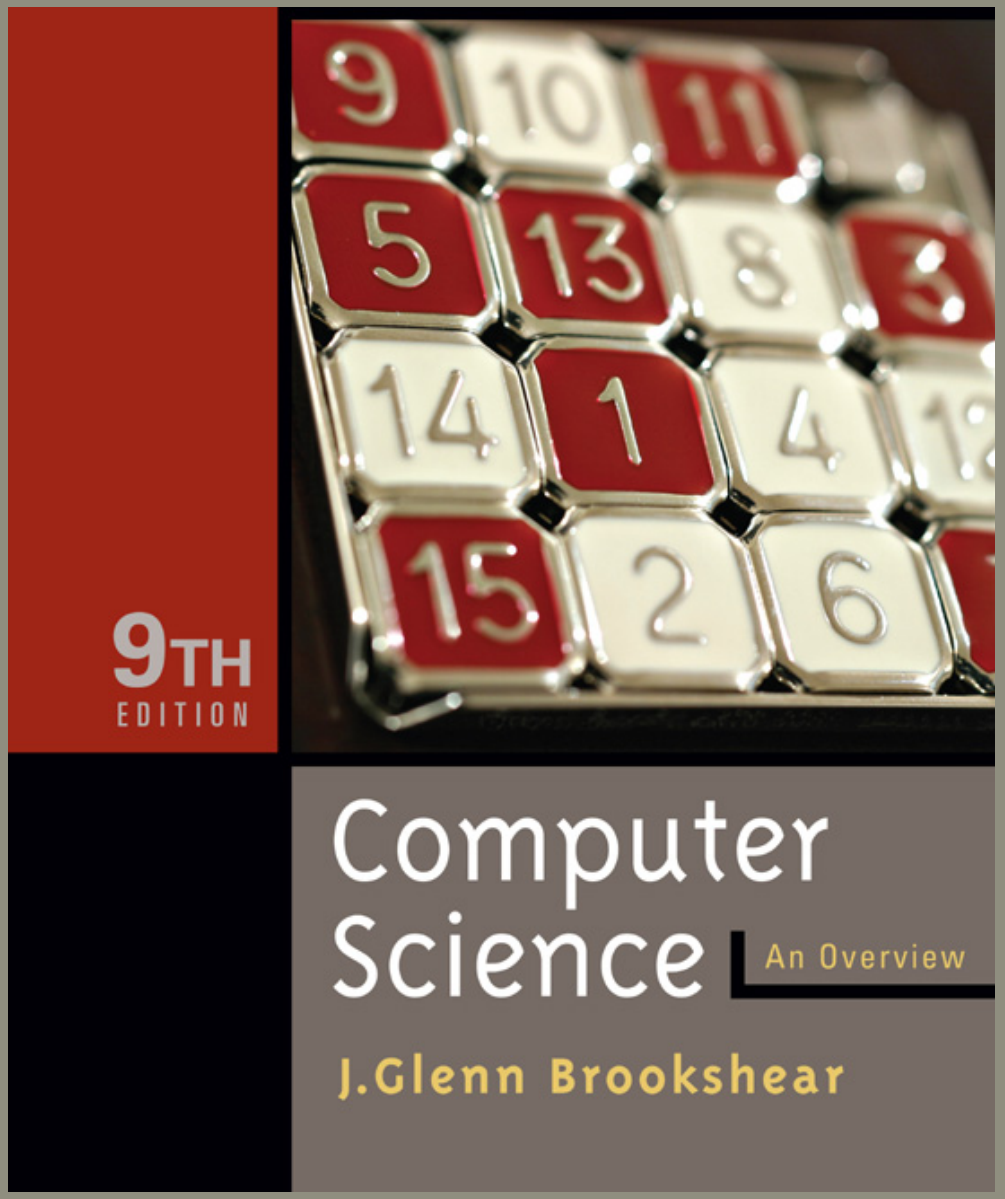
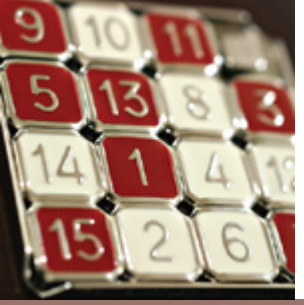
Chapter 7

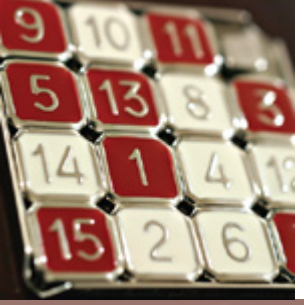
Software Engineering





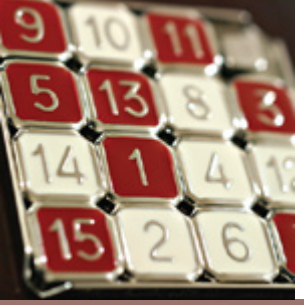
Chapter 7: Software Engineering

- 7.1 The Software Engineering Discipline
- 7.2 The Software Life Cycle
- 7.3 Software Engineering Methodologies
- 7.4 Modularity
- 7.5 Tools of the Trade
- 7.6 Testing
- 7.7 Documentation
- 7.8 Software Ownership and Liability



The Software Engineering Discipline

- Distinct from other engineering fields
  - Prefabricated components
  - Metrics
- Practitioners versus Theoreticians
- Professional Organizations: ACM, IEEE, etc.
  - Codes of professional ethics
  - Standards



Computer Aided Software Engineering (CASE) tools

- Project planning
- Project management
- Documentation
- Prototyping and simulation
- Interface design
- Programming

