Midterm Review

CS 4354
Summer II 2016

Jill Seaman

Midterm Exam

- Tuesday, July 26
- Closed book, closed notes, clean desk

- Java Lectures, Chapter 1
- OOD Process and Case Study, Chapter 2

- 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
  - ✦50 pts: 25 Multiple choice questions (scantron), may include:
    - Tracing code (what is the output)
    - Reading diagrams (what does it mean)
  - ✦50 pts: Coding and Design documentation:
    - Writing Use Cases, Writing CRC cards
    - Drawing UML diagrams (class, sequence, state)
    - Writing programs/classes/code in Java
- 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
- Primitive types, control flow, operators, assignment (like C++)
- Classes, fields, methods
- Objects are references (pointers underneath)
- Parameter passing (pass by value, but objects can be mutated)
- Constructors, this
- Packages, directories, import statement
- Java library, API

Java: Introduction

- String, toString
- ArrayList
- arrays

- Javadoc, how to document the elements of a program

- Access specifiers: public, private, protected, [package]

Java: Input/Output

- Input using a Scanner
- Output using System.out.println()
- Wrapper classes (Integer, Float, Double, etc)
- Formatting using the DecimalFormatter and/or String.format

- Object serialization
  - ✦ObjectInputStream, ObjectOutputStream
  - ✦readObject, writeObject
  - ✦Understand how it works (when to cast)
  - ✦Don't memorize the exceptions

Java: Inheritance

- Interfaces
  - ✦Using, Defining and implementing Interfaces
  - ✦Sorting: implementing Comparable<T> or Comparator<T>
- Inheritance
  - ✦class hierarchy: superclass, subclass, (extends keyword)
  - ✦overriding methods
  - ✦constructors
- Polymorphism
  - ✦upcasting, polymorphic functions, dynamic binding
- Abstract methods and classes

Java: Collections and Exceptions

- Collections
  - ✦LinkedList<T>
  - ✦Iterator<T> (next(), hasNext(), remove())
  - ✦iterator() method

- Exceptions
  - ✦Semantics (how exceptions are thrown/caught) and syntax
  - ✦Catch or specify requirement (how to satisfy)
  - ✦Runtime exceptions
  - ✦Create your own exception classes (and instances)

Ch 2: The Object-Oriented Design Process

- Analysis, Design, Implementation
- Objects and Classes
- Identifying Classes and Responsibilities
- Identifying Relationships
  - ✦Dependency
  - ✦Aggregation
  - ✦Inheritance
- Use Cases
  - ✦Actor
  - ✦textual descriptions (set of steps), with variations
  - ✦single interaction between actor and system.

Ch 2: The Object-Oriented Design Process

- CRC cards
  - ✦Classes, Responsibilities, Collaborators
  - ✦Index cards describing each class
  - ✦Walkthrough use cases to generate/develop the cards
- Class Diagrams
  - ✦Classes, attributes, operations, associations
  - ✦unidirectional, bidirectional associations
  - ✦Dependency, Aggregation (or association), Inheritance
  - ✦Multiplicity {1, 0..1, 0..n, 1..n}
  - ✦one-to-one, one-to-many, many-to-many

Ch 2: The Object-Oriented Design Process

- Sequence Diagrams
  - ✦Describes interactions between objects
  - ✦Objects, lifelines, activation boxes
  - ✦Messages from one object to another (must be methods on the receiving object), messages run in sequence top to bottom.
- State Machine Diagrams
  - ✦States an object can go through in response to external events,
  - ✦State is a node
  - ✦Transition is a directed edge labeled with the event that causes it
- For all of the types of diagrams:
  - ✦Be able to draw simple diagrams, like for Assignment 3
  - ✦Be able to read (understand, interpret) 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.