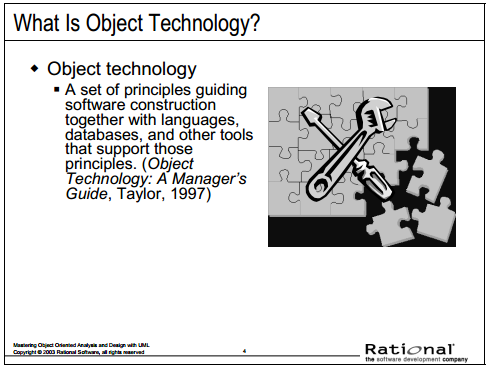
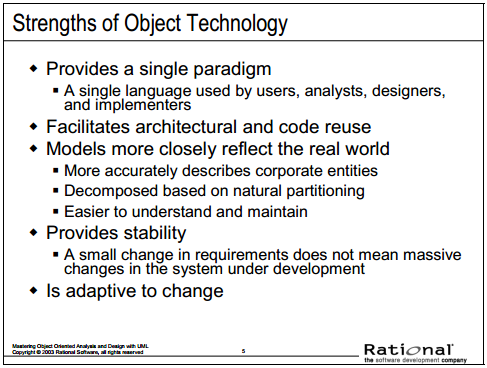
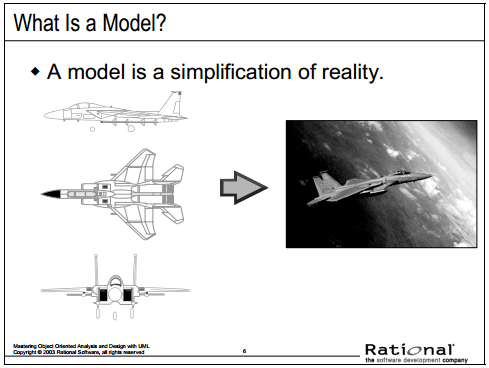
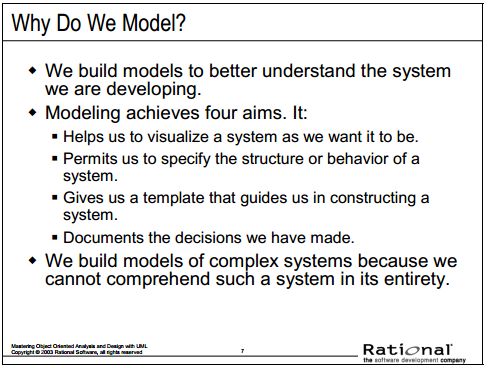
Apply Object-Oriented (OO) concepts: abstraction, encapsulation, inheritance, hierarchy, modularity, and polymorphism to the development of a robust design model