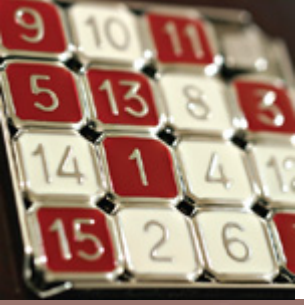
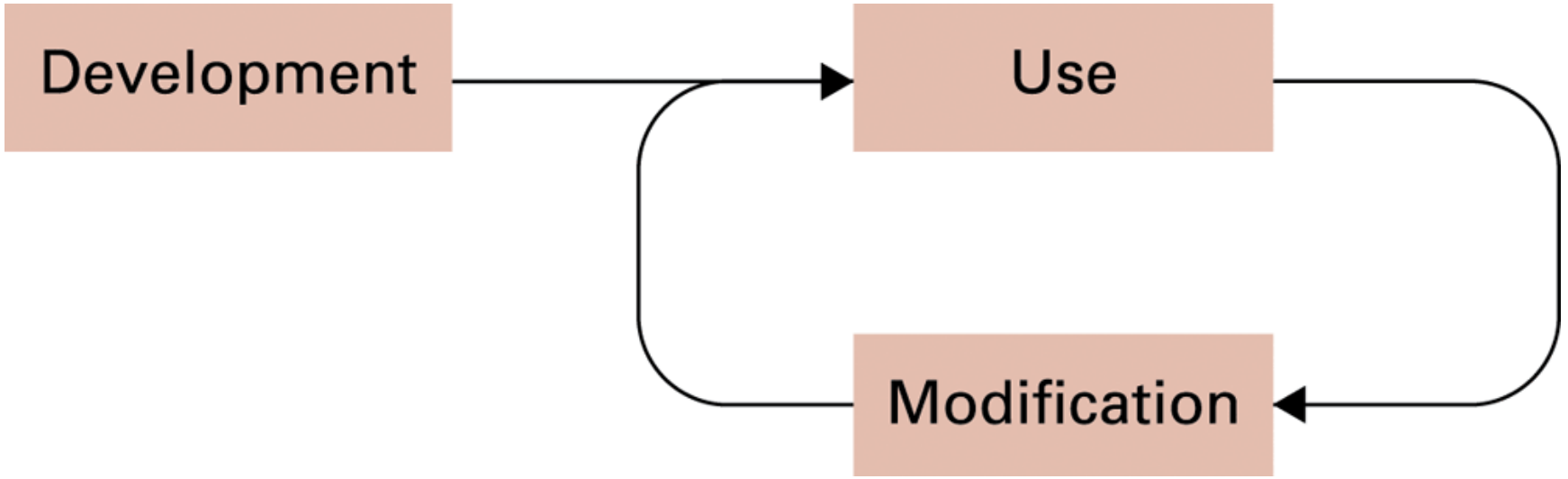


Figure 7.1 The software life cycle



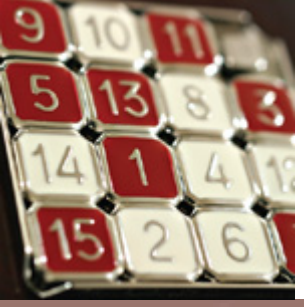
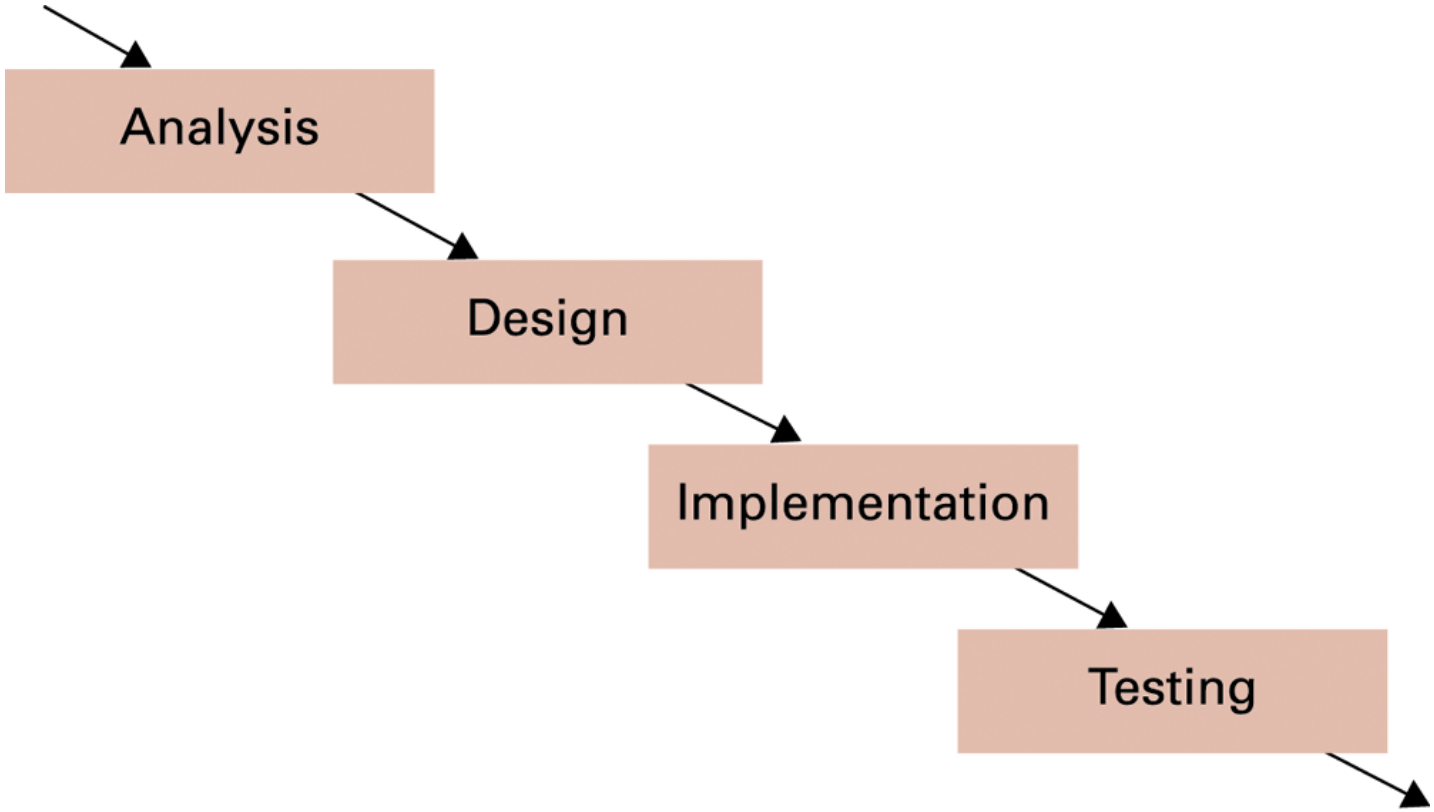
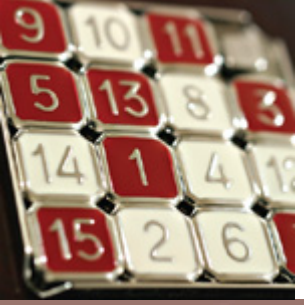


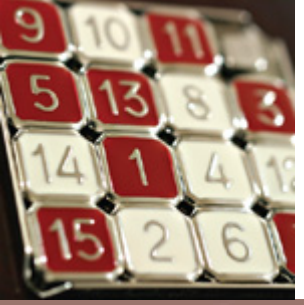
Figure 7.2 The development phase of the software life cycle





