context-sensitive Help.

## **METHODS**

Methods contain the program code that performs the logic of the application. In an OPEN*workshop* application all program code is organized into Methods, each being associated with specific user actions or system functions.

### **View Method**

Called whenever a View is preparing a row to display. Ensures that all data items required for display are available, and calculates any values required by the View.

### **Pre-Processing Method**

Called whenever a View or Screen is preparing to allow a user to edit a data item.

### **Post-Processing Method**

Called whenever a View or Screen has completed editing a data item. May be used to provide complex data validation or to modify related data items in the current row, for example.

### **Insert Method**

Called whenever a View is required to add a new row. Can be used to set initialized values or to verify that the addition of a new row should be allowed.

### **Link Trigger Method**

Called whenever a record in a file is to be updated. The Trigger Method is responsible for ensuring that the Data-names are being updated according to application rules and that any associated data will also be updated appropriately. Trigger Methods are the means by which OPEN*workshop* applications ensure that Referential Integrity is preserved throughout the application.

### **File Suffix Method**

Builds up a file suffix. Is used to manage files in directories.

### **After Read Method**

Called whenever a record has been read by a Screen. Prepares the record for display or editing.

### **Application Method**

Implement the Application logic. Users can initiate a Method by selecting the appropriate option from a menu. Methods can also call other Methods. Such Methods may be designed to perform any purpose the application designer needs.

# COMMUNICATING BETWEEN CLASSES

## The CONNECT Directive

The CONNECT directive is the most important directive in OPENworkshop. It allows “connections” to be made from one Object or Class to another. Views, Screens, Menus, Help, Reports and Queries can be connected directly between themselves.

CONNECT	From Data Element Name Post-Process	From Menu
HELP	As data element name pre-process	As data element name pre-process
MENU	As data element name pre-process	As data element name pre-process
METHOD	Method return directives will not be evaluated	Method return directives will not be evaluated. If the menu is a selection type menu then selection parameters will be passed to the called method in the MSG\$[2] string array.
QUERY	As data element name pre-process	As data element name pre-process
REPORT	As data element name pre-process	As data element name pre-process
SCREEN	As data element name pre-process	As data element name pre-process
VIEW	As data element name pre-process	As data element name pre-process

The power and flexibility of OPENworkshop is greatly enhanced by allowing intervening Methods to modify the behavior of the Classes and the behavior of the connections between them.

The CONNECT directive is the framework that links OPENworkshop Classes and Methods together. It not only invokes the required Object, it is also the vehicle for passing messages to the invoked Object.

Developers can CONNECT from a View, Screen, Menu, or Method to any of the OPENworkshop Classes or Methods. You can use CONNECT within a Class. For example, a View can use CONNECT to initiate another View, or even to create a second instance of the same View.

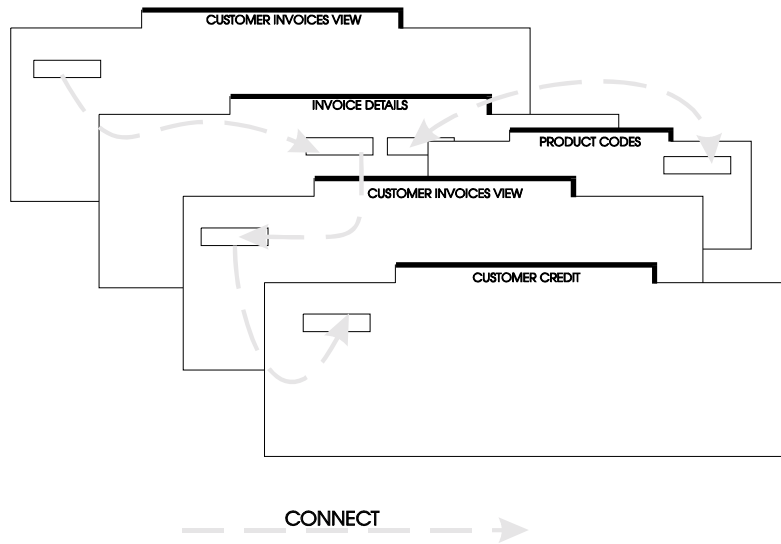
Messages carried by the CONNECT directive are used to pass information to the invoked Class or Method. They can be used to communicate information about the Object that the target should work on, or modify the behavior of the target.

