Design Patterns (1)

31 March 2011
CMPT166
Sean Ho
Trinity Western University

See also:
Vince Huston,
JavaCamp,
OODesign.com

TRINITY WESTERN UNIVERSITY

Outline for today

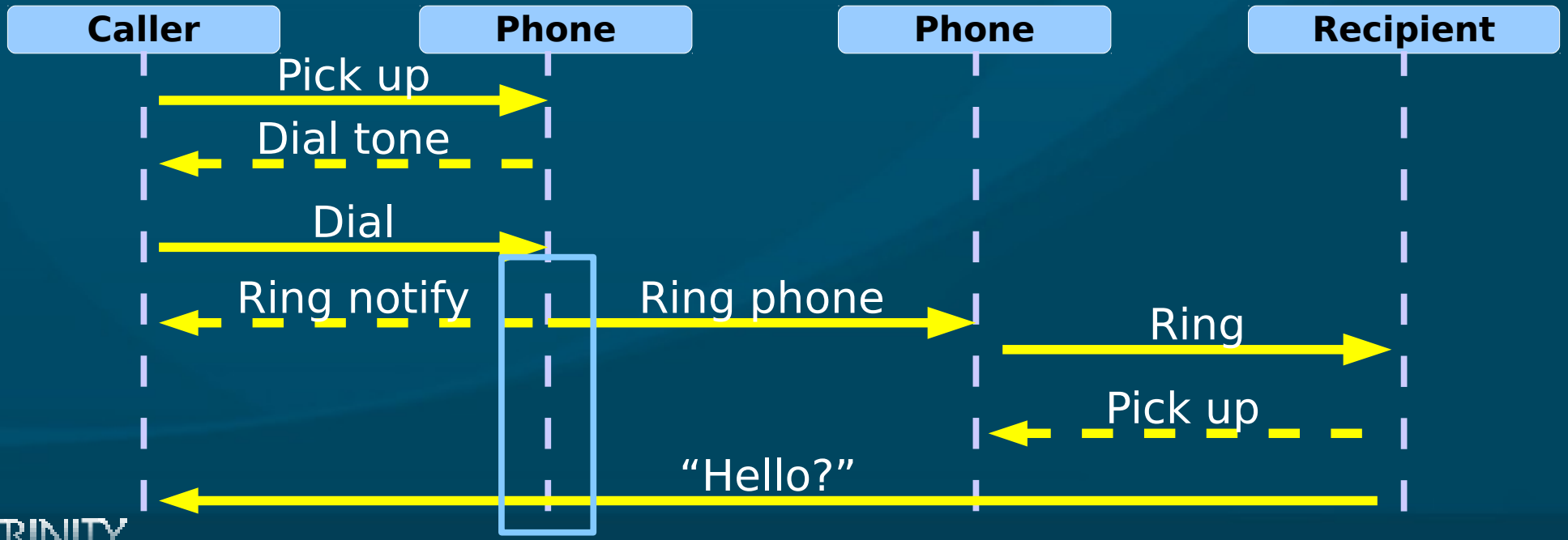
- UML Sequence diagrams
  - Lifecycles, if/else blocks, async messages
- Design Patterns
  - Creational patterns
    - Factory Method
    - Abstract Factory
    - Builder
    - Prototype
    - Singleton

UML and reusable designs

- Diagrams for
  - Use-case scenarios
  - Component / CRC diagrams
  - Class diagram
  - Sequence diagram
  - More: state diagrams, activity diagrams, etc.
- Christopher Alexander, “Notes on the Synthesis of Form”, Harvard University Press, 1964
- Ref: Gamma, Helm, Johnson, Vlissides, “Design Patterns: Elements of Reusable OO Software”