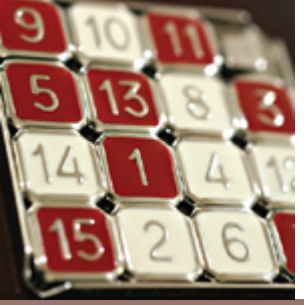
Analysis Stage

- Requirements
  - Application oriented
- Specifications
  - Technically oriented
- Software requirements document



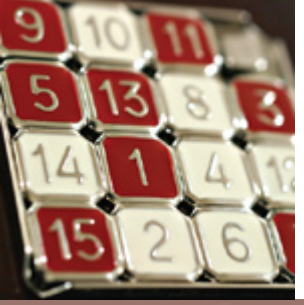
Design Stage

- Methodologies and tools (discussed later)
- Human interface (psychology and ergonomics)



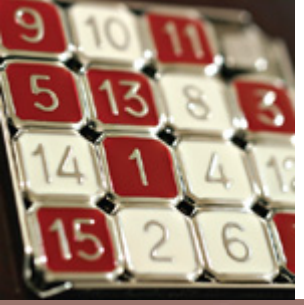
Implementation Stage

- Create system from design
  - Write programs
  - Create data files
  - Develop databases
- Role of “software analyst” versus “programmer”



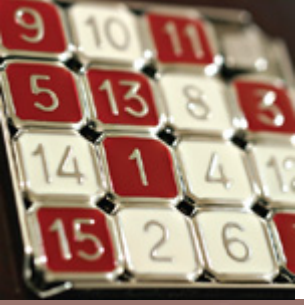
Testing Stage

- Validation testing
  - Confirm that system meets specifications
- Defect testing
  - Find bugs



Software Engineering Methodologies

- Waterfall Model
- Incremental Model
  - Prototyping (Evolutionary vs. Throwaway)
- Open-source Development
- Extreme Programming



Modularity

- Procedures -- Imperative paradigm
  - Structure charts
- Objects -- Object-oriented paradigm
  - Collaboration diagrams
- Components -- Component architecture

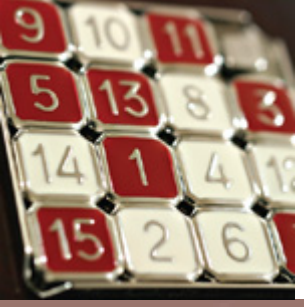
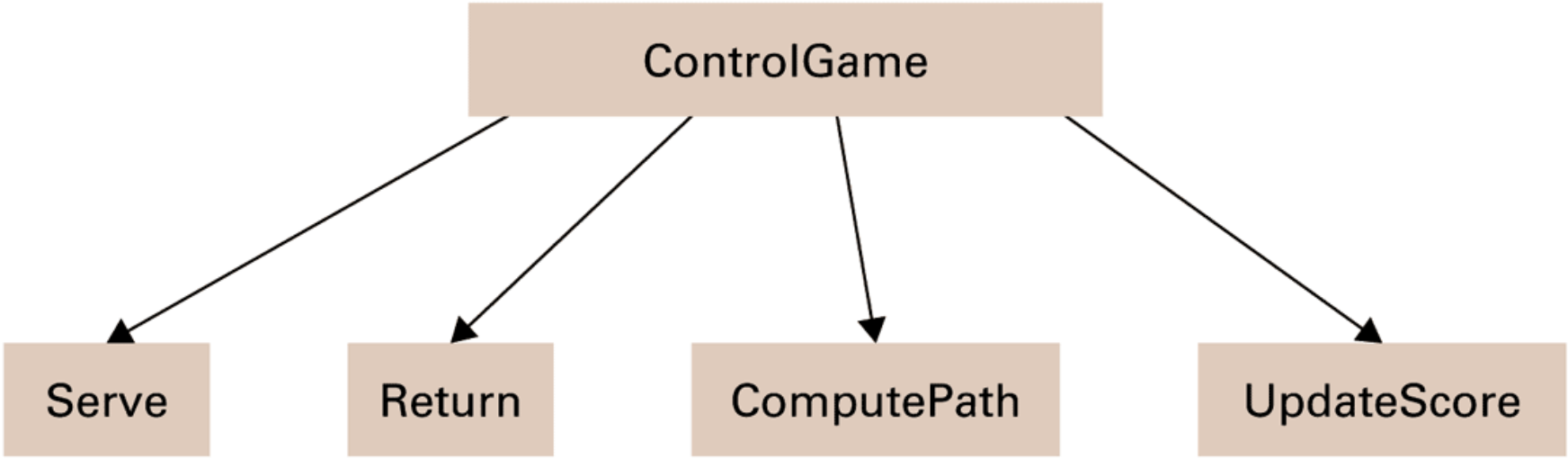