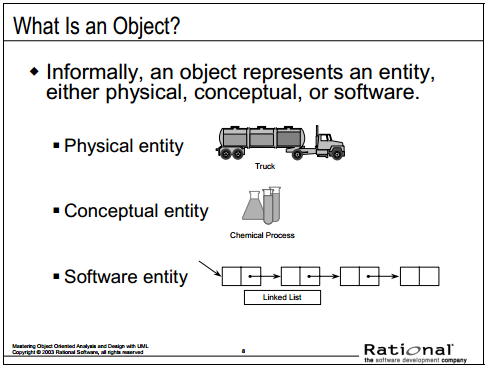
* **Concepts of Object Orientation**







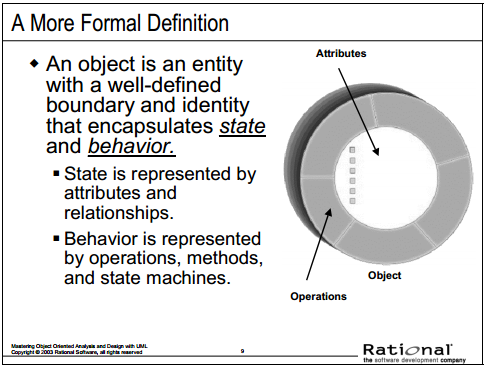


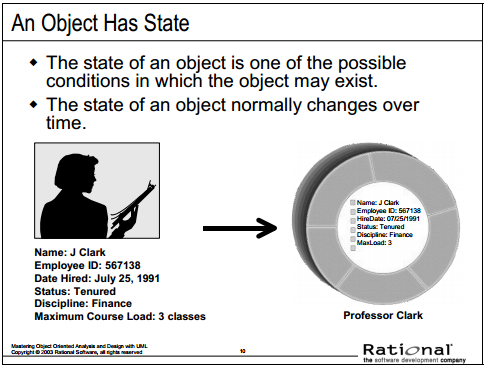


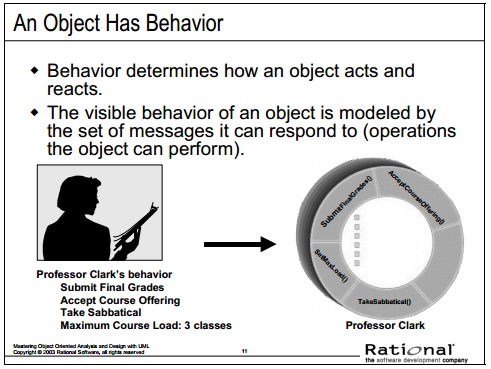
• Objects allow the software developer to represent real-world concepts in the software design. These real-world concepts can represent a physical entity such as a person, truck, or space shuttle.

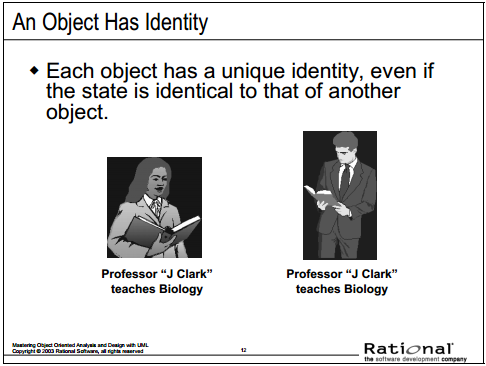
• Objects can be concepts like a chemical process or algorithms.

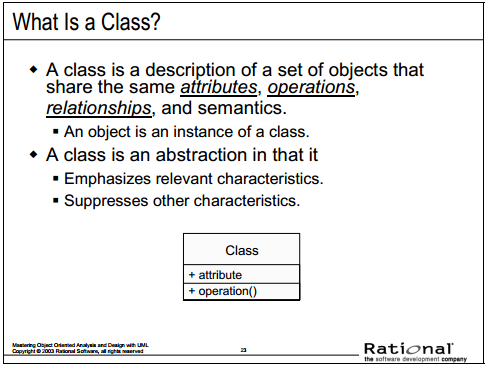
• Objects can even represent software entities like a linked list.