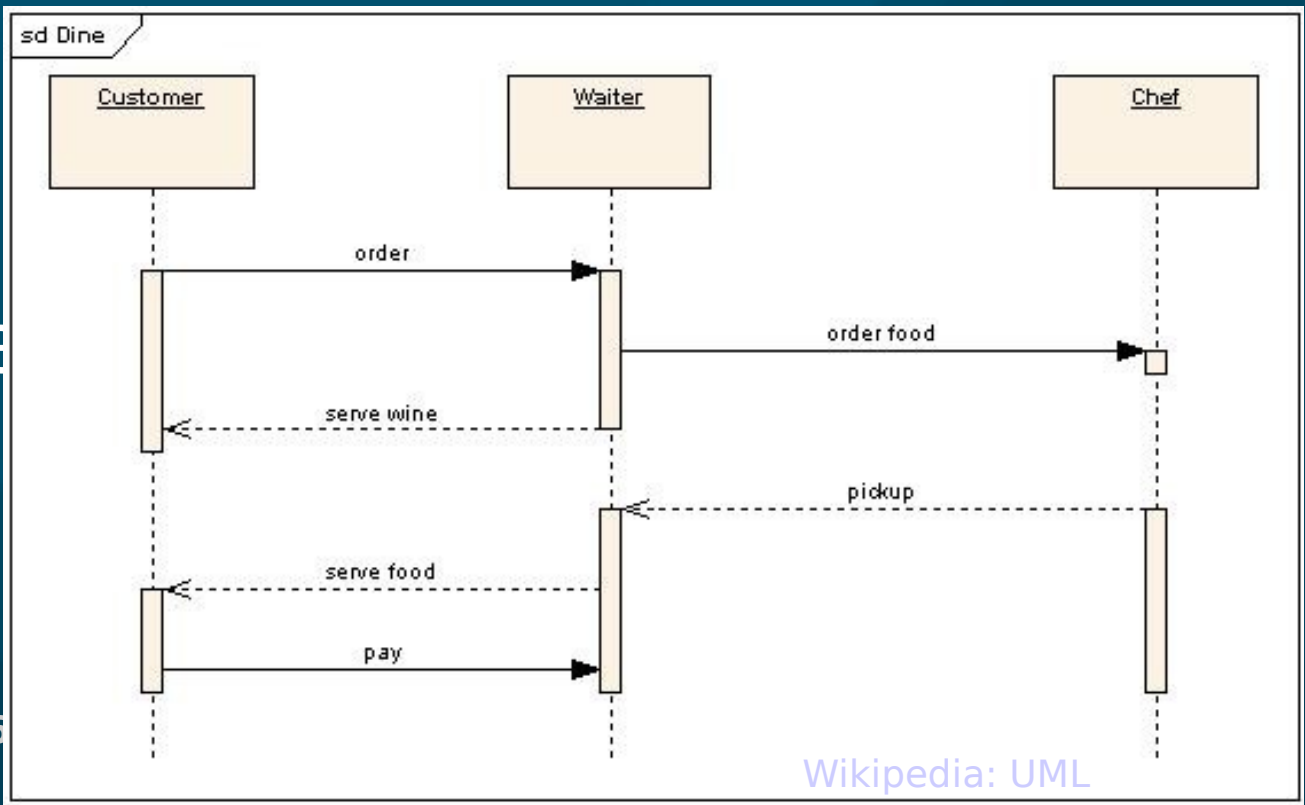
Sequence diagrams

- Describes behaviour via messages passed
  - Lower-level than use-cases
- Each object / actor / comp. has a lifeline
- Messages: arrows Activities: vertical box



Activity lifecycle

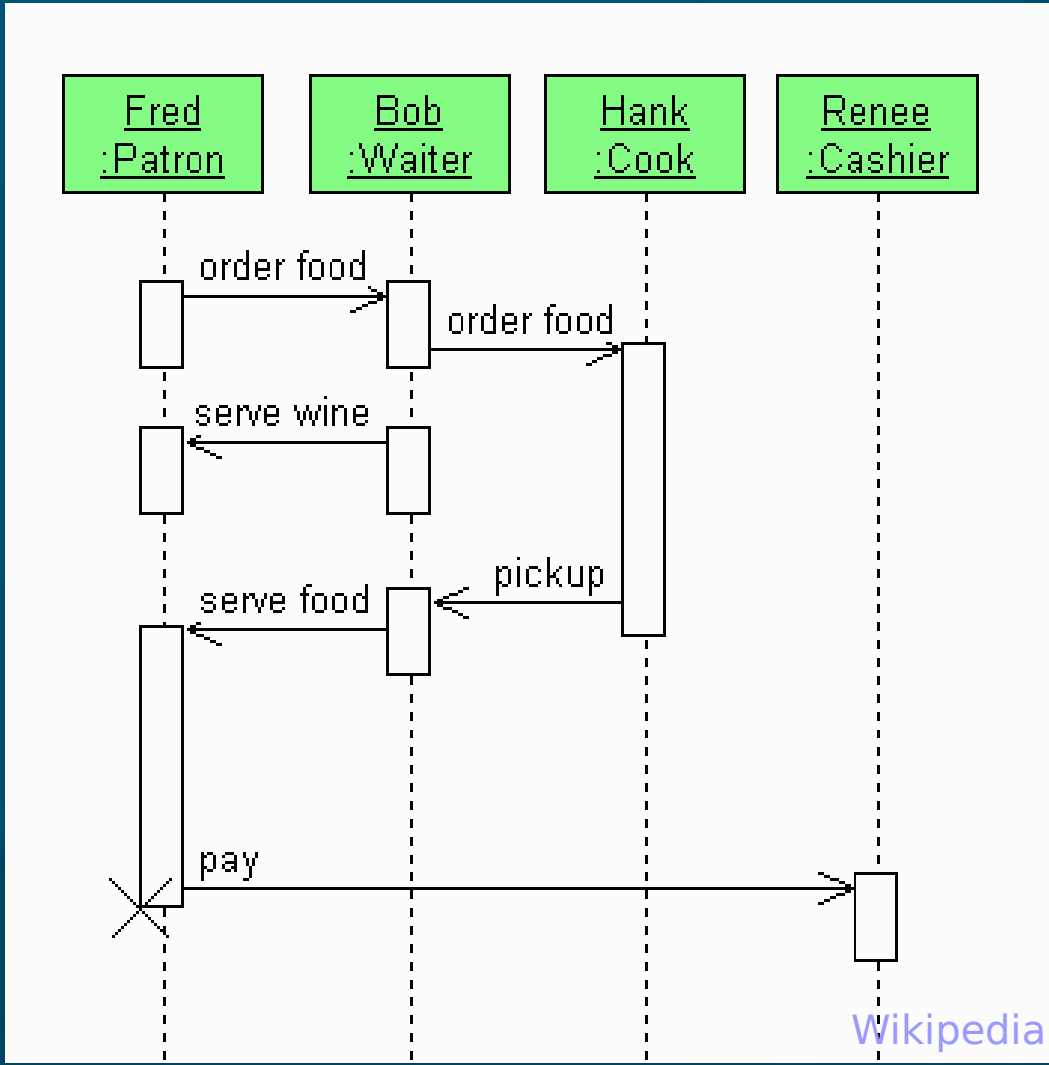
- Each vertical box represents an activity performed by the actor / object / comp.
- Vertical length is duration of the activity
- Messages may trigger activities to start
- On completion, an activity may trigger a return message (dotted)



