Revision Notes for 7COM1025 Programming for Software Engineers

Object-Oriented Programming (OOP)

- Key Concepts
    - Class: A blueprint/template to create specific types of objects. Every class has two main features:
        - Attributes (fields)
        - Behaviour (methods)
    - Object: Basic unit representing real-world entities. Instances of classes.
    - Method: Functions defined within classes that operate on class instances.
    - Attribute: Properties or fields that an object knows about itself.
- Principles of OOP
    - Encapsulation:
        - Bundling data and related functions, restricting direct access to object components.
        - Make fields private, provide mutator/accessor methods.
    - Inheritance:
        - Mechanism where a subclass inherits properties and behaviour of a superclass.
        - Promotes information management in a hierarchical structure.
    - Polymorphism:
        - Ability of an object to take on many forms.
        - Enables dynamic behaviour where the actual method invoked is determined at runtime.
    - Abstraction:
        - Focuses on essential details and ignores irrelevant details.
        - Enables thinking about a component without worrying about implementation.
- Other Terms
    - Cohesion: Measures how well-focused a class is in fulfilling a single, well-defined purpose.
    - Coupling: Degree of reliance of a module on other modules. Desirable to minimize coupling.

Design Patterns

- Categories of Patterns
    - Creational Patterns (object creation)
        - Factory: Simple decision-making class that returns a subclass of a base class.
        - Abstract Factory: Super-factory that creates other factories.
        - Builder: Builds complex objects incrementally.
        - Singleton: Ensures only one instance of a class exists.
    - Structural Patterns (composition of classes/objects)
        - Adapter: Translates a server interface for a client.
        - Bridge: Abstraction binding multiple implementations.
        - Decorator: Extends an object transparently.
        - Facade: Simplifies the interface for a subsystem.
    - Behavioural Patterns (object communication/behaviour management)
        - Observer: One-to-many dependency where all observers are updated when the subject changes.
        - Strategy: Defines a family of algorithms, encapsulates each one, and lets the client choose.
        - Command: Request/Action as first-class objects.
        - Template Method: Algorithm with steps supplied by a derived class.
        - Mediator: Coordinates interactions between objects.
- Pattern Elements
    - Pattern Name: Short description of the design problem.
    - Problem: Context and description of the design problem.
    - Solution: Relationships, responsibilities, and collaborations.
    - Consequences: Results and trade-offs.

Frameworks

- Framework: Integrated set of collaborating software artifacts providing a reusable architecture for a family of applications.
    - Characteristics:
        - High-level architecture reuse.
        - Programming language specific.
    - Example: Spring framework
- Difference Between Frameworks and Patterns
    - Frameworks:
        - High-level architectural reuse.
        - Implementation-specific.
    - Patterns:
        - Lower-level, more primitive.
        - General and abstract.

Java Programming

- Static Keyword
    - Static Variable:
        - Shared among all instances of a class.
        - Gets memory only once at class loading time.
    - Static Method:
        - Belongs to the class rather than an instance.
        - Can access/modify static data members.
- Checked vs Unchecked Exceptions
    - Checked Exceptions:
        - Known to the compiler.
        - Must be handled using try-catch blocks.
    - Unchecked Exceptions:
        - Runtime exceptions not checked at compile-time.
        - Typically caused by logical errors.

QUESTION 1 Explain the following terms:

a) Class; (5 marks)

A class is a template or blueprint or a set of instructions to create a specific type of object.

b) Encapsulation; (5 marks)

Encapsulation is a mechanism of bundling the data, and the functions that use them. It is a way of restricting or controlling access to certain object components.

c) Inheritance; (5 marks)

Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another. With the use of inheritance, the information is made manageable in a hierarchical order.

d) Attribute; (5 marks)

Attributes are the properties or fields that an object knows about itself. An attribute is another term for a field.

e) Cohesion; (5 marks)

Cohesion refers all about how a single class is designed. Cohesion is the Object Oriented principle most closely associated with making sure that a class is designed with a single, well-focused purpose. The more focused a class is, the cohesiveness of that class is more.

QUESTION 2

a) List and explain the three categories of design patterns. List two pattern names for each category. (15 marks)

The design patterns can be categorised as: Creational: Deal with initialising and configuring classes and objects. Address problems of creating an object in a flexible way. Separate creation from operation/use; Structural: Deal with decoupling interface and implementation of classes and objects. Address problems of using O-O constructs like inheritance to organise classes and objects; Behavioural: Deal with dynamic interactions among societies of classes and objects. Help you define the communication between objects in your system and how the flow is controlled in a complex program. Creational pattern names: any two from Singleton, Factory, Abstract Factory, Builder, Prototype Structural pattern names: any two from Composite, Facade, Adapter, Bridge, Decorator, Flyweight, Proxy Behavioural pattern names: any two from Mediator, Observer, Chain of Responsibility, Command, Iterator, Interpreter, Memento, State, Strategy, Template Method, Visitor

b) Explain Factory Pattern and Abstract Factory Pattern. (10 marks)
Factory Pattern provides a simple decision making class that returns one of several possible subclasses of a base class depending on the data that are provided; allows the sub-classes to choose the type of objects to create; does not expose creation logic to the client and refer the created object using a standard interface; also known as Virtual Constructor; the most used design pattern in Java.