CONNECT directives are simple in concept, uncomplicated to define, but extremely powerful in execution.

### CONNECT From a View or Screen

For a View or a Screen, CONNECT directives are placed in the Data-name Pre- and Post-Processing definitions in the associated *Dictionary-IV* Format. A Pre-Process is executed whenever the Cursor is positioned on the Data-name in a View or a Screen. A Post-Process is executed when a user has completed editing a Data-name.

Note that these CONNECT definitions are associated with a Data-name. In other words, having been specified once, they are in effect wherever the Data-name is used, possibly in many Views and Screens throughout the application. One definition is all you need to make and maintain! What is more, you can guarantee the behavior of the system is consistent for the user, wherever that Data-name is displayed.



### CONNECT to a View

The CONNECT directive allows you to define which rows should be placed in the View, and in which order to sort them.

Customer File						
Cust Code	Cust Sales	Customer Contact	Customer Name	Sr Cd	Customer	State
100100	15827.00	Tex Rogers	Toot-Your-Horn	AF	Port Lavaca	TX
100102	16628.04	Sue Thompson	Computer Inc.	JJ	Madison	NJ
100103	16951.04	Robert Brock	Today's Company	JS	Bridgewater	NJ
100104	17222.36	Sarah Smith	ACME Inc.	AF	Dayton	NJ
100105	17498.45	Walter Snider	Lumber Inc.	HP	Big Horn	ND
100106	17609.96	Dennis Gohlke	OK Development	JJ	Port Lavaca	TX
100107	10310.16	John Dworaczyk	Memory Lanes	JS	Victoria	TX

Rows in a View can be Selected and Sorted

The general structure of the CONNECT VIEW directive simple to use and understand, as is shown in the following syntax:

```
CONNECT VIEW "View-name"  
[USING Data-name or Expression]  
| [USING RANGE FROM Data-name or Expression;  
    TO Data-name or Expression;]  
| [SELECT WHEN Expression;]  
[SORT BY n]
```



# DEVELOPMENT ENVIRONMENT

In addition to the focus on usability for the OPEN*workshop* end-user, Thoroughbred has concentrated on providing a highly flexible and accessible environment for developers. To a great extent this has been achieved by the simple tactic of implementing the developer's interface to the system using OPEN*workshop* Classes and Methods.

The developer's environment positively encourages an incremental development methodology. All OPEN*workshop* Classes are easily available to the developer at all times, and can be modified as the application is running. Most changes that the developer can make are effective immediately, so the developer can see the results immediately.

For example, when a Menu is displayed the developer can select "Edit" to edit the definition of that Menu. When the editing is finished, OPEN*workshop* re-displays the Menu in its revised form, so the developer can see and check the results immediately. The same facility is provided for View, Screen and Help. Similar capabilities exist for Link, Format, and Data-name.

OPEN*workshop* also provides the developer with a comprehensive testing and debugging toolkit. Data and control status can be inspected at any time through an online debug system.

A complete cross-referencing system shows the structure and relationships within the system for the developer. These relationships are displayed using OPEN*workshop* Views, thus providing all the search and selection facilities the developer needs. In addition, the developer can point to any item in the View, and select it to display or edit the OPEN*workshop* Object directly, further increasing the high degree of accessibility of OPEN*workshop* Class definitions.

Any environment that is so open to developers needs to be protected from accidental or over-inquisitive access by end users. OPEN*workshop* has security protection that disables or enables this access, by passwords and user logins.



# GETTING STARTED IN OPENWORKSHOP

Thoroughbred makes getting started with OPENworkshop fast and efficient. Most important of all, it will provide the developer with the tools to succeed. As with any new technology there is some learning to do.

OPENworkshop offers the developer strong benefits in productivity, as well as a more flexible user interface. But to gain from these benefits it is essential that developers invest in training. In fact, Thoroughbred is so convinced of this need that it is a *requirement* that any new customer have at least one member of staff attend a training course. Having taken this step, the developer will be well positioned to move forward.

Developers who already have experience with Thoroughbred's *Dictionary-IV*, and have applications that are based on it, can migrate their applications to OPENworkshop easily. Many *Dictionary-IV* items can be used directly, and Thoroughbred provides conversion utilities for others.