Figure 7.3 A simple structure chart



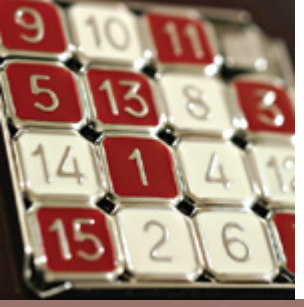
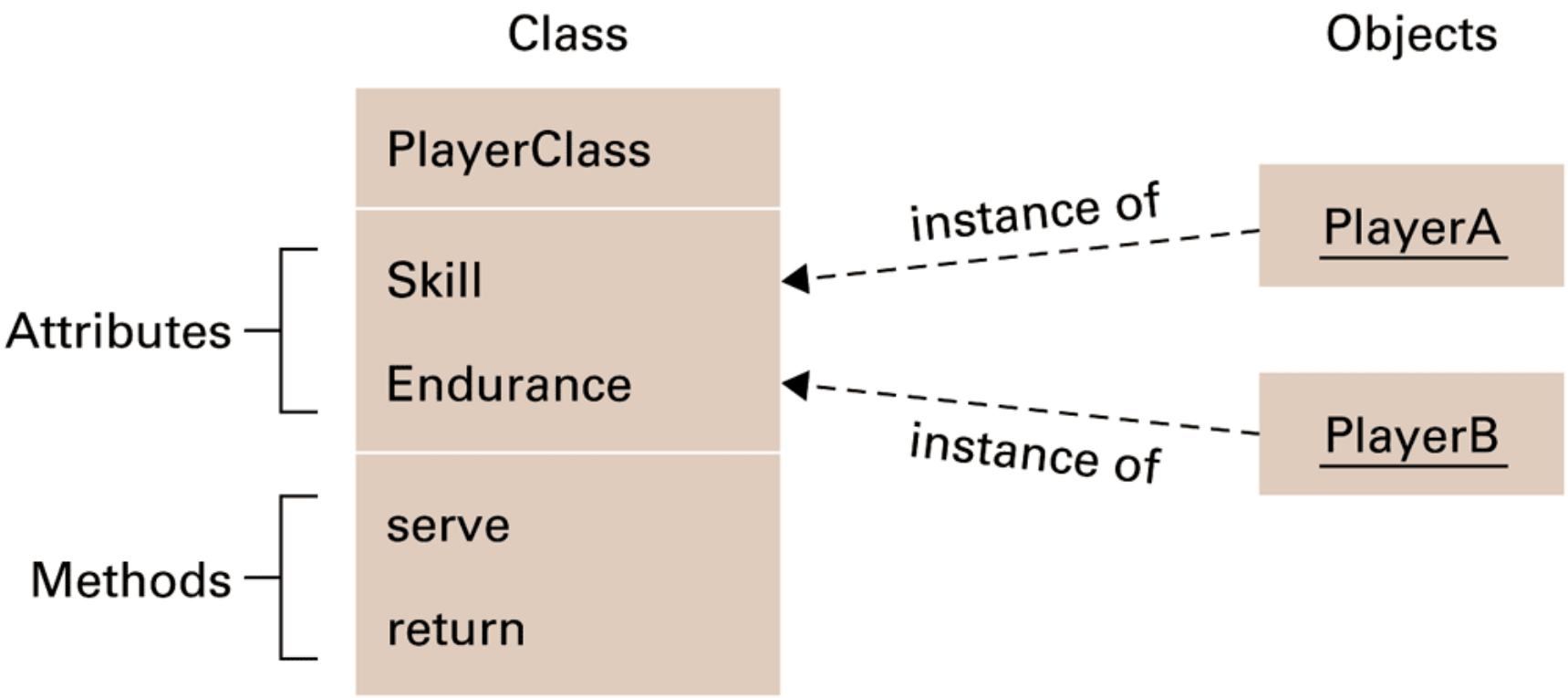


Figure 7.4 The structure of PlayerClass and its instances



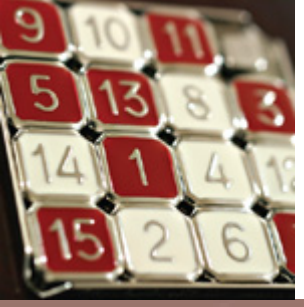
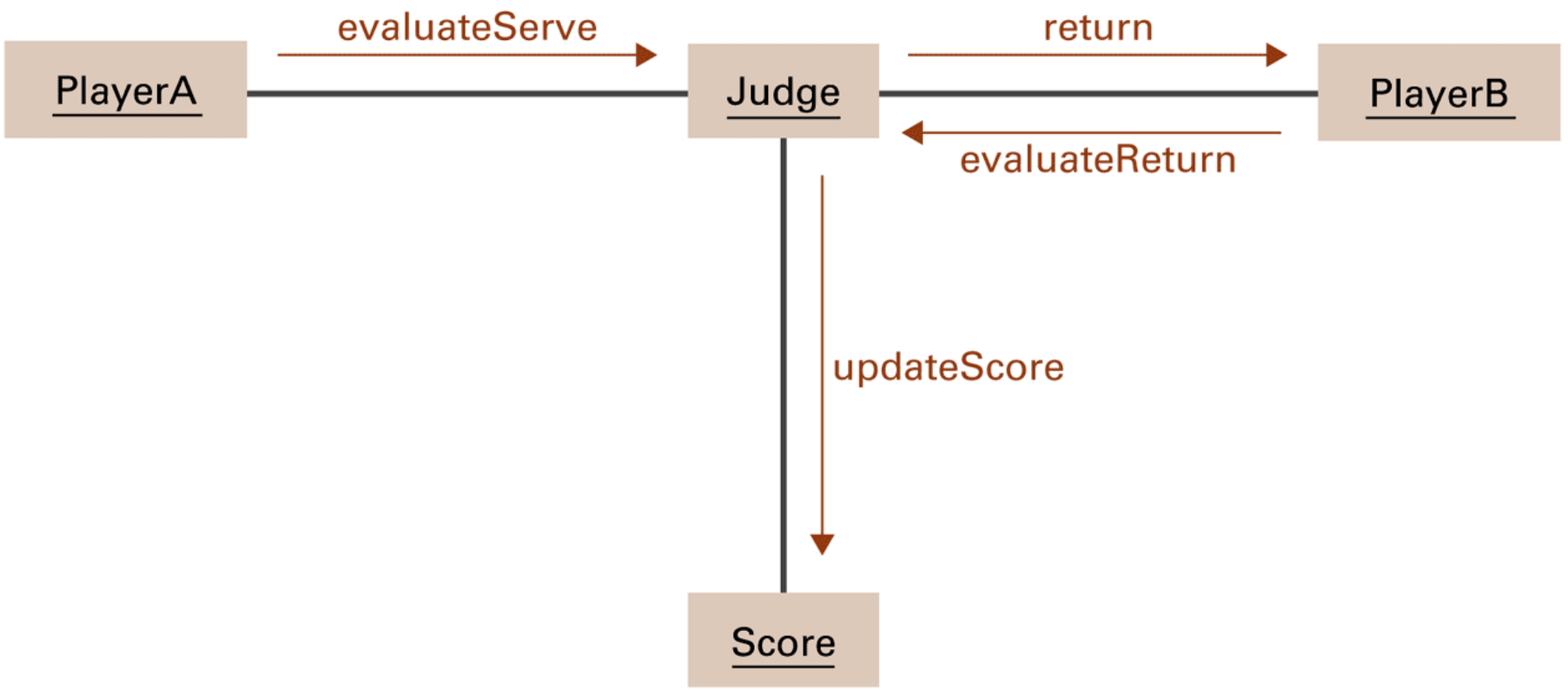