Wikipedia: UML

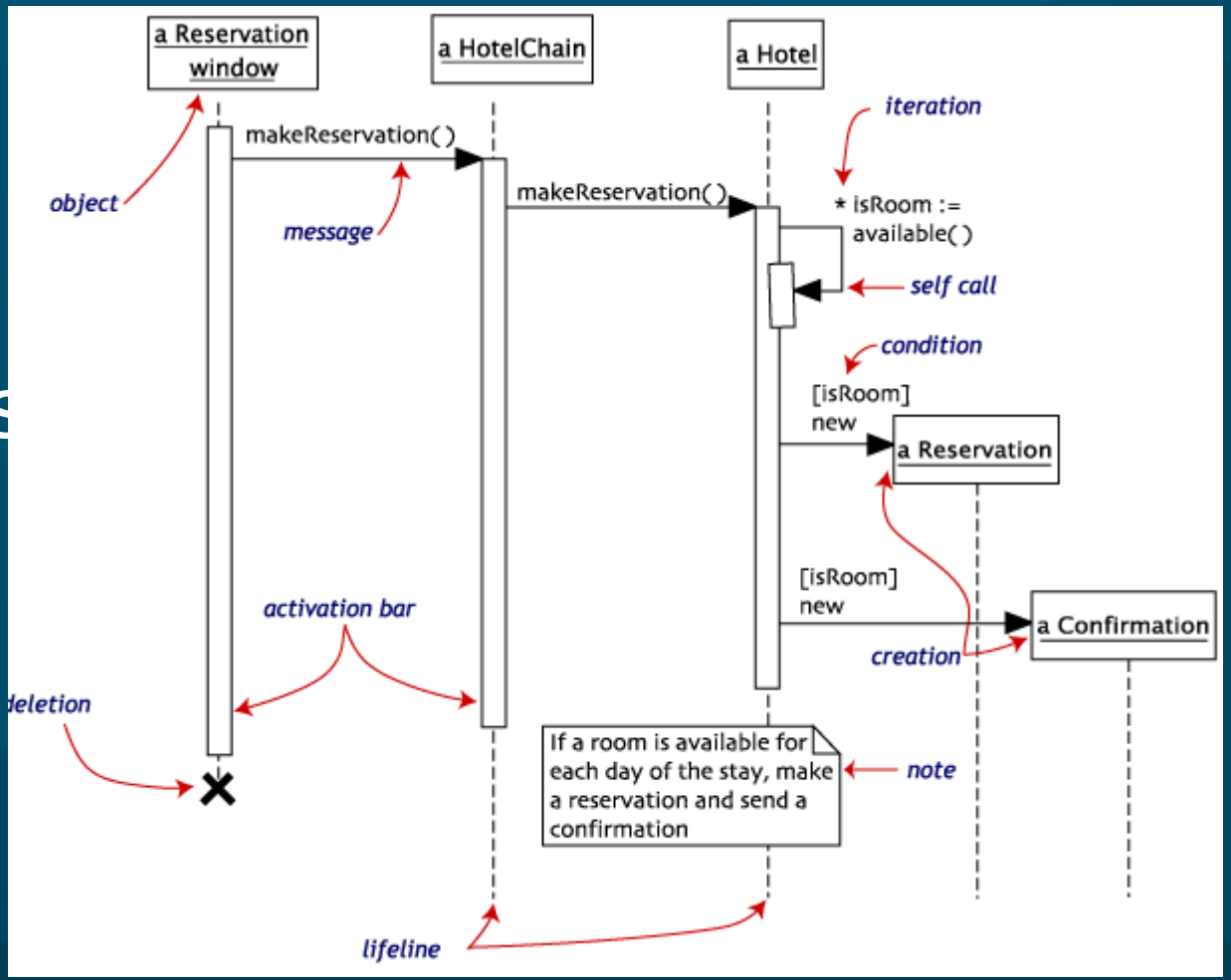
Object lifecycle

- When an object dies, an 'X' marks the end of its lifeline:



