**Software Engineering Principles**

Principles

High Cohesion

Encapsulate data (classes, oracle types and packages, object-based programming)

Avoid run-time flags

Test a flag only once in a process path

Limit functionality of objects (procedures, classes, packages, etc.)

Small argument list

Separate Form and DB processing

Separate standard processing from exception handling

Separate methods and behaviors for an item from those of groups of items

Procedure name describes its behavior

Procedure name agrees with names of its arguments

Function name describes the return value

Low Coupling

Avoid system/program globals

(see also High Cohesion)

Self-Documenting

Names are meaningful, context-dependent, consistent, in agreement

Use defined/declared constants. Avoid literal constants

Procedure name documents what it does, not how it does it

Variable, Constant names document their contents, not their form

Visual Impact - visible blocks

Context-dependent unqualified names

Exceptions

Module, class headers document object services and inter-procedure processing

Procedure header documents how to use it

Loop counter and array index variables may be named i, j, r, c

Limited Tuning

Tune only on demonstrable need

Procedural Call-tree should reflect the structure of the data it processes

Consequences

Code Collapse - (reduce lines of code by 1/3 or more)

Easier to read (fewer comments, smaller code, clearer names)

Faster Development

More robust

Greater flexibility

Eliminate unnecessary tuning

Tuning is easier and has a smaller scope

Easier to maintain

Fewer side effects

Localized changes fix problems of all callers

Most errors caught during compilation

Fewer flags and globals

Less to learn

Less total code

Less ancillary code

Can choose level of learning

More to learn

Larger call trees

Get confidence in called procedures

Data

I/O

Input

Output

Internal

TradeProcessed=ProcessTrade(TradeText, Atrade)

ProcessTrade(TradeText,NewTrade)

Begin

read text

put results in NewTrade

NewTrade.time = TimeFromText(TradeText)

return True

X == ProcessTrade(A,B)