Midterm Exam

- Monday, October 15
- Closed book, closed notes, clean desk
- Textbook: Chapters 1 and 2
- Java Lectures
- 25% of your final grade
- I recommend using a pencil (and eraser)
- I will bring extra paper and stapler, in case they are needed.

Exam Format

- 100 points total
  - Multiple choice questions
  - Drawing UML diagrams
  - Writing programs/functions/code
  - Tracing code (what is the output)

- Each question will indicate how many points it is worth (out of 100)

Ch 1: Introduction:
Object-oriented analysis, design, implementation

- Object-oriented analysis: finding+describing domain objects
  - concepts
- Object-oriented design: design software objects to fulfill system requirements
  - class diagram
- Object-oriented programming/implementation
  - Java classes

- The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, (as well as for business modeling)
Ch 1: Introduction: Object-oriented software development

- Requirements elicitation
  - Define purpose of system in terms of actors and use cases
- Analysis
  - Produce a system model that is correct, complete, consistent, etc.
- System design
  - Define design goals, decompose system into subparts, deployment diagram
- Object design
  - Produce detailed object model, with solution domain objects
- Implementation
  - Developers translate the solution domain model into source code.
- Testing: find differences between system and the models

Ch 2: Modeling with UML: Modeling concepts

- System: organized set of communicating parts
- Model: not a diagram!
  - Abstraction of a system: focuses on interesting aspects, ignores irrelevant details
  - Separate model for each problem
- System model: set of all models built during development
- Three models of a software system:
  - Functional Model: functionality from user's point of view (use case diagrams)
  - Object Model: structure of the system (class diagrams)
  - Dynamic Model: behavior of the system (sequence diagrams, state diagrams, activity diagrams)

Ch 2: Modeling with UML: UML diagrams

- View: depicts selected aspects of a given model
- Notations: graphical or textual rules for representing views
  - A UML class diagram is a graphical view of the object model
- Class: abstraction that captures structure and behavior
- Inheritance: refined class inherits from base class, adds more
- Object: instance of a class
- Abstract class: superclass with no instances
- Application domain: all aspects of customer's "problem"
  - Object-oriented analysis: models this domain
- Solution domain: modeling space of all possible solutions
  - Object-oriented design: models this domain
- Use Case Diagrams
  - Actor, Use case (textual descriptions, scenarios)
  - Relationships: communication, inclusion, extension, inheritance
- Class Diagrams
  - Classes, attributes, operations, objects, associations
  - Boxes with three compartments, lines are links/associations
  - Unidirectional, bidirectional associations
  - Roles, multiplicity
  - Aggregation, composition
  - Qualification
  - Inheritance
Ch 2: Modeling with UML: UML diagrams

• Interaction Diagrams
  ✦ Sequence diagrams (and communication/collaboration diagrams)
  ✦ Objects along top with timelines, time goes down,
  ✦ Labels on arrows indicate messages from one object to another (must be methods on the receiving object)

• Activity Diagrams
  ✦ Sequence and coordination of lower level behaviors
  ✦ Rounded rectangles=activities, lines are control flow
  ✦ Decisions (diamonds), forks and joins (concurrency), swimlanes

• State Machine Diagrams
  ✦ States an object can go through in response to external events,
  ✦ State is a node, event is a directed edge labeled: Event[Guard] / Action

Java: Introduction

• Compilation, execution (byte code)
• Features
  ✦ Object-oriented, inheritance, polymorphism, garbage collection
  ✦ Exception handling, concurrency, Persistence, platform independence
• Objects are references (pointers)
• Types:
  ✦ Primitive types
  ✦ arrays
  ✦ classes, methods
• Operators, assignment, control flow
  ✦ Similar to C++

Java: Input/Output

• Byte Streams
  ✦ Standard I/O streams
• Character Streams
  ✦ Readers, Writers
• Reading from the keyboard
  ✦ use EasyIn or scanner
• Writing to the screen (formatting)
• Object serialization
  ✦ ObjectOutputStream, ObjectInputStream
  ✦ readObject, writeObject
• General File I/O

Java: Inheritance

• Composition
• Inheritance
  ✦ hierarchy, superclass, subclass,
  ✦ overriding methods, upcasting, constructors
• Polymorphism
  ✦ upcasting, extensibility
• Abstract methods and classes
• Interfaces
  ✦ Multiple inheritance
  ✦ Sorting: implementing Comparable
  ✦ Extending an interface
Java: Exceptions and Threads

• Exceptions
  ✦ Semantics (how exceptions are thrown/caught), syntax
  ✦ Catch or specify requirement
  ✦ finally block
  ✦ Runtime exceptions

• Threads
  ✦ Thread class, Runnable interface
  ✦ Using the above to implement multi-threading
  ✦ Thread methods