INF 111 / CSE 121: Software Tools and Methods

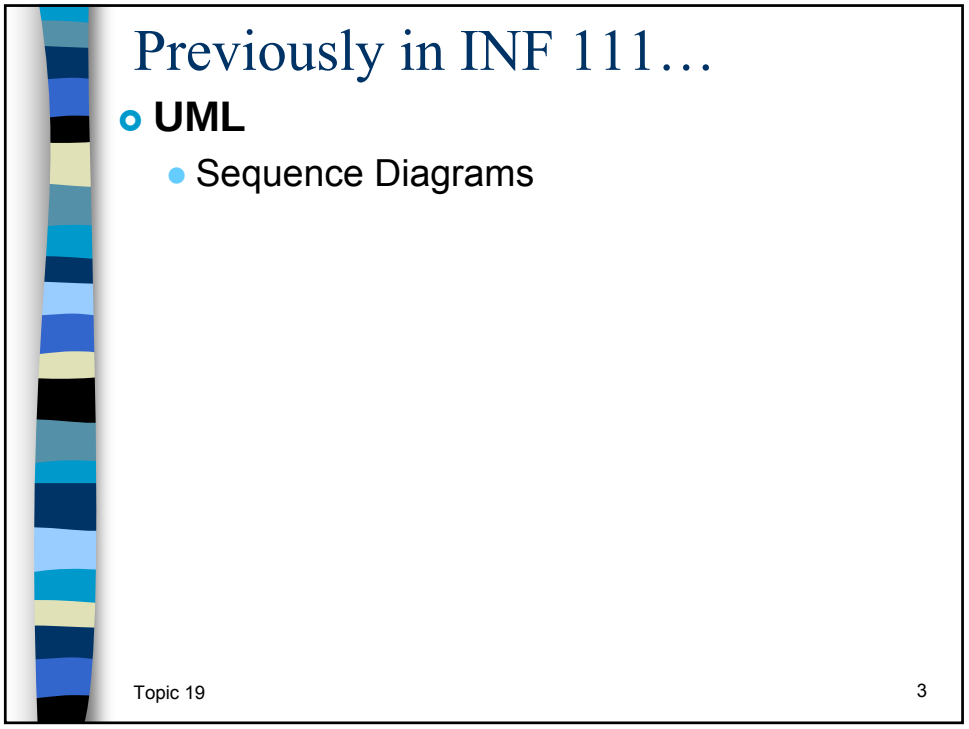
Lecture Notes for Fall Quarter, 2007

Michele Rousseau

Set 19

Announcements

- **Homework Due 11/21 @ 3p**
- **Quiz #3 will be available in the distribution center this afternoon**
- **Emailing the TA…**


Previously in INF 111…
UML
Sequence Diagrams
Topic 19
3


Review
Topic 19
4

Today's Lecture

- **Review Quiz #3 Answers**
- **UML**
  - Package Diagrams
  - State Transition Diagrams
  - Activity Diagrams

Package Diagrams

- **What is a *package*?**
  - A construct that enables you to organize model elements into groups
  - Classes or use cases
- **A *package diagram* is a diagram with packages and their dependencies**

Why use package diagrams?

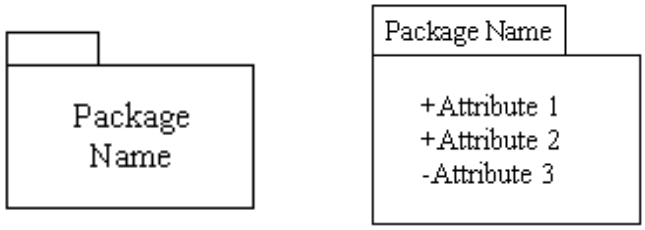
- **Increases the level of abstraction for complex diagrams**
- **Depict a high-level overview of your requirements or architecture/design**
  - A collection of use case or class diagrams
- **To logically modularize a complex diagram**
- **To organize Java source code**

