Midterm-review

Advanced OO Programming – COSC 3P91

Introduction to Object Oriented Programming

- **Design Principles**
  - Abstraction Principle
  - Program to an Interface
  - Favour Composition over Inheritance
- **Data Abstraction**
  - Abstract Data Types
- Encapsulation
- Object-Oriented Concepts
  - Classes
    - static modifiers
    - Nested Classes
      - Local classes
      - Anonymous Classes
    - Abstract Classes
  - **Inheritance**
    - **Subtyping**
    - Inheritance for specification
    - Inheritance for extension
    - Inheritance for specialization

Introduction to Object Oriented Programming

- Unified Modeling Language
  - Definition
  - Purpose
  - Benefits
  - **Class diagrams**
    - Elements and Relationships
    - Notation
    - Relationship Variations
      - Inheritance
      - Composition, aggregation, association, dependency

Generics, Polymorphism, and Interfaces

- Interfaces
  - **Definition**
  - **Difference between Abstract classes and interfaces**
  - Interface as a Type
  - Implementing and extending interfaces
  - **Abstract Methods**
  - **Default Methods**
  - **Static Methods**
- Overriding versus overloading
- Overriding and hiding
- **Final Classes and Methods**
- Enumeration types

Generics, Polymorphism, and Interfaces

- Generics
  - Definition
  - **Generic Type**
    - **Type parameter and type argument**
  - Generic Class
  - Instantiating a Generic Type
  - Multiple Type Parameters
  - Raw Types
  - **Generic Methods**
  - **Bounded Type Parameters**
  - **Generic Subtypes**
  - Type Inference
  - Target Types
  - Restrictions on Generics
  - **Wildcards**
    - Bounded Wildcards
    - Wildcards and Subtyping

Utility Classes, Collections, Files, and Streams

- **Lambda Expressions**
  - Anonymous Classes?
  - Functional Interfaces in Java
  - **Syntax**
  - Parameters
  - Method References
  - Lambda Expressions and Method References
- Utility Classes
  - **Collection**
  - Set
  - **List**

Utility Classes, Collections, Files, and Streams

- Input and Output in Java
  - **Streams**
    - Input
    - Output
    - SequenceInputStream
    - Filtering
  - Piped input and output
  - Character Streams

Exception Handling

- **Definition**
- Best Practices
- Exception Handler
- Catching Exceptions
- Types of Exceptions
  - Checked exception
  - Unchecked exception
- **try Block**
- **catch Block**
- **finally Block**
- try-with-resources Statement
- Stack Winding
- **Throwing Exceptions**

Exception Handling

- **Chained Exceptions**
- Logging
- Creating Exception Classes
- **Advantages** of Exceptions
  - Separating Error-Handling Code from "Regular" Code
  - Propagating Errors Up the Call Stack
  - Grouping and Differentiating Error Types
- **Java Exception Antipatterns**
- Assertions
  - Simple
  - Complex