Figure 7.5 A simple collaboration diagram



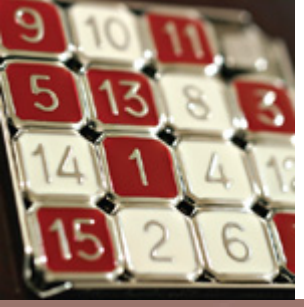
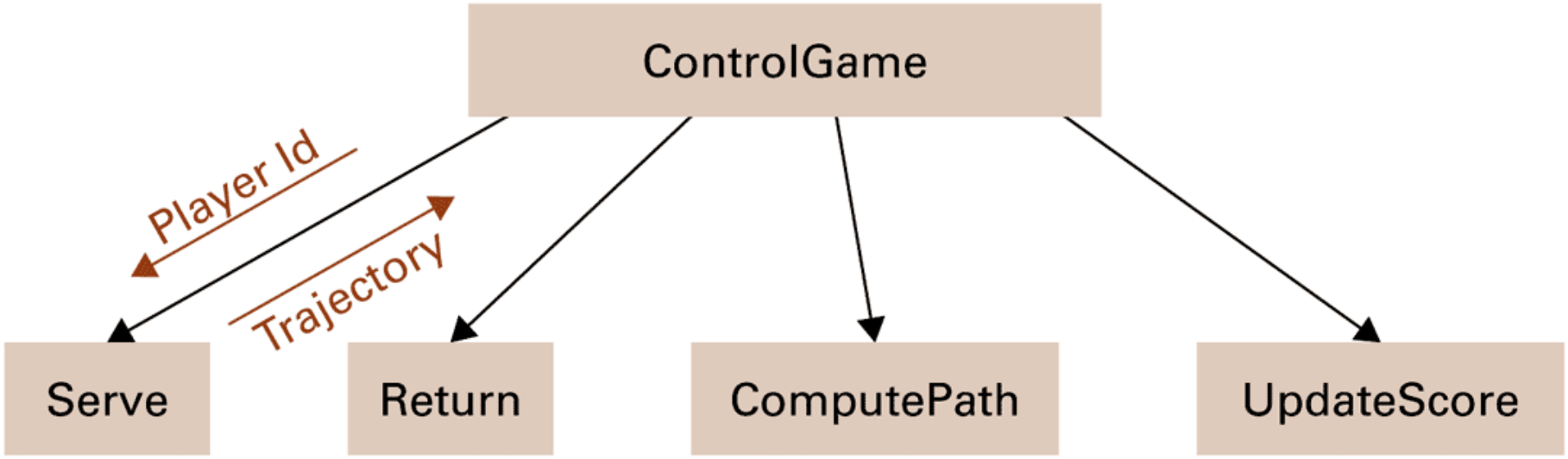
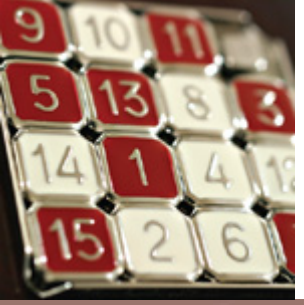


Figure 7.6 A structure chart including data coupling





Coupling versus Cohesion

- Coupling
  - Control coupling
  - Data coupling
- Cohesion
  - Logical cohesion
  - Functional cohesion

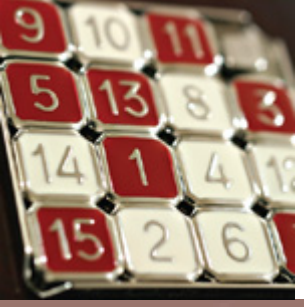
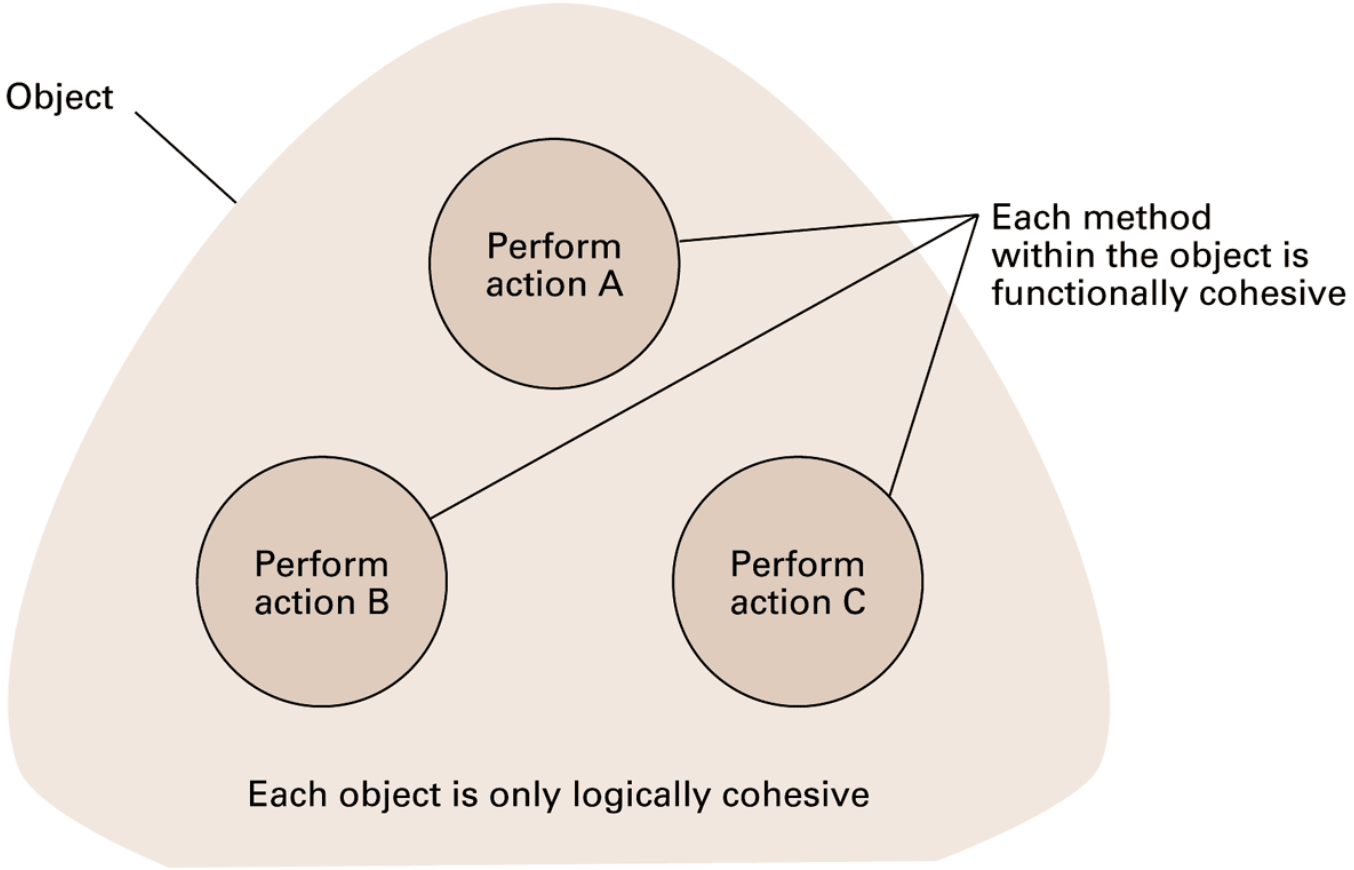
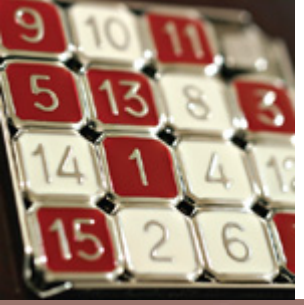