Not limited to class and use case diagrams

Because diagrams can get messy

Package Diagrams: Notation

- **Represented as tabbed folders**

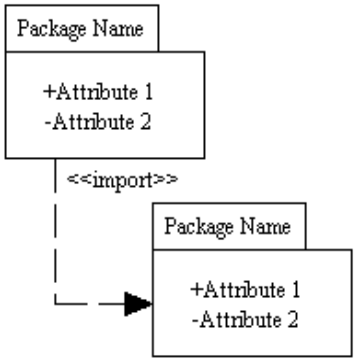


- **Can use visibility markers**
  + Public
  - Private
  # Protected

Relationships

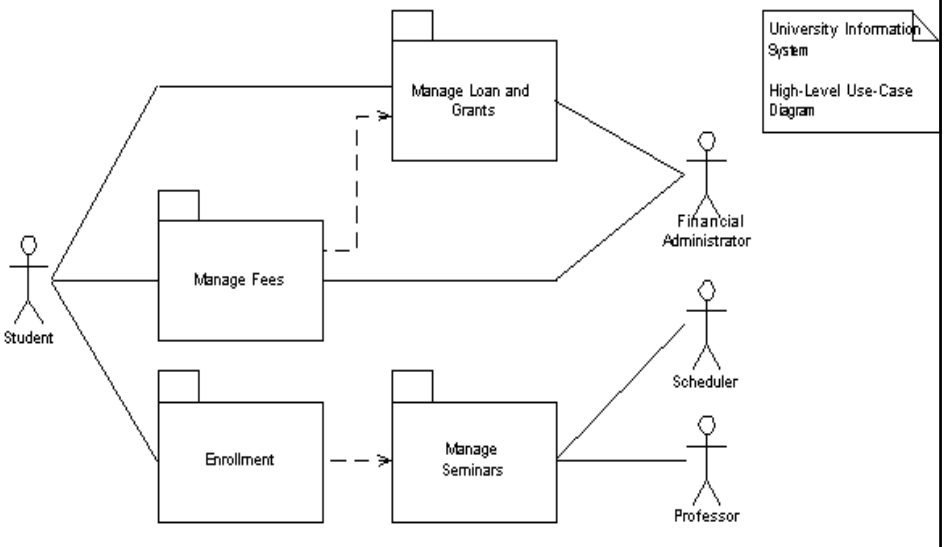
Two Types

- **Dependencies**
  - Changes to one package affects another
  - Import is one type that grants access
  - Represented by a dashed arrow

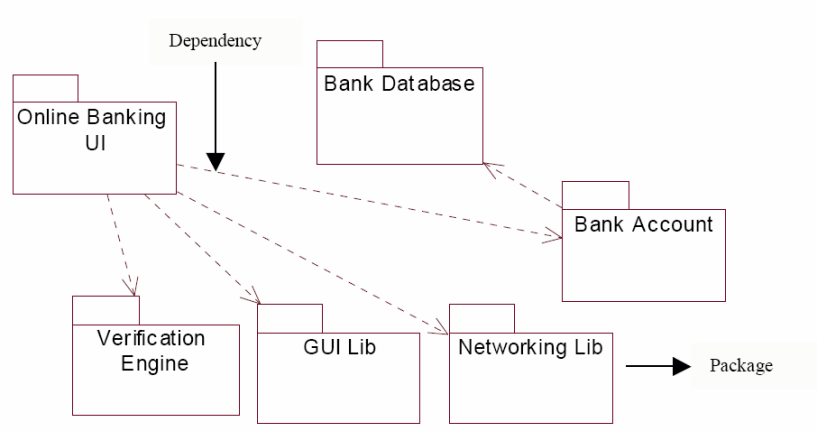


- **Generalizations**
  - Represented with an open arrow just like in previously discussed diagrams

Use Case Example



Class Diagram Example



Some Basic Tips on Packages

- **Use Simple, Descriptive Names**
- **Use when you need to Simplify Diagrams**
- **Packages Should be Cohesive**
- **Avoid Cyclic Dependencies Between Packages**

State Transition Diagrams

- State Transition Diagrams show the *dynamic behavior* of a class instance or of a whole system
- *State*: the duration of time during which an object is doing an activity.
- A ***state diagram*** is a **graph in which**
  - nodes correspond to states and
  - directed arcs correspond to transitions
  - labeled with event names.

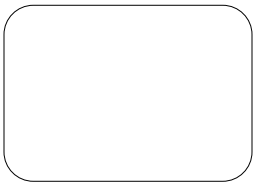
When to use :
Necessary for those objects whose behavior across many use cases needs to be understood

State Transition Diagrams

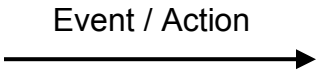
- An ***event*** occurs at a point in time and
  - transmits information from one object to another
- An ***action*** occurs in response to an event and cannot be interrupted
- An ***activity*** is an operation with certain duration that can be interrupted by another event
- A ***guard*** is a logical condition placed before a transition that returns either a true or a false.