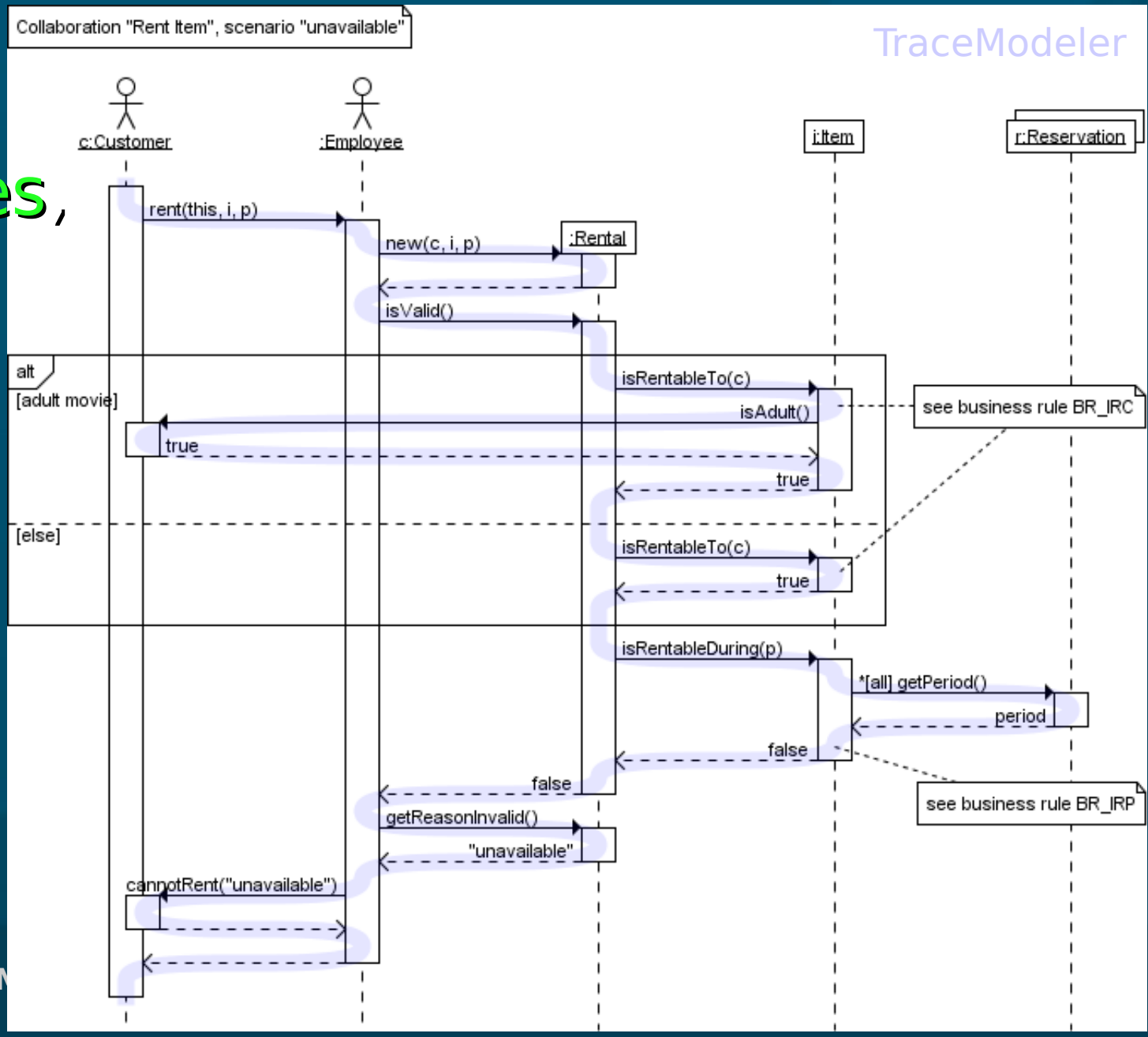
Self-called messages

- An object may send a message to itself:
  - Arrow loops back to itself
- A message may also be sent multiple times:
- Usually put loops in a shaded box



