

Discovering and Rejecting Classes

- B. Meyer, Object-Oriented Software Construction, 2nd Ed, Prentice-Hall, 1997
- Meyer is author of Eiffel language/method

Sources of Class Ideas

Source of Ideas	What to look for
<i>Existing Libraries</i>	<ul style="list-style-type: none"> •Classes that meet needs of application
<i>Requirements Documents</i>	<ul style="list-style-type: none"> •Terms that occur frequently •Terms with explicit definitions •Terms with assumed meaning

Sources of Class Ideas (2)

Source of Ideas	What to look for
<i>Discussion with customers and users</i>	<ul style="list-style-type: none"> •Important abstractions in the application domain •Jargon of application domain •Both <i>conceptual</i> and <i>material</i> objects
<i>Documentation for similar systems in same domain</i>	<ul style="list-style-type: none"> •Useful design abstractions •See above

Sources of Class Ideas (3)

Source of Ideas	What to look for
<i>Non-O-O systems or system descriptions</i>	<ul style="list-style-type: none"> •Data elements passed as arguments •Shared memory areas •Important files •Records (or structs) used by several procedures of modules •Entities in ER modeling (DB)

Sources of Class Ideas (4)

Source of Ideas	What to look for
<i>Discussions with experienced designers</i>	<ul style="list-style-type: none"> •Design classes used successfully in other similar systems
<i>Algorithms and Data Structures literature</i>	<ul style="list-style-type: none"> •Known data structures with efficient operations
<i>OO design literature</i>	<ul style="list-style-type: none"> •Relevant design patterns

Rejecting Classes

Danger Sign	Suspicion
<i>Class with verbal name (infinitive or imperative)</i>	May be a procedure, not a class
<i>Class with only one public method</i>	May be a procedure, not a class
<i>Class described as "performing" something</i>	May not be a proper data abstraction

Rejecting Classes (2)

Danger Sign	Suspicion
<i>Class with no methods</i>	May not be an ADT, or missed methods
<i>Class involving several abstractions</i>	Should be split into one class per abstraction
<i>Class that inherits from another class, but has few features itself</i>	A case of taxonomic overkill