Figure 7.7 Logical and functional cohesion within an object





Tools of the Trade

- Data Flow Diagram
- Entity-Relationship Diagram
  - One-to-one relation
  - One-to-many relation
  - Many-to-many relation
- Data Dictionary

Figure 7.8 A simple dataflow diagram

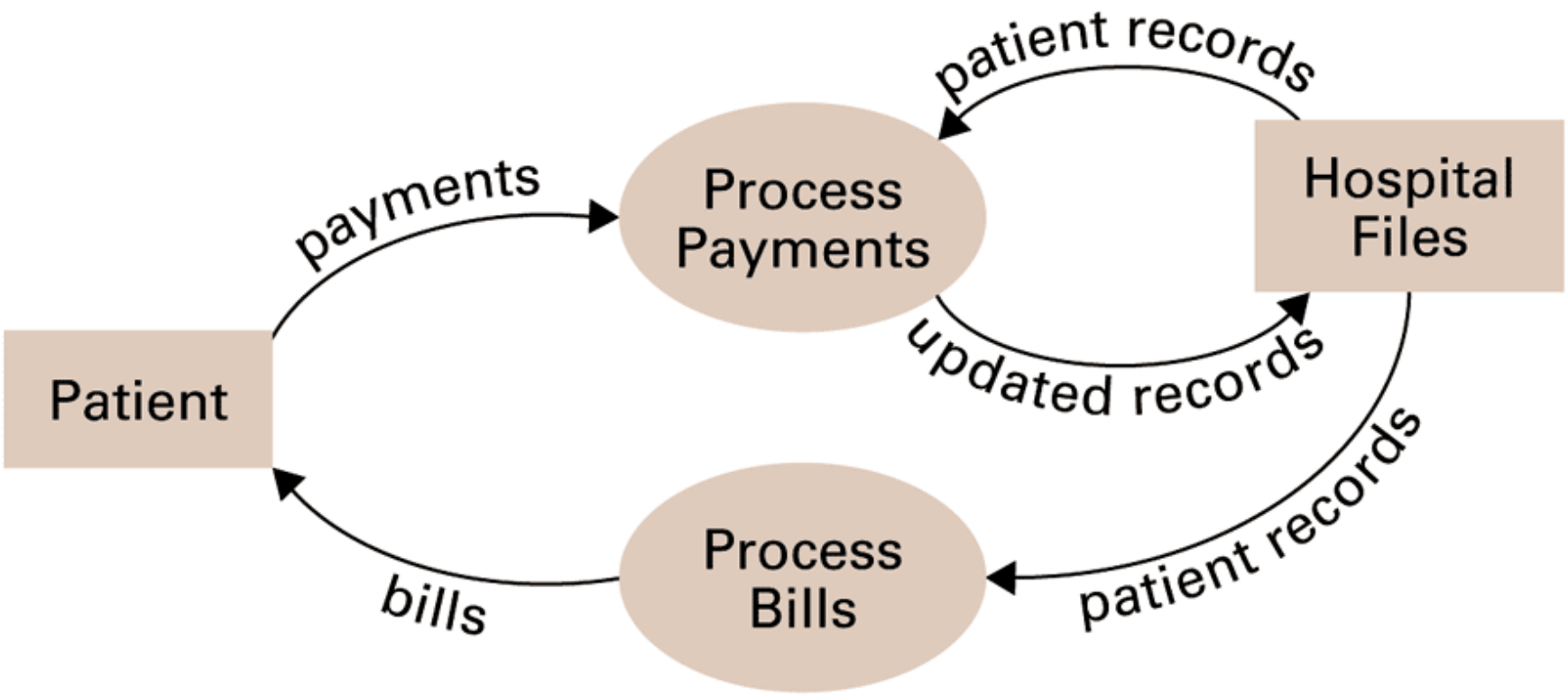


Figure 7.9 A simple entity-relationship diagram

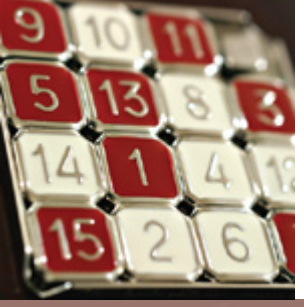
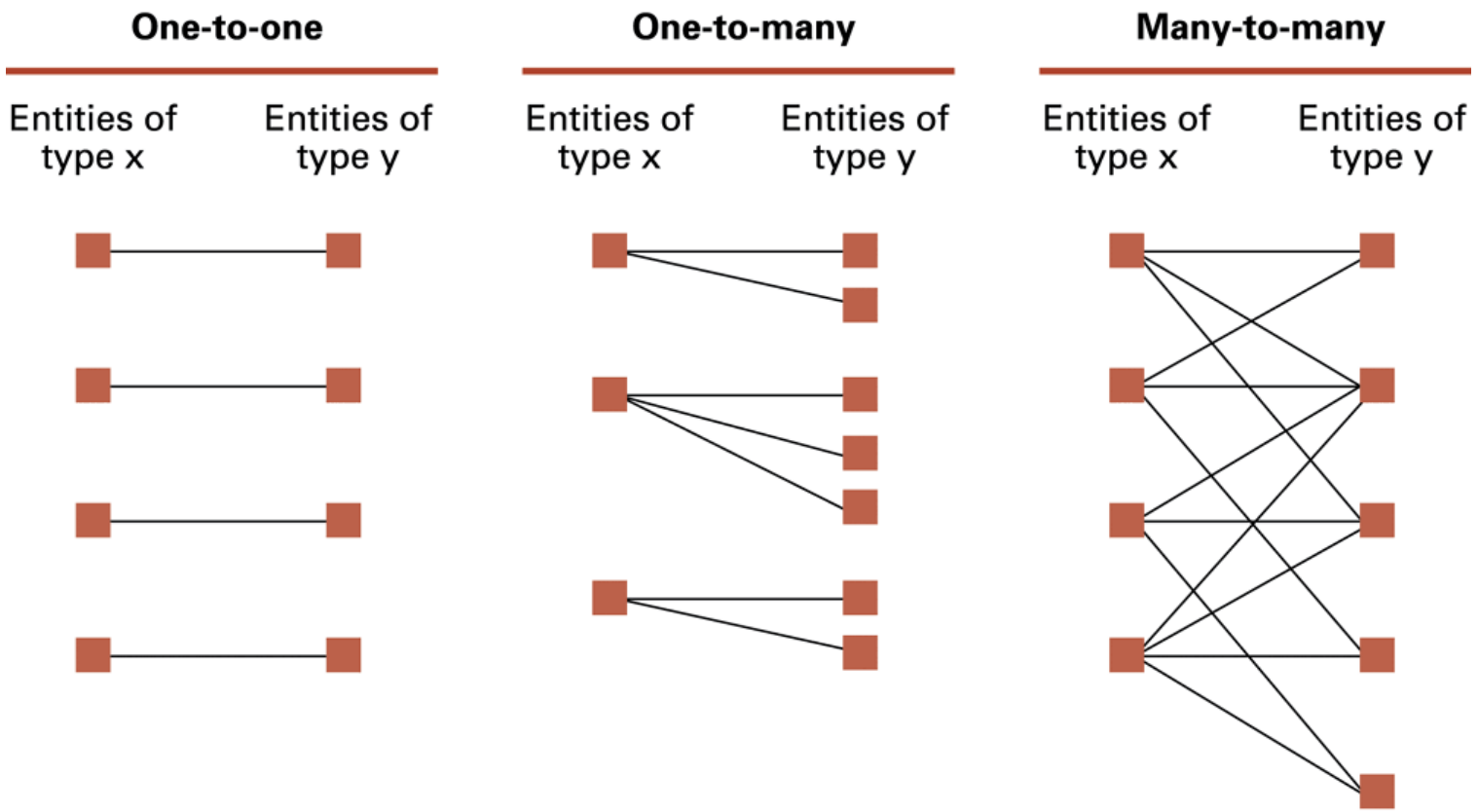
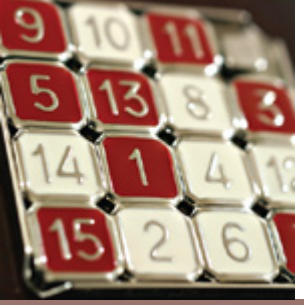


Figure 7.10 One-to-one, one-to-many, and many-to-many relationships between entities of types X and Y