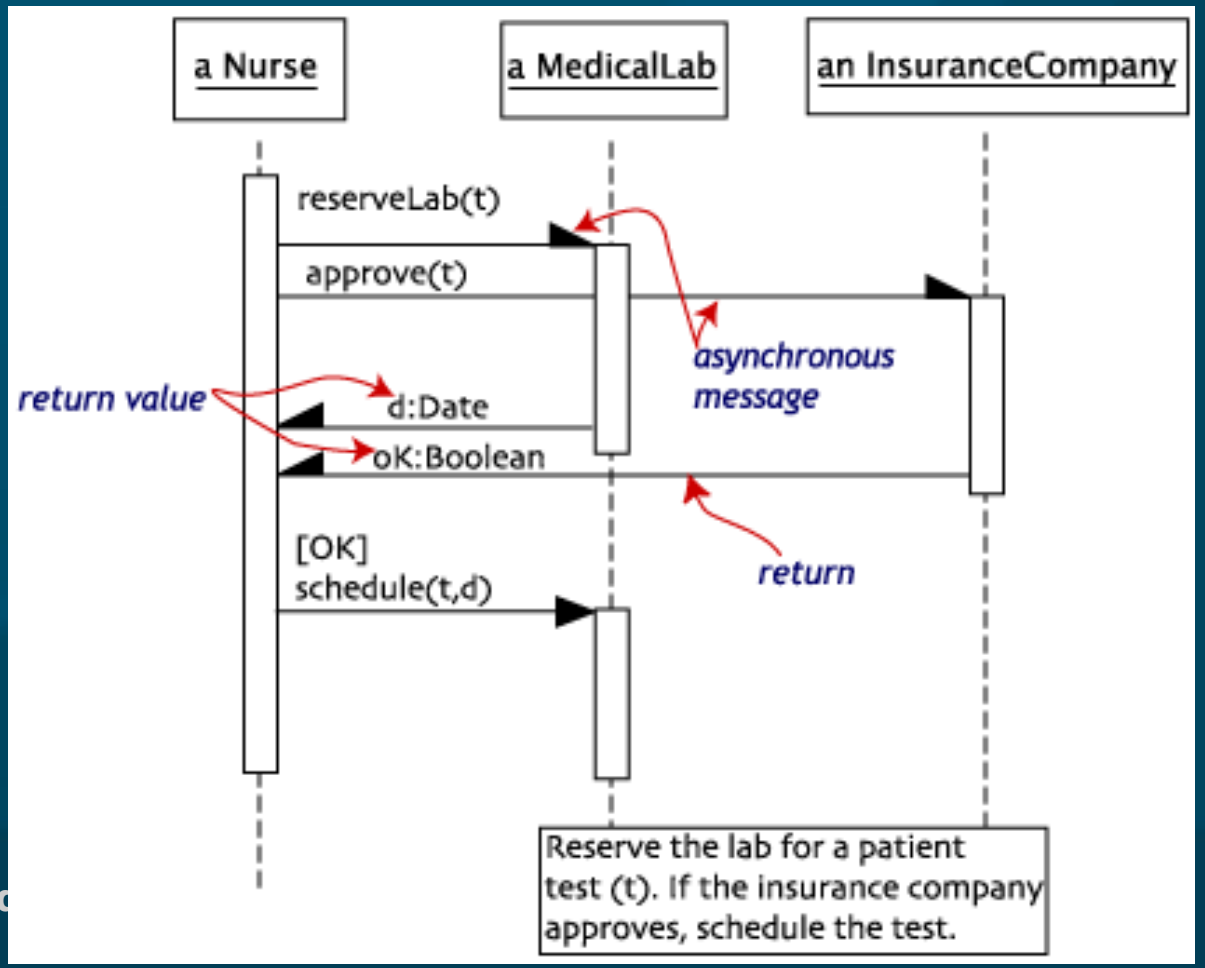
If/else clauses

- Put both cases in dotted boxes, with the if condition



Asynchronous comm.

- Synchronous messages (full arrowhead) require recipient to be listening right then
- Asynchronous messages (half arrowhead) can be read by recipient at a later time
  - Phone call vs. postal mail
  - Chat vs. email

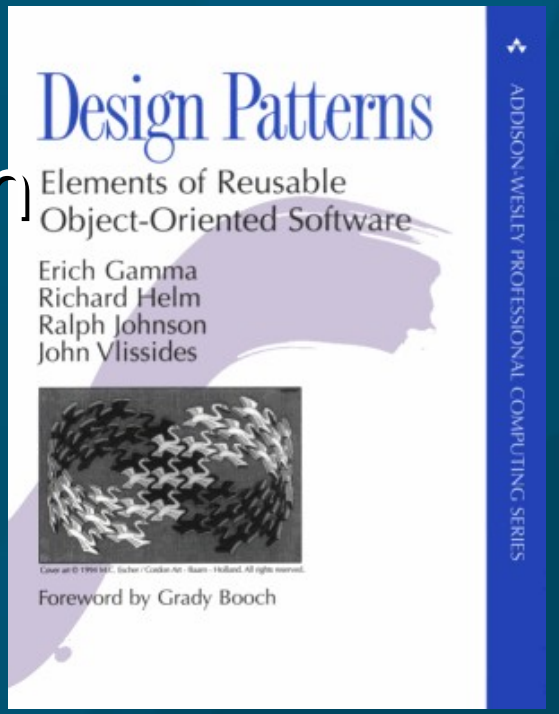


UML Software Tools

- Eclipse UML2Tools plugin (Model Dev. Tools)
  - Help → Install New… → Add …
    Paste URLs for MDT and EMF
  - Other plug-ins (M2T) can auto-generate code from your diagrams!
- Dia: free diagram editor (not just UML)
- IBM Rational Modeler
  - free version of commercial Rhapsody prod.
  - Complex but complete UML modelling tool
- Or draw your own by hand!