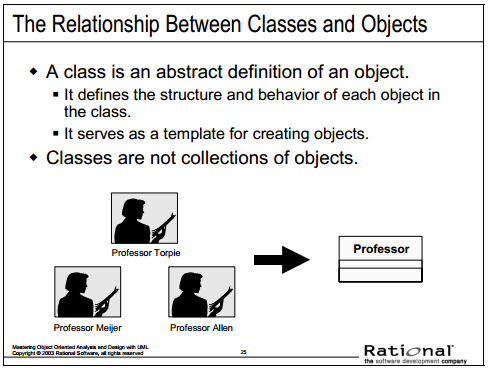


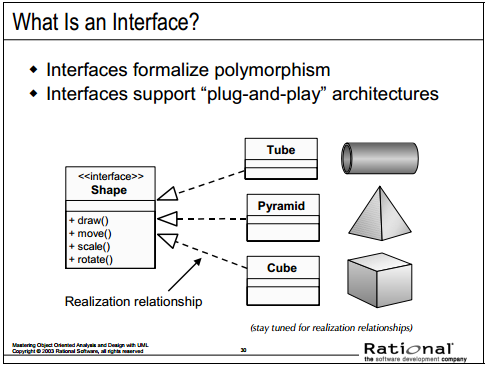


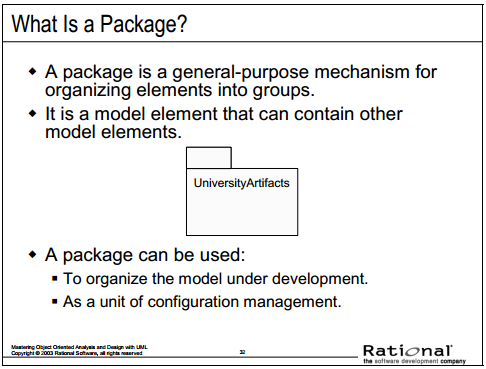


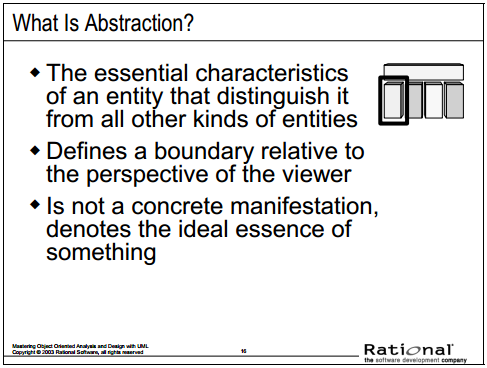


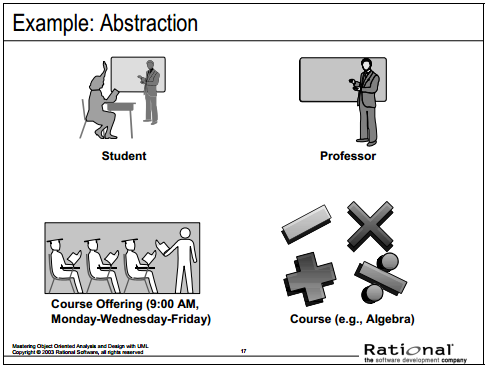












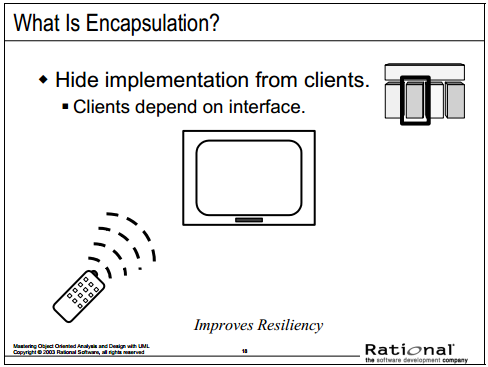
The following are examples of abstraction.

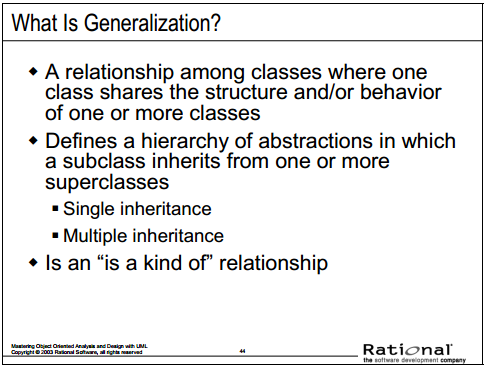
• A student is a person enrolled in classes at the university.

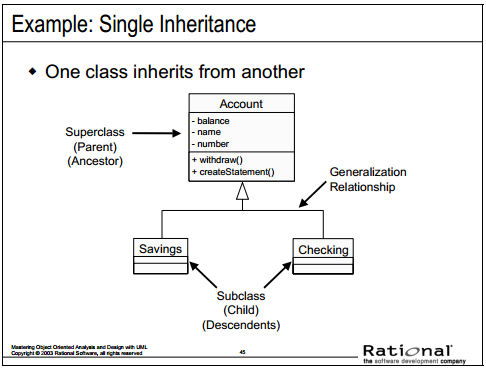
• A professor is a person teaching classes at the university.