State Transition Diagrams: Notation

- State symbol:



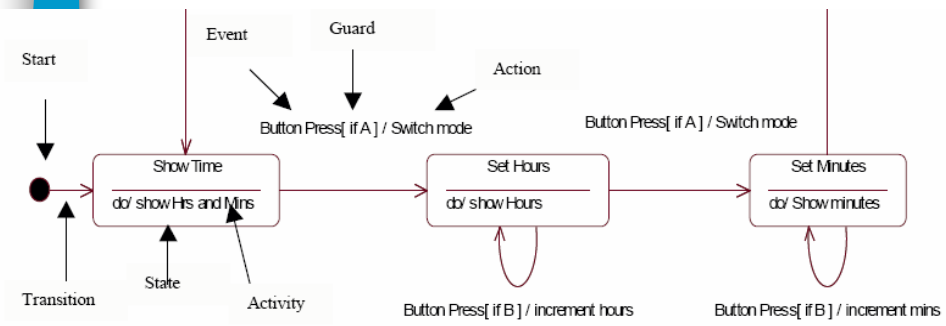
- Transition Symbol:



Example: Simple Digital Watch

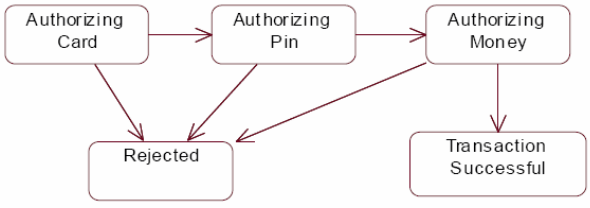
- Has a display and two buttons to set it
  - **A & B Button**
- Watch has two modes of operation
  - **display time - hours:minutes**
  - **set time – two modes**
    - **Set hours**
    - **Set minutes.**
- The "A" button is used to select modes.
  - **On each press, the mode advances in sequence:**
    - **display →set hours→set minutes→display etc.**
  - **Within the sub modes, the "B" button is used to advance the hours or minutes once each time it is pressed.**
- Buttons must be released before they can generate another event.

State Transition EX: Digital Watch



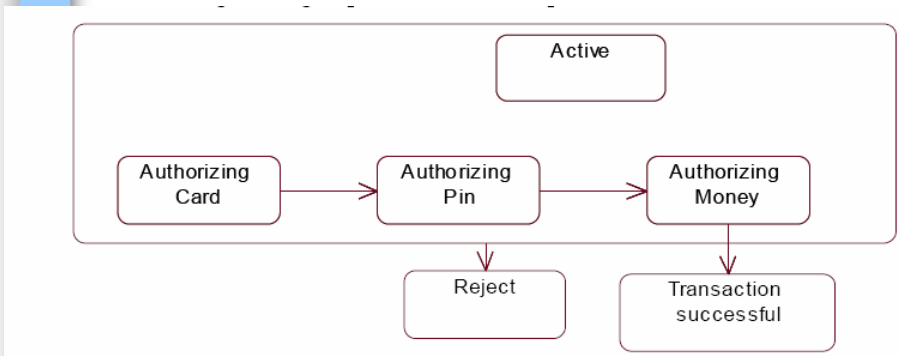
Super States

- **Simplify complex diagrams by grouping states**
- **Without superstate**
- **Rejected can arise from all 3 authorizing states**



With Superstate

- **All the authorizing states grouped as "active" superstate**

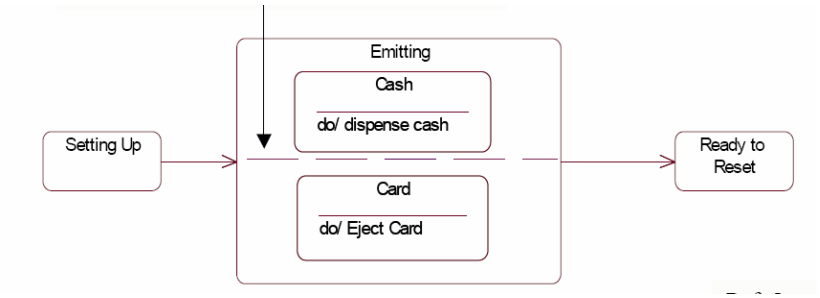


Concurrent State Diagrams

- To represent objects in simultaneously occurring states when performing an activity.
- Concurrent state diagrams are useful in modeling objects that have independent behaviors doing the same activity
- Represented by a dashed line in a state

Be careful not to make a single state too complex

Concurrent State Example: ATM



Activity Diagrams

- **A flow chart with support for parallel behavior**
- **Branches and Merges model the conditional behavior**
- **Branch: has a single incoming transition multiple, conditional, outgoing transitions**
- **Merge: where conditional behavior terminates**

Each branch has a corresponding merge

- **Represented as a Diamond**

Activity Diagram (2)

- **Forks and Joins model parallel behavior**
- **Fork: has a single incoming transition and multiple outgoing transitions (exhibiting parallel behavior)**
- **Join: synchronizes the parallel behavior**
  - All parallel behaviors complete at the join

Each Fork has a corresponding Join

- **Represented as a thick line**
- **Conditional Thread: A condition on the thread originating from the fork to create an exception for the join rule.**

Activity Diagram (3)

- **Synch State: synchronizes different activities so they make a transition to the next activity at the same time**
- **When to use Activity Diagrams?**
  - When modeling parallel behavior or
  - Documenting the logic of a business process

Activity Diagram:

ATM Ex.