Design patterns

- A pattern is a named abstraction
  - from a recurring concrete form
  - that expresses the essence of
  - a proven general solution
- A pattern has three parts:
  - some recurring problem from the real world
  - the context of the problem (when to solve it)
  - the rule telling us how to solve it
- Describes a class of problems and how to solve them

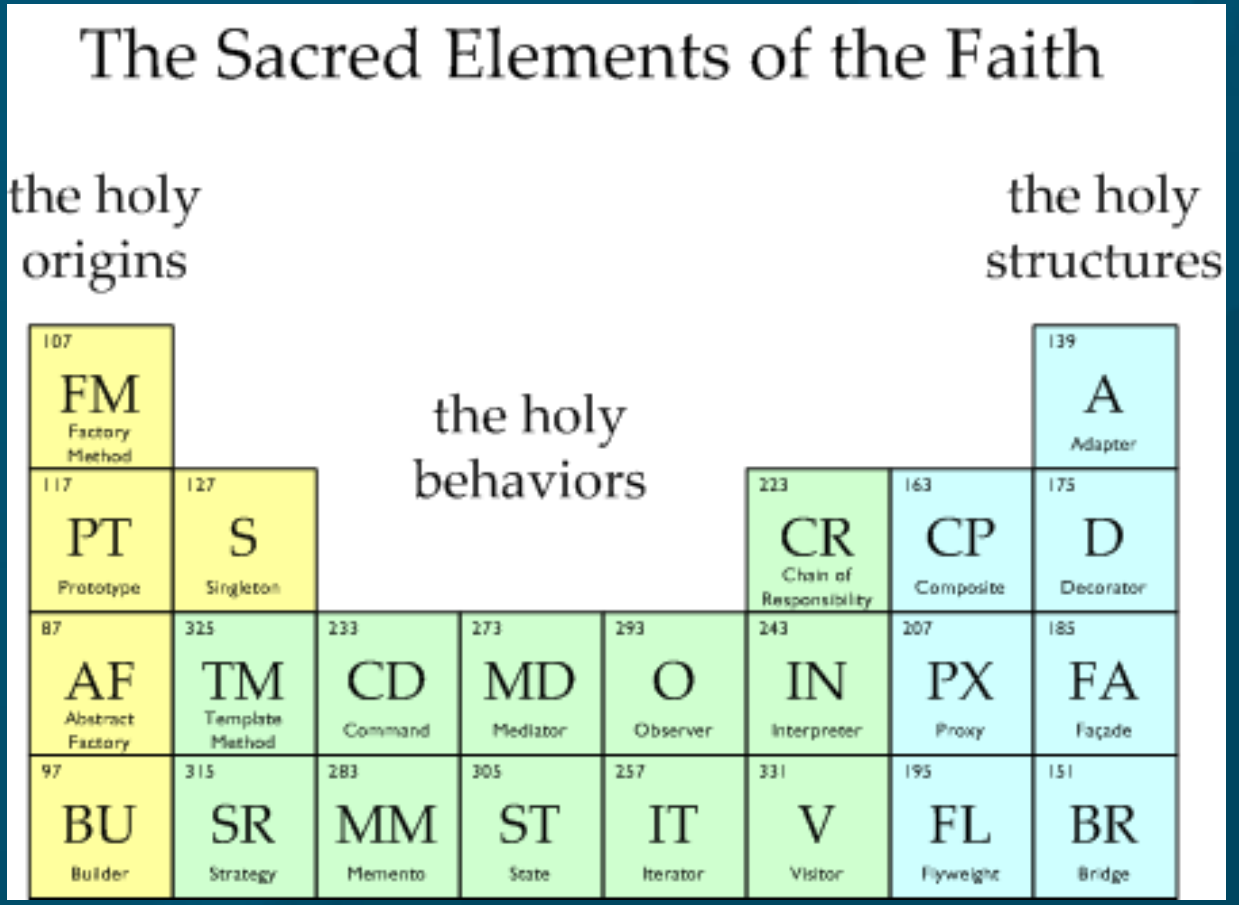


Classes of patterns

- Conceptual/architectural
  - Structural organization of software systems
  - Set of predefined components
  - Relationships between components
- Design
  - How to refine each component
  - Commonly recurring structure of components
- Programming idiom
  - How to code a particular component feature

Classes of patterns (GoF)

- Creational
  - Interfaces to generate new objects
- Structural
  - How to organize a large system in components
- Behavioural
  - How components interact with each other to accomplish a common goal