Unified Modeling Language

- Use Case Diagram
  - Use cases
  - Actors
- Class Diagram

Figure 7.11 A simple use case diagram

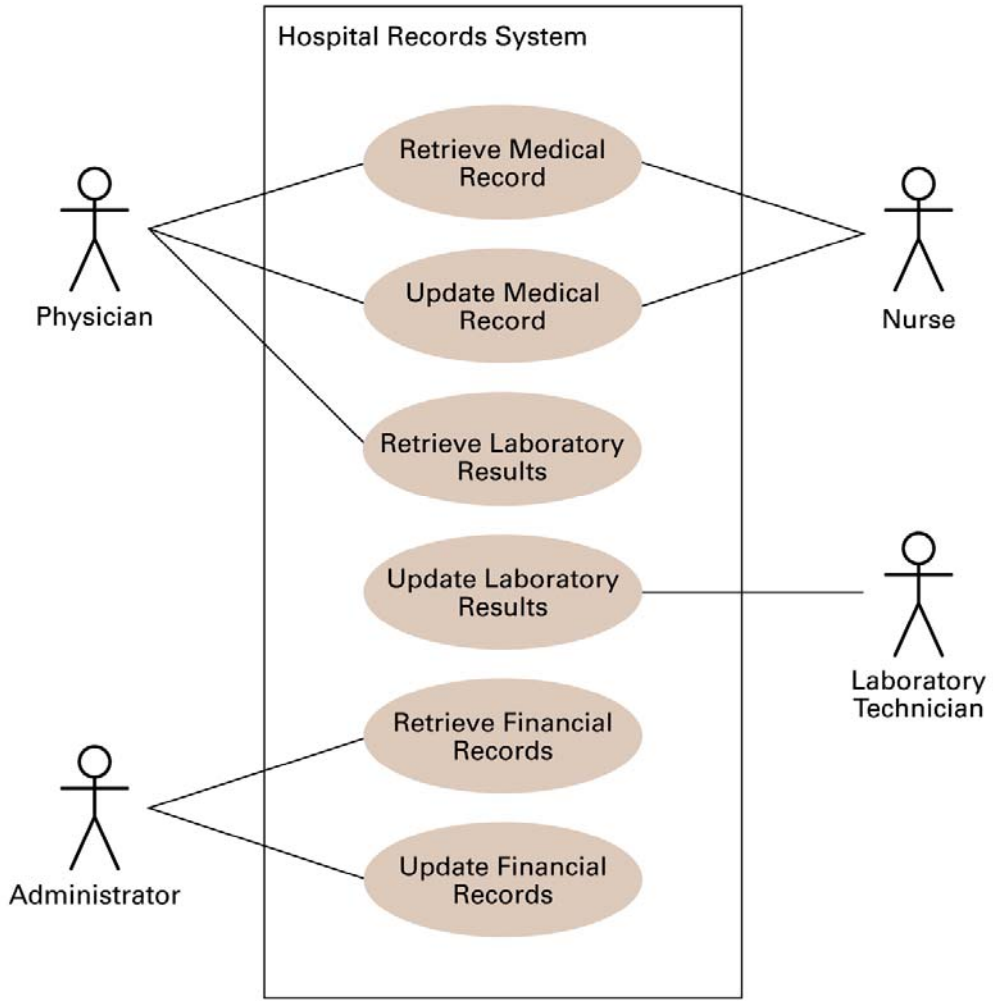


Figure 7.12 A simple class diagram

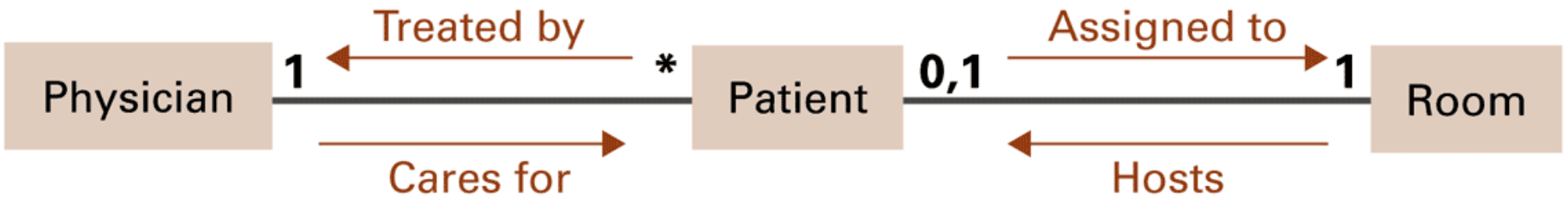
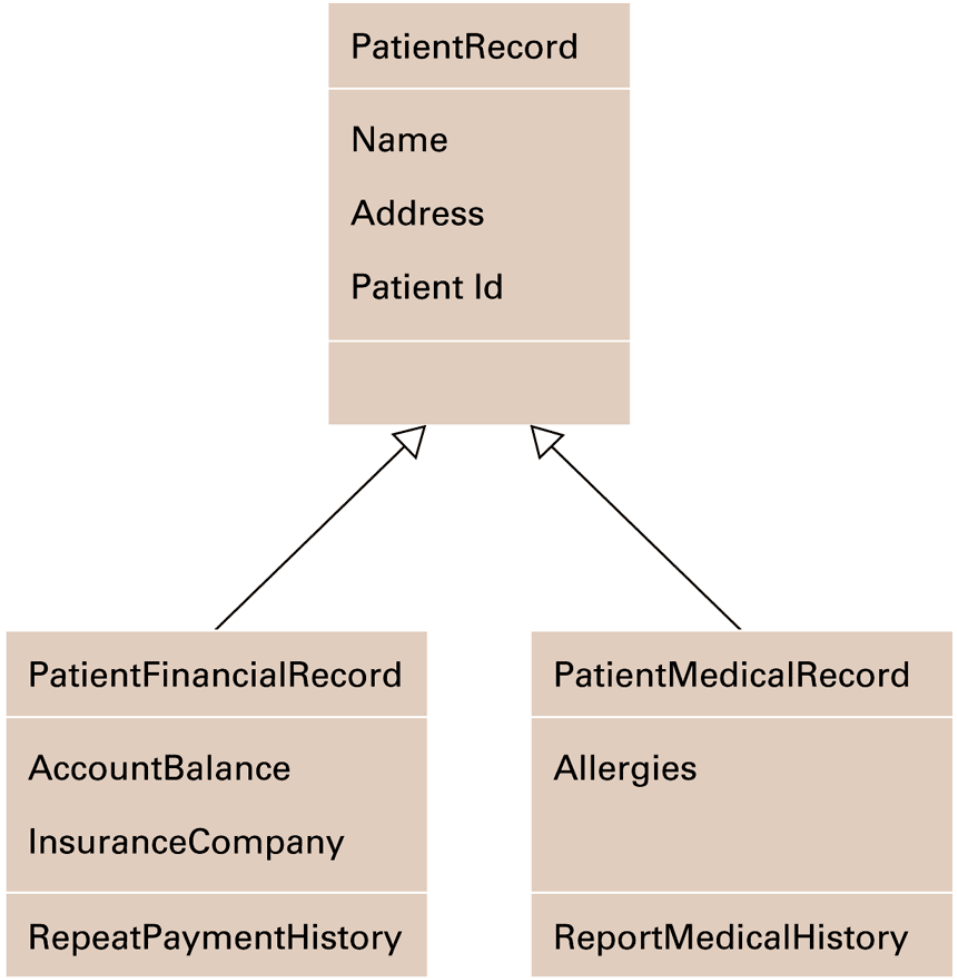
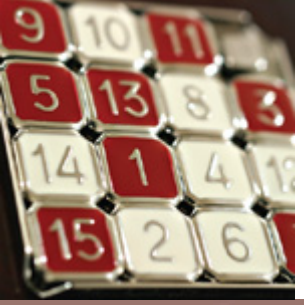


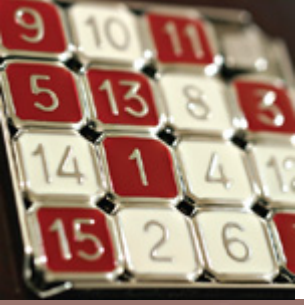
Figure 7.13 A class diagram depicting generalizations





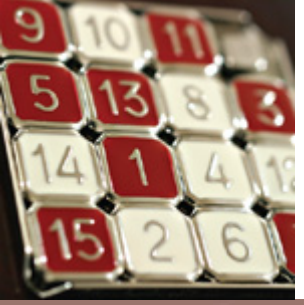
Structured Walkthoughs

- “Theatrical” experiment
- Class-responsibility-collaboration cards



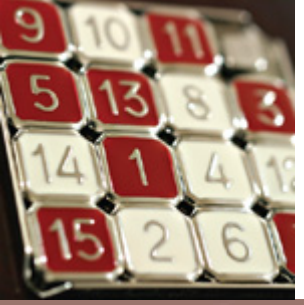
Design Patterns

- Well designed “templates” for solving recurring problems
- Examples:
  - Adapter pattern: Used to adapter a module’s interface to current needs
  - Decorator pattern: Used to control the complexity involved when many different combinations of the same activities are required
- Inspired by the work of Christopher Alexander in architecture



