CSCI 135 – Introduction to Computer Science I
Exam III Study Outline

I. Java Basics
- A. How Java is compiled and run
- B. Java program parts
- C. Eclipse / Command Line Interface

II. Data Types
- A. Variables
  1. Declaration
  2. Assignment
  3. Scope of a variable
- B. Basic data types
- C. Type Conversion
- D. Boolean operators
- E. Creating random numbers

III. Error Types and Debugging
- A. Runtime errors (semantic errors)
- B. Compile time errors (syntax errors)
- C. Debugging
  1. Preventing bugs
     a. Programming process
        1. Understand the problem
        2. Work out the logic
        3. Write code
           a. Break problem into parts – code and test one at a time
        4. Test/Debug
  2. Finding bugs
     a. Trace print statements

IV. Conditionals
- A. Format
- B. How to use them

V. Loops
- A. while
- B. for
- C. do… while
- D. Format
- E. How to use them

VI. Programming Style
- A. Comments
- B. Naming conventions – variables, classes, constants
- C. Whitespace
- D. Curly braces

VII. Command Line Input

VIII. Standard Input

- A. Interactive input
- B. Redirecting input from a file
    1. Reading from a file
- C. Redirecting output to a file
- D. Piping the output from one file to the input of another
- E. StdIn.java

IX. Arrays

- A. Declaration
- B. Creation
- C. Initializing array elements
- D. Accessing array elements
- E. Two-dimensional arrays

X. Graphics and Audio

- A. StdDraw.java and StdAudio.java
- B. Drawing Shapes
- C. Changing Color
- D. Window Coordinates (and changing them)
- E. Drawing Images
- F. Changing Window Size
- G. Animation Loop
- H. Keyboard Input
- I. Adding Sound

XI. Static Methods

- A. Using Static Methods
    1. Static methods we've already used
- B. Defining our own Static Methods
- C. Concepts
    1. Return Data Type
    2. Parameters (and data types)
    3. Modularity
    4. Calling a Method
    5. Flow of Control
    6. Variable Scope
- D. Terminology
    1. Access Modifier
    2. Return Type
    3. Parameters/Arguments
        a. Pass by Value
        b. Pass by Reference
    4. Method Name
    5. Return Statement
    6. Method Signature
    7. Overloading

XII. Libraries and Clients

- A. Terminology
    1. Library
    2. Client
    3. API

4. Implementation
B. Unit Testing
C. Modular Programming
D. Dependency Graph

XIII. Recursion
A. Mathematical Induction
1. Base Case
2. Induction Step
B. Recursion vs. Iteration
C. Things to Avoid
1. Missing Base Case
2. No Convergence
3. Stack Overflow

XIV. Classes and Objects
A. Classes as Data Type
B. Classes (Templates)
C. Objects (Instances of a Class)
1. Instance Variable – things they know
2. Instance Methods – things they do
3. Access Modifiers for both
4. Instantiating objects
D. Constructors
E. Arrays of Objects
F. Null Value

XV. Objects, Primitives and References
A. Primitive Type vs. Object Reference Type
B. Orphaned Objects
C. Garbage Collection
D. Aliasing
E. this

XVI. Designing Data Types
A. Data Encapsulation
1. Hiding internal representation
2. Separating implementation from design specification
3. Client / API / Implementation
4. Access modifiers
a. public
b. private
c. (none) - default
d. protected
e. final
5. Immutability
B. Equality
C. Procedural Programming (verbs) vs. Object Oriented Programming (nouns)
D. Instance methods your class may need:
1. Constructor(s)
2. equals()
3. toString

4. Getters (Accessor Methods)
5. Setters (Mutator Methods)

XVII. Software Development Life Cycle

A. Requirements Analysis (Understand the Problem)
B. Design (Work out the Logic)
C. Implementation (Convert it to Code)
D. Test/Debug
E. Maintenance

XVIII. Object Oriented Design

A. Identify Objects (Nouns)
B. Identify what Information Each Class Needs
C. Identify what each Class Needs to Do
D. Identify Use Cases
E. Define the API
F. Define the Instance Variables
G. UML Diagrams for Object Oriented Design
H. Measures of Good Design
    1. (Low) Coupling
    2. (High) Cohesion
    3. Sufficiency
    4. Completeness
    5. Primitiveness
I. Design Concepts
    1. Simplicity
    2. Abstraction
    3. Encapsulation
    4. Modularity
    5. Abstraction Hierarchy
    6. Strong Data Typing
    7. Concurrency
    8. Object State, Behavior and Identity
    9. Classes vs. Objects
    10. Inheritance