Designing Classes

October 11, 2007

Design Principles

- High cohesion - everything in a class is related
- Low coupling - a class has limited dependencies on other classes
- Abstraction - a class can be used easily without knowing how it is implemented
- Encapsulation - a class hides design decisions, making them easy to change

Thursday, October 11, 2007
Identifying Classes

- Examine problem domain. Model real world objects with classes:
  - Physical things: dice, thermometer
  - Organizations: team, college
  - People: player, student, customer

Identifying Classes (2)

- Examine solution domain
  - Agents: objects specialized in a particular activity, like StringTokenizer, BufferedReader
  - Events: mouse actions, window actions
  - External software/hardware: files, sensors, databases
Unified Modeling Language

Collection of graphical notations used to document designs

Class diagrams - show classes and static connections between them

Sequence diagrams - show dynamic behavior of objects carrying out an action

State diagrams - show "lifecycle" of an object
This Sounds Silly But...

- Start identifying classes by carefully reading an English description of the system
  - Nouns -> classes, or maybe attributes of classes
  - Verbs -> methods
Questions

- When is a noun a class, when is it an attribute and when is it neither?
- If a verb is a method, which class does it belong to?

Identify Relationships between Classes

- Association: “has”
  - Example: a car has an engine
  - Will turn into an instance variable
- Dependency: “uses”
  - Example: an event handler depends on the event
  - Often parameters to methods
- Inheritance / implements