Midterm Review

CS 4354
Summer II 2014

Jill Seaman

Midterm Exam

- Tuesday, July 22
- Closed book, closed notes, clean desk

- Java Lectures (see Thinking in Java text)
- Textbook: Chapter 2

- 30% 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/classes/code
  - ✦Tracing code (what is the output)
  - ✦Reading diagrams (what does it mean)

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

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
- Java library, API
  - ✦import statement

Java: Introduction

- Javadoc, how to document the elements of a program
- Operators, assignment, control flow
  - ✦Similar to C++
- String, toString
- Constructors, this
- Packages
- static, final
- public, private, protected, [package]
- ArrayList (or arrays)
  - ✦Know how to write code with one of these

Java: Input/Output

- Input using a Scanner
- Output using System.out.println()
- Formatting using the DecimalFormatter

- Object serialization
  - ✦ObjectInputStream, ObjectOutputStream
  - ✦readObject, writeObject
  - ✦Understand how it works.

Java: Inheritance

- Reuse: Composition and Inheritance
- Inheritance
  - ✦class hierarchy: superclass, subclass, (extends keyword)
  - ✦overriding methods, upcasting, constructors
- Polymorphism
  - ✦upcasting, polymorphic functions, dynamic binding, extensibility
- Abstract methods and classes
- Interfaces
  - ✦Implementing 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 (how to satisfy)
  - ✦Runtime exceptions
  - ✦Create your own exception classes (and instances)
- Threads
  - ✦Thread class, Runnable interface
  - ✦Using the above to implement multi-threading
  - ✦Thread methods

Ch 2: Modeling with UML: UML diagrams

- Use Case Diagrams
  - ✦Actor, Use case, textual descriptions
  - ✦Relationships: communication, inclusion, extension, inheritance
- Class Diagrams
  - ✦Classes, attributes, operations, objects, associations
  - ✦Boxes with three compartments, associations and links
  - ✦unidirectional, bidirectional associations
  - ✦Roles, multiplicity
  - ✦Aggregation, composition
  - ✦Inheritance (and abstract class, interface)

Ch 2: Modeling with UML: UML diagrams

- Sequence Diagrams
  - ✦Describes interactions between objects
  - ✦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 various activities
  - ✦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

Ch 2: Modeling with UML: UML diagrams

- For all of the types of diagrams:
  - ✦Know when to use them (in what software engineering activity, special contexts).
  - ✦Be able to draw simple diagrams, like for Assignment 3
  - ✦Be able to read (understand, interpret) somewhat more complicated diagrams.

Sample Questions: Multiple Choice

- You want to draw a diagram to show the different screens that your mobile phone app can display, including the events and conditions that cause the app to move from one screen to another. What kind of UML diagram is best for that?
  - (a) Use Case diagram
  - (b) Class diagram
  - (c) Activity diagram
  - (d) Sequence diagram
  - (e) State machine diagram
- _____ is an implicit type conversion performed in Java that permits an object of a subclass type to be used where an object of its superclass type is expected.
  - (a) Upcasting
  - (b) Polymorphism
  - (c) Dynamic binding
  - (d) Extensibility

Sample Questions: UML Diagrams

- Draw a **class diagram** showing the structure of data about employees of a given company. The employees attributes include name, street address, city, state, zip, and an id number. The employees may be full time, in which case they have an annual salary, or they may be part time, in which case they have an hourly pay rate. Departments have names and are composed of a collection of employees, but each employee can be in only one department. Employees work on one or more projects, which also have names. Projects may have multiple employees assigned to them. Include attributes, associations, and multiplicity, in your diagram.

- Draw a **state diagram** of the control software for a DVD player that has 5 buttons (play, pause, fast-forward, rewind, stop). The buttons are the external events, the states are idle, playback, fast-forwarding, re-winding, and paused.

Sample Questions: Java programming

- Implement in Java the Employee class structure from the question above that asks you to draw a class diagram. The Employee class should have a polymorphic function called weeklyPay. For Full time employees, their weekly pay is their salary divided by 52. For part time employees their salary is their hourly pay rate time 40. In another class called Driver, define a main function that creates an array or ArrayList of Employees of two full time and one part time employees, then iterates over the list and outputs the name and weekly pay for each employee.