Slide 1

Administrivia

- Reminder: Next phase of project (requirements analysis plus initial design) due Wednesday after break. Also individual evaluations.
- Remember that Eclipse on lab machines has a plug-in for drawing UML diagrams. Might be nice to use for (re)drawing your use-case diagrams.
- Course "useful links" has preliminary links to online sources of more information about today's topics. More coming soon.

Slide 2

Model-View-Controller Design Pattern

- (What's a "design pattern"? "High-quality solution to a frequently occurring problem in some domain.")
- "Model-View-Controller" (MVC) frequently recommended as a way of structuring an application with a GUI. Idea is to separate overall design into three pieces . . .

Slide 3

Model-View-Controller Design Pattern, Continued

- Model — underlying data, “business logic”, etc.

  For ATM example we talked about earlier in the semester, this would be whatever represents the actual bank accounts — classes for individual accounts, with methods to validate user, add money, remove money, query, etc., etc.

- View — view of underlying model.

  For ATM example, this would be look/feel/behavior of GUI. Notice that you could have different ones . . .

- Controller — something that couples model and view, i.e., translates interactions with the view into actions to be performed on the model.

  For Java program, “listener” methods and main program.

Slide 4

UML Class Diagrams

- These should be more or less familiar to you from PAD II, but you may have used them only to represent class/subclass and interface representations.
- Much other functionality . . .

Slide 5

UML Class Diagrams — Class Symbol

- Abbreviated form — box with class name.
- Full form — box with three sections:
  - Class name.
  - Attributes — each with name and type.
  - Operations — each with “formal signature” consisting of name and input/output arguments (each with name and type).

  Attributes and operations can specify visibility (public, protected, or private).
- Much additional notation (for class attributes/operations, abstract operations, etc.). One useful for Java — label interfaces with <<interface>> above class name.

Slide 6

UML Class Diagrams — Generalization Construct

- Idea here is to represent inheritance relationships.
- Draw as arrow with open head (points from subclass to superclass).
- Dashed line represents implementing an interface.

UML Class Diagrams — Association Construct

Slide 7

- Idea here is to represent some type of relationship between instances of classes.
- Draw as line from one class to another, with label naming the relationship. Can also have text at each end that shows role played by each “end” of association, and something showing multiplicity (zero, one, or many).

UML Class Diagrams — Composition

Slide 8

- One type of association is “composition” – composite object that contains other objects (“components”). Typically composite object contains different components and doesn’t exist without them; components don’t exist outside it.
- Draw as line from one class to another, with solid diamond at composite-object end.

UML Class Diagrams — Aggregation

- Another type of association is “aggregation” — e.g., forest is aggregation of trees. “Aggregate object” can have one or more “constituent objects.”
- Draw as line from one class to another, with open diamond at aggregate-object end. Show multiplicity on both sides.

Slide 9

Minute Essay

- How much of what we talked about today is familiar to you?
  - Model-view-controller?
  - Class diagrams for representing attributes and operations.
  - Class diagrams for representing inheritance relationships.
  - Class diagrams for representing associations.

Slide 10