While experienced *Dictionary-IV* developers will be at an advantage, OPENworkshop features some fundamental new concepts that must be understood. Therefore Thoroughbred requires that they too take part in training before beginning to develop in OPENworkshop.

OPENworkshop runs on most popular operating systems<sup>2</sup>. Once installed on the selected hardware and operating system, an example application is available for developers to use as a starting point for their subsequent development activities.

Developers will find that it is easy and quick to create a prototype skeleton of their intended application. In fact this is the best way to familiarize yourself with the environment. After this reflect on the initial design decisions made for the prototype. Then re-specify the Classes before starting to develop the first real application. The OPENworkshop facilities positively encourage this incremental development style.

OPENworkshop gives you tangible, demonstrable results very quickly. It is an ideal environment to use if your business needs to customize applications for customers, or to develop new applications for customer requirements. You can show your customer the prototype as it is developed, and amend it step by step with them.

---

<sup>2</sup> UNIX, VMS, Windows 3.x, Windows 95, Windows NT.



# BIBLIOGRAPHY

## **Object Analysis and Design Comparison of Methods**

Author: Object Management Group  
Publisher: John Wiley and Sons

## **Principles of Object Oriented Analysis and Design**

Author: James Martin  
Publisher: Prentice-Hall

## **Object Oriented Methods**

Author: Ian Graham  
Publisher: Addison-Wesley

## **Object Lifecycles: Modeling the World in States**

Author: Sally Shlaer & Stephen J. Mellor  
Publisher: Prentice-Hall



# OPENWORKSHOP AT YOUR FINGERTIP

## *Definition's*

### **Object Oriented**

*Data controls the application*

*Applications can be developed incrementally*

*Relational Integrity Maintained*

*Can Modify Object Rules*

*Global Definition Based*

### **Object Classes**

#### ***Data Classes***

##### **Data Name**

Defines Item attributes

Specifies Methods used when data is created, displayed or amended

##### **Format**

Defines the Data item, and how it is stored

##### **Link**

Specifies Files and Indexes

Specifies Default Presentation Class

Specifies Trigger Method for updates

Insures data and Referential Integrity

##### **Library**

Catalogues the relevant Classes to an application or subsystem

**Presentation Classes**

**View - It is more than a browse**

Browse - Displays Data as Rows and Columns

Multiple Link joins

Calculated values

Drill-downs

Modify Data

**Screen**

Displays Data as a Form

**Message**

Displays a message with user response possible

**Menu**

Catalogs options for user selection including list and matrix

**Help**

Context sensitive help text

**Report**

Custom and systems report library

Programmable report designer supporting

Report design and formatting

Calculated and prompt entered data

**Query**

SQL Query/Report generator

Point and click user interface

GWW connection to spreadsheets

## **Methods**

### **View Method**

Called when a View prepares a row to display  
Ensures that data items are available, and  
calculates values required by the View

### **Pre-Processing Method**

Called when a View or Screen allows data entry  
or modification

### **Post-Processing Method**

Called when a View or Screen completes editing  
a Data item  
Data validation

### **Insert Method**

Called when a View or Screen adds a new row  
Default values and verification of new  
data

### **Link Trigger Method**

Called when a record is being updated  
Ensures that Data-names are being updated  
according to the application rules  
Ensures Referential Integrity

### **After Read Method**

Called when a record has been read by a  
screen, and prepares for editing

### **Application Method**

Implements application logic and rules

## CONNECT

### *Links the Classes and Methods*

Invokes the required Object

Passes messages to the Object

Pre or Post Process

Initiate additional views, Screens, Menus, Queries, Reports, Messaging, Methods

### *Re-Entrant*

When invoked the current environment is preserved, and restored upon completion

## Object Concepts

### *Inheritance*

Obtain Characteristics from a Superclass

### *Polymorphism*

Same expression or message can operate on Objects of different Classes

### *Recursive*

Objects can send messages to their Methods recursively, or to themselves

### *Persistence*

Created Object transcends time and its creator

## Three Tier Compliance

### *Presentation*

**Graphic (GUI)** - with VIP4

**Character** - with same application

**HTML** - with VIP4

***Rules***

**OPENworkshop**

***Database***

**Thoroughbred**

**ORACLE**

**ODBC**

**Environments Supported**

***UNIX***

***VMS***

***MS Windows, 95, NT***