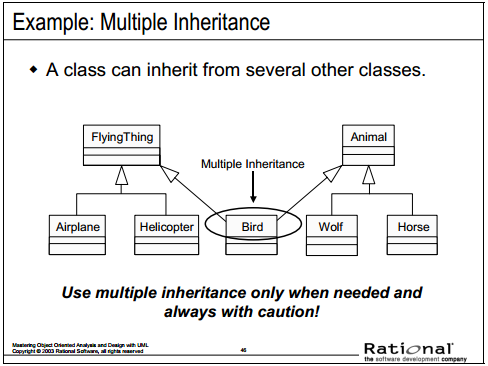
• A course is a class offered by the university.

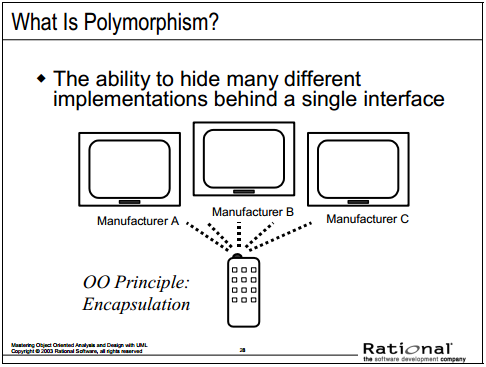
• A course offering is a specific offering for a course, including days of the week and times.

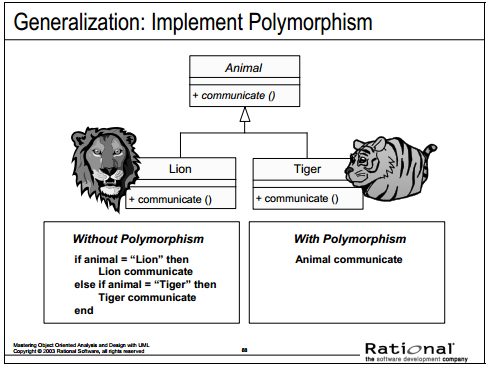


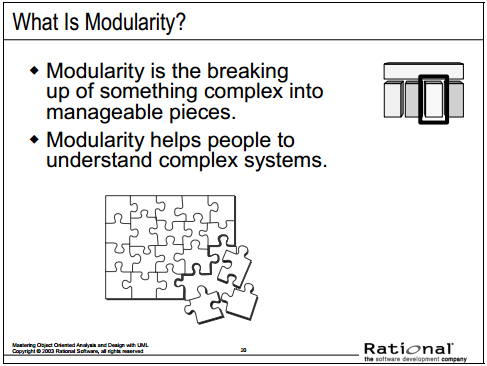


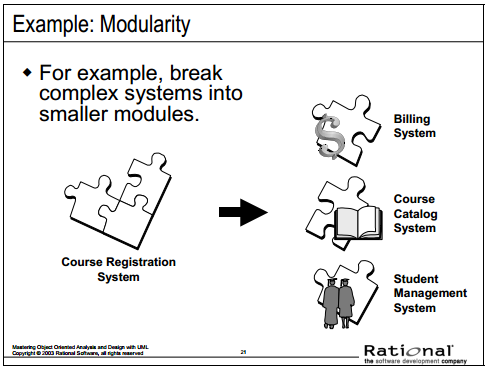












Often, the system under development is too complex to understand. To further understanding, the system is broken into smaller blocks that are each maintained independently. Breaking down a system in this way is called **modularity**. It is critical for understanding a complex system.

For example, the system under development is a Course Registration system. The system itself is too large and abstract to allow an understanding of the details. Therefore, the development team broke this system into three modular systems, each independent of the others.

• The Billing System

• Course Catalog System

• Student Management System

* **Analysis and Design Overview**

