CS390 - FPP Midterm Review Points

The midterm will consist of 2 kinds of questions, as follows:

Theory – These will be composed of Knowledge-based questions, including Short answer questions, True/False questions and Multiple Choice questions.

Programming Skill/Java Coding/Problem solving – The question(s) of this kind will be ones requiring problem-solving, and testing your programming and Java coding skills. For instance, you may be given a problem statement/description and required to read it thoroughly, understand the requirements and then implement Java code for the solution.

The midterm exam will be computer-based exam; you will be expected to use a computer to access the exam via Sakai/ProctorTrack. There will be some coding on the exam, so you should be prepared to write code in Java, using the JDK, Eclipse IDE or any other IDE or Java language coding/execution tool(s).

The exam duration will be 2 hours (timed from 10am to 12noon).

The following are relevant review topics/questions:

Lesson 1 – Introduction to Java and the Eclipse Development Environment

1. Java as a software platform – the Java programming language, the Java Runtime Environment (JRE), the JVM, the Java Class Library, the JDK and tools (such as the Java Compiler, Jar, Javadoc, JShell etc.)
2. IDEs for Java programming – Eclipse, IntelliJ IDEA, Netbeans etc.
3. The Object-oriented programming (OOP) paradigm and Java as an OOP language.
4. Unit Testing - the JUnit testing framework and its integration within Eclipse IDE.
5. Beginning Java Application development – Java Console/Cmdline Application, etc.

Skills:

- How to obtain the JDK and Eclipse and how to setup and work with a Java software development environment/tools.
- Coding a simple, first Java Console (Command-line) application, using a simple Code/Text Editor and compiling on the command window, using the Java language compiler tool (javac.exe).
- Coding, compiling and running a simple, first Java Console application using Eclipse.
- How to deploy and run a Java App (e.g. creating executable jar, create a run script etc).
- How to create and run Test cases using the JUnit test library/framework with Eclipse.

Lesson 2 – Fundamentals of programming with Java

6. Data types:
    a. The Primitive data types
    b. Other data types (object types)
7. Operators:
    a. Arithmetic operators
    b. Unary (arithmetic) operators (increment and decrement operators)
    c. Relational and Boolean/Logical operators
    d. Bitwise operators
8. Expressions and Statements – ternary expressions, assignment statement etc.
9. Java Strings
10. Control Flow:
    a. Conditional logic
    b. Loops and iteration:
        i. For… loop
        ii. While loop
        iii. For each
    c. The switch statement
11. Arrays
12. Java language keywords – access specifiers, operators, types etc.
13. Coding style rules
14. Comments in source code; including Javadoc
15. Declaration – Class, Variable, Constant, Method etc.
16. Reading input data from the console – the Scanner class, System.in etc.

Skills

- Writing correct code in Java language to solve a problem.

Lesson 3 – Objects and Classes

17. Object Oriented Programming paradigm and OOP concepts and principles
18. What is a class?
19. What is an Object?
20. How Ojects are Created, Used and Destroyed
21. The Garbage Collector and Garbage Collection process
22. Some essential classes in the Java class library:
    a. How to handle dates using:
        i. java.util.Date class
        ii. java.util.GregorianCalendar
        iii. java.time.LocalDate
23. User-Defined clases
24. Constructors

25. The “this” keyword
26. Access levels and access modifiers (specifiers) – private, package-level (default), protected, public.
27. Accessors (getters) and Mutators (setters).
28. How to prevent data mutability using Cloneable interface and the clone() method e.g. issue with returning a java.util.Date type from a getter method.
29. New Date/Time API – use of java.time.LocalDate
30. How to make a class immutable.
31. Wrapper classes of primitives, Boxed primitives and autoboxing.
32. Common methods in Wrapper classes – parseXXX, compareTo
33. The static keyword, Static fields and methods, Defining Constants, and Using Enums.
34. How objects and variables are stored in memory (i.e. registers, the stack, the heap, static storage etc)
35. Passing parameters to methods – Call by Reference versus Call by Value
36. Data formatting – String.format(), System.out.printf, java.text.SimpleDateFormatter etc.
37. Overloading Constructors; calling one constructor from another
38. Overloading methods
39. Initialization of field data in a class
40. Initialization blocks – object initialization block, static initialization block
41. Sequence (order) of execution when a constructor is called
42. Making a constructor private and its use in implementing the Singleton pattern (the lazy initialization and not-threadsafe way)
43. Use of packages in Java and their naming convention
44. Package level access – for class, variables, methods
45. Importing classes; static imports
46. Principles of Good obect design:
    a. Encapsulation
    b. Single Responsibility principle

Skills

- Applying correct coding style.
- Writing syntactically correct Java code
- Applying OOP principles in coding a Java solution

Lesson 4 – Inheritance, Interfaces and Polymorphism

- What is inheritance? Give example by: Draw a UML diagram; Write a simple code example
- Subclass versus Superclass
- Base class versus Derived class
- Data Abstraction

- Generalization versus Specialization; general versus specific
- Identifying correct Inheritance relationship – the “IS A” test and LSP. What is LSP?
- What is Polymorphism?
- Rules for subclass constructor
- Order of execution
- Abstract class versus Interface
- Interfaces in Java
    - Java.lang.Comparable, ~~java.util.Comparator – how are these 2 related?~~
    - Functionals Interfaces
    - Default methods in Interface; Static methods in interface
- Inheritance and the java.lang.Object super class
    - toString() method
    - the equals() method: how to override
        - instanceof strategy
        - same class strategy
        - Issues concerning override of equals(), use of ‘final’ to prevent further inheritance and favoring Composition over Inheritance
    - ~~hashCode() and rules governing relationship with equals()~~
    - clone() method and Cloneable interface
        - Shallow copying
        - Deep copying
- The protected keyword and inheritance hierarchy; implementing the clone() method
- The “Open-Close” principle
- Reflection API:
    - What is Reflection? And what can it be useful for?
    - The Class class
    - The Contructor class