DAC 0521 MET-M Writeup - Java Module 1 Session 04

Object Oriented Programming (OOP) - It is a programming methodology (a style based on sound principles) in which the implementation of a large software is divided into smaller units called objects each containing its own state (data) and supporting its own behavior(operations).

Basic Elements

1: Class - It defines a set of variables known as fields whose values indicate the state of a particular type of object and a set of functions known as methods whose operations describe the behavior of that object.
An activatable type is a class that supports creation of new objects known as its instances and such a class defines a special method known as its constructor which is called whenever a new instance of that class is created to put this instance in its initial state.

2: Subclass - It (derived class) extends an existing class known as its superclass (base class) to define additional fields or methods or to override (provide new behavior) for its existing methods.
An abstract type is a non-activatable type which can define a set of pure (behavior less) methods known as its interface and these methods must be implemented by its activatable subtypes.

Basic Mechanisms

1: Binding - Every object as a unique identity and when a method defined by a class is called on its object the identity of this object is automatically passed (as an argument) to the implementation of that method.

2: Dispatch - A method defined by a class can be called on an object of its subclass and if this subclass has overridden that method then the implementation provided by the subclass will be invoked.

Basic Relationships

1: Containment - A object of one class exhibits ‘has a’ relationship with an object

of another class which it holds in one of its own fields. This relationship is called composition (Hotel has a Room) if the outer object controls life-time of the inner object otherwise it's called aggregation (Room has a Guest).

2: Inheritance - An object of a subclass exhibits 'is a' relationships with its superclass. This relationship is called specialization (sports-car is a car) if the superclass is activatable (non-abstract) otherwise it is called realization (car is a vehicle).

SOLID Principles

1: Single Responsibility Principle (SRP) - There should be exactly one reason for changing the implementation of an object.
Rule of Abstraction - Define members in a class which are required by all of its objects in an identical manner.

2: Open Closed Principal (OCP) - The functionality exposed by an object should open for extension but closed for modification.
Rule of Encapsulation - Publish methods of a class while hiding the fields of that class.

3: Liskov's Substitution Principle (LSP) - An object of a subtype should be consumed as if it an object of the original type.
Rule of Polymorphism - Invoke methods of a subclass indirectly through its superclass.

4: Interface Segregation Principle (ISP) - The consumer of an object should not be forced to depend upon an interface of that object which it does not require.
First Rule of Loose Coupling - Include identical methods implemented by different classes as pure methods in their common abstract supertype.

5: Dependency Inversion Principal (DIP) - An object should be consumed through its interface and not through its implementation.
Second Rule of Loose Coupling - Expose an activatable type indirectly through its abstract supertypes.