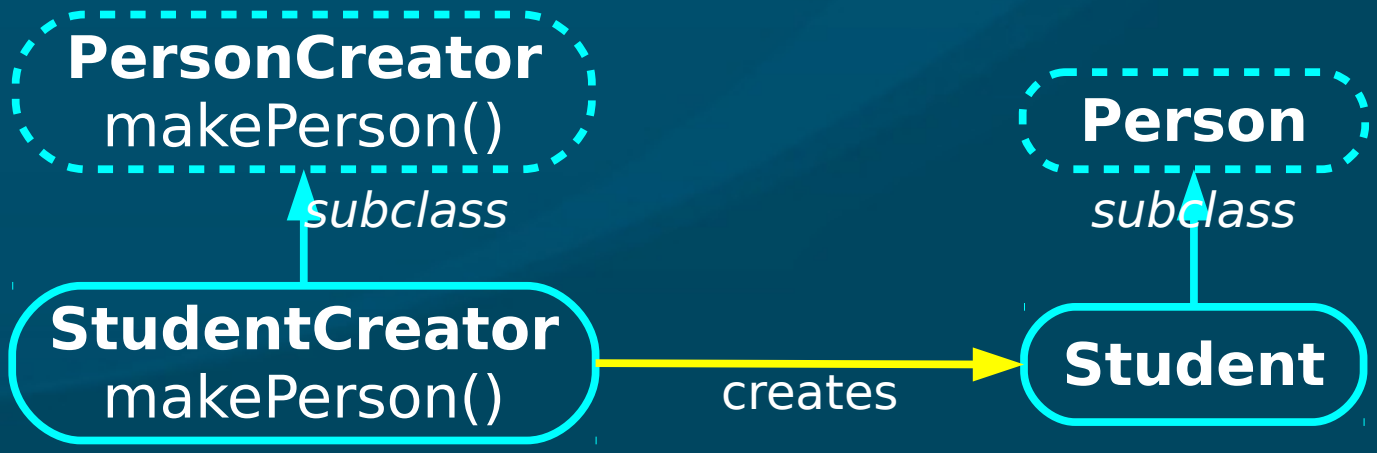
Creational patterns

- Factory Method: create a variety of objects
- Abstract Factory: group of related obj factories
- Builder: delegate creation of components
- Prototype: clone a template object
- Singleton: enforce having only one instance

Creational: factory method

JavaCamp, Wikipedia

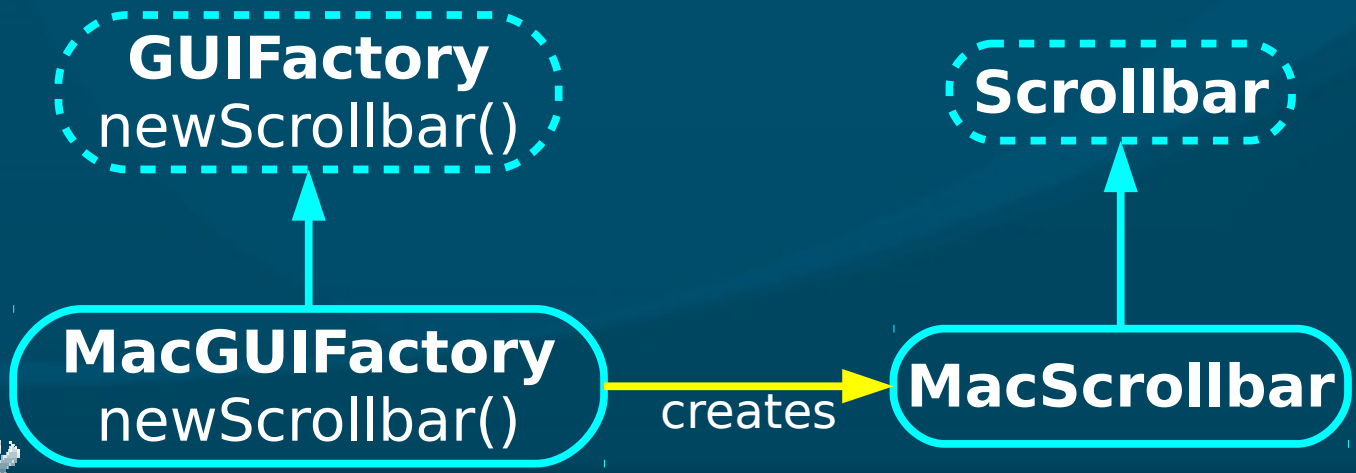
- An interface to create an object, but without specifying which subclass
- Analogy: plastic injection-mould determines shape of output
- e.g., need to create a new Person; don't know in advance if it's Student, Staff, or Faculty

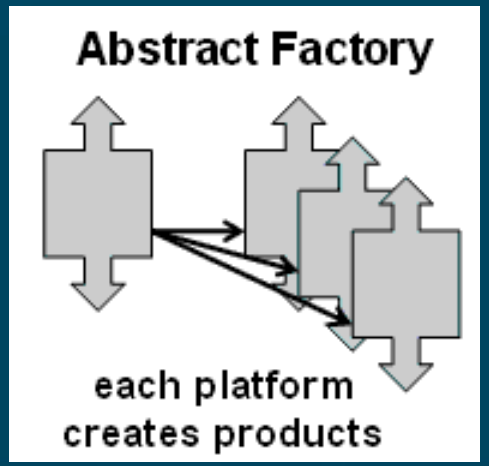


Creational: abstract factory

JavaCamp, Wikipedia

- Family of similar factories
  - Client code doesn't know/care which concrete factory is used
  - May use a collection of factory methods
- Analogy: press to stamp out auto parts
- e.g., adaptable look-and-feel of GUI widgets

