Final Exam Review

CS 3354
Spring 2017

Jill Seaman

Final Exam

- Monday May 8
  - ✦section 251: 2:00pm - 4:30pm, Derr 234
  - ✦section 252: 8:00am - 10:30am, Derr 240
- Closed book, closed notes, clean desk
- Comprehensive: covers entire course
- 35% of your final grade
- Bring your ID card
- Bring a number 2 pencil and eraser.
- No headphones/earphones

Exam Format

- 100 points total
  - ✦48 pts: 24 Multiple choice questions (scantron), may include:
    - Tracing code (what is the output)
    - Reading diagrams (what does it mean, is it a good design)
  - ✦52 pts: Coding and Design:
    - Drawing UML diagrams (class, sequence)
    - Writing and modifying programs/classes/code/JUnit tests in Java
    - Applying Design Patterns
- Each question will indicate how many points it is worth

Content

100 points total, **approximate** break down:

- Units 1+2 (30pts) Java Basics, Inheritance, Collections, Exceptions
- Unit 3 (20pts) OOD + UML
- Unit 4 (20pts) Class design + test (JUnit)
- Unit 5 (15pts) Design Patterns
- Unit 6 (15pts) Multithreading

Unit 1: Java Basics

How are these different from C++?

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

Unit 1: Java Basics

- String, toString, ArrayList, and arrays
- Javadoc, how to document the elements of a program
- Input using a Scanner, Output using System.out.println()
- Wrapper classes (Integer, Float, Double, etc)

- Object serialization
  - ✦ObjectInputStream, ObjectOutputStream
  - ✦readObject, writeObject
  - ✦Understand how it works
  - ✦Don't memorize the exceptions

Unit 2: 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

Unit 2: Java: Collections and Exceptions

- Collections
  - ✦LinkedList<T>, ArrayList<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)

Unit 3: Object-Oriented Design and UML

- Identifying Classes and Responsibilities
- Identifying Relationships
  - ✦Dependency, Aggregation, Inheritance
- Use Cases
  - ✦textual descriptions (set of steps), with variations
  - ✦single interaction between actor and system.
- CRC cards
  - ✦Classes, Responsibilities, Collaborators
  - ✦Index cards describing each class
  - ✦Walkthrough use cases to generate/develop the cards

Unit 3: Object-Oriented Design and UML

- Class Diagrams
  - ✦Classes, attributes, operations, associations
  - ✦Dependency, Aggregation (or association), Inheritance
  - ✦Multiplicity {1, 0..1, 0..n, 1..n} one-to-one, one-to-many, many-to-many
- Sequence Diagrams
  - ✦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

Unit 4: Class Design + Test

- The Importance of Encapsulation
  - ✦Sharing Mutable References unintentionally
  - ✦Separating Accessors and Mutators
  - ✦Side effects
  - ✦Sharing Mutable References intentionally (Law of Demeter)
- Analyzing the Quality of an Interface
  - ✦cohesion
  - ✦completeness
  - ✦convenience
  - ✦clarity
  - ✦consistency
- See Assignment 4

Unit 4: Class Design + Test

- JUnit
  - ✦Open source framework for writing and running unit tests
  - ✦Provides automation
  - ✦Annotations: @Test, @Before, @After, @Ignore
  - ✦Assert Methods: fail, assertTrue, assertFalse, assertEquals, assertNull
  - ✦Separation of testing code from production code
  - ✦Testing methods, classes, and collaborating classes.
  - ✦Be able to write simple test cases, using the Annotations and Assert methods as we did in Assignment 4.

Unit 5: Design Patterns

- Concepts
  - ✦Delegation
- Design Patterns
  - ✦ Iterator Pattern
  - ✦ Adapter Pattern
  - ✦ Strategy Pattern
  - ✦ Observer Pattern
  - ✦Composite Pattern
  - ✦Decorator Pattern
  - ✦Facade Pattern
- Be familiar with the class diagrams.
- Be able to apply them.
- See Assignment 5

Unit 6: Multithreading

- Thread class and Runnable interface
- Thread methods
  - ✦run()
  - ✦start()
  - ✦yield()
  - ✦sleep()
  - ✦join()
  - ✦interrupt()
- Race conditions, solved using:
  - ✦Locks
  - ✦Synchronized methods
- Deadlocks and Conditions
- See Assignment 6

Sample Questions: Multiple Choice

- You are designing a datatype to store a mathematical expression composed of binary operations (plus, minus, times, divide) and numbers, such as: (3+4)/(6*2). These expressions can be drawn as a tree with the operations as the node values and the numbers as the leaves. What design pattern should you use for this?

  (a) adapter (b) composite (c) facade (d) iterator

- Which of the following methods in the java.lang.Thread class creates a separate thread (i.e. initiates concurrency)?

  (a) join() (b) start() (c) run() (d) interrupt()

Sample Questions: Class Design

- Rewrite (part of) the following code, so that it does not violate encapsulation (information hiding):

```
public class Time {
   private int hours, minutes;
   public Time (int h, int m) { hours = h; minutes = m; }
   public Time (Time x) {hours = x.hours; minutes = x.minutes; }
   public String toString {
      return Integer.toString(hours)+”:”+Integer.toString(minutes);
   void addMinute() {
     if (minutes==59) {
        minutes = 0; hours++;
     } else
        minutes++;
   }
}
public class TravelClock {
   private Time currentTime, alarm;
   private int temperature;
   Time getCurrentTime() { return currentTime; }
}
```

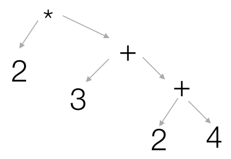
Sample Questions: JUnit

- Write a JUnit test for the Time class to test the addMinute method:

```
public class Time {
   private int hours, minutes;
   public Time (int h, int m) {
      hours = h; minutes = m;
   }
   public String toString {
      return Integer.toString(hours)+":"+Integer.toString(minutes);
   void addMinute() {
     if (minutes==59) {
        minutes = 0; hours++;
     } else
        minutes++;
   }
}
```

Sample Questions: Design Patterns

- Use the COMPOSITE pattern to implement arithmetic expressions involving the binary operations of addition and multiplication of numbers. Add an eval() function to evaluate the expression.



$2 * (3 + (2 + 4))$

- this example evaluates to 18.

Sample Questions: Java programming

- 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. Salaried employees have an annual salary. Hourly employees have an hourly pay rate. Departments have names and a group 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.
- Implement in Java the Employee class structure described above. The Employee class should have a polymorphic function called weeklyPay. For Salaried employees, their weekly pay is their salary divided by 52. For Hourly 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.

Office Hours during finals week

Day	Date	Time
W	5/3	11:00am-12:00noon
Th	5/4	2:00pm-3:00pm
M	5/8	11:00am-12:00noon
T	5/9	11:00am-12:00noon
		and by appt!