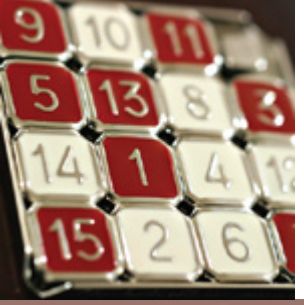
Software Testing Strategies

- Glass-box testing
  - Pareto principle
  - Basis path testing
- Black-box testing
  - Boundary value analysis
  - Redundancy testing
  - Beta testing



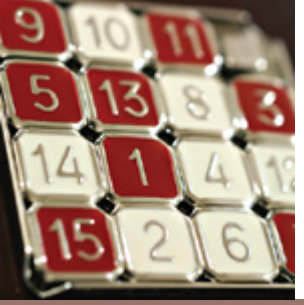
Documentation

- User Documentation
  - Printed book for all customers
  - On-line help modules
- System Documentation
  - Source code
  - Design documents
- Technical Documentation
  - For installing, customizing, updating, etc.



Software Ownership

- Copyright
  - The “substantial similarity” test
  - Filtration criteria: what is not copyrightable
    - Features covered by standards
    - Characteristics dictated by software purpose
    - Components in the public domain
  - The “look and feel” argument



Software Ownership (continued)

- Patents
  - "Natural laws" are traditionally not patentable
- Trade secrets
  - Non-disclosure agreements are legally enforceable