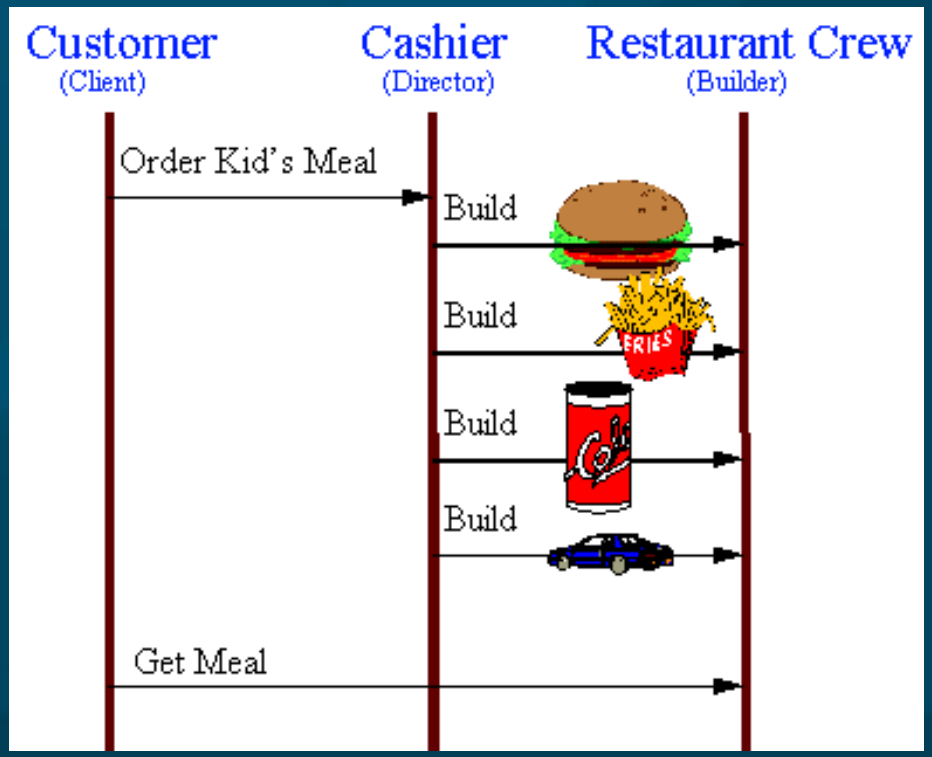




Creational: builder

JavaCamp, Wikipedia

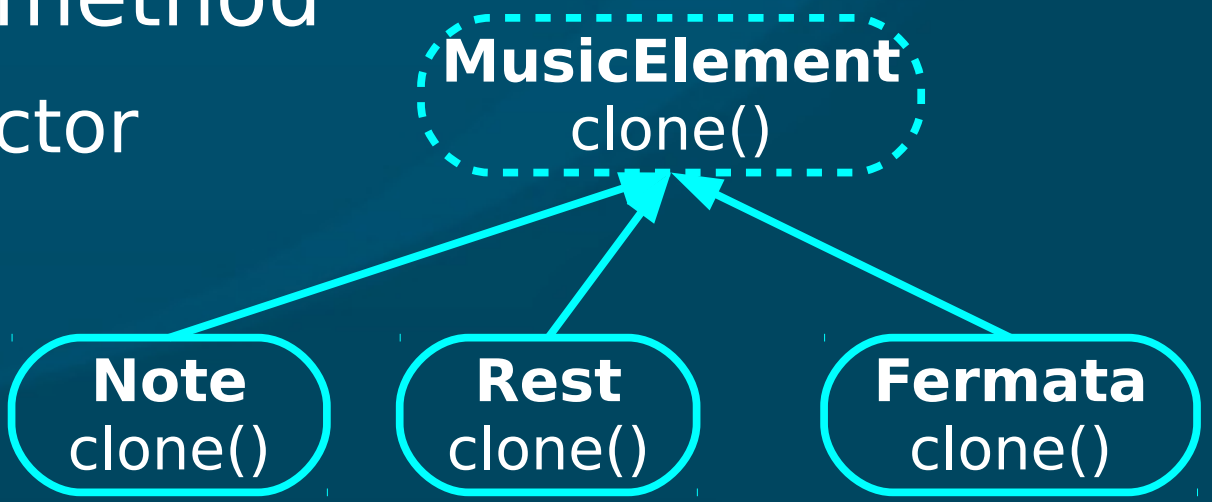
- Separate construction of a complex object from its representation
  - Analogy: assembling fast food kids' meals
- Director class parses the request and representation
- Hierarchy of Builders make the objects



Creational: prototype

JavaCamp, Wikipedia

- Create new objects by copying a prototype
  - Analogy: biological cell division
  - e.g., sheet-music editor: copy and paste notes
    - Staves are objects; each note is an object
- Design each object so it knows how to copy itself: clone() method
  - Copy constructor



Creational: singleton

JavaCamp, Wikipedia

- Ensure a class only has one instance, and provide a global point of access to it
  - Analogy: only one Prime Minister
- Can implement using private constructor
  - Provide a static get method for the singleton

```
public class PrimeMinister {
  private PrimeMinister thePM;
  private PrimeMinister() { /* create new PM */ };
  public static getPM() {
    if (!thePM) thePM = new PrimeMinister();
    return thePM; }
```