Assignment 3

Q5. Achieve loose coupling in java by using OOPs concept?

Ans)To achieve loose coupling in Java using OOPs concepts, you can follow these rules:

1. Encapsulation: Encapsulate the internal details of a class and provide public methods to interact with the class.
2. Abstraction: Use interfaces or abstract classes to define contracts and provide a common interface for interacting with different implementations.

3. Dependency Injection: Instead of directly creating dependencies inside a class, use dependency injection to provide them from external sources. This allows for easier swapping of dependencies and reduces the coupling between classes.
4. Dependency Inversion Principle: Follow the principle of depending on abstractions, not on concrete implementations. This means programming against interfaces or abstract classes rather than specific implementations. This allows for flexibility and easy substitution of implementations.
5. Interface Segregation Principle: Split interfaces into smaller, more specific interfaces to avoid forcing classes to implement methods they don't need. This helps in reducing dependencies and provides more flexibility.

Q6. What is the benefit of encapsulation in java?

The benefits of encapsulation in Java can be summarised as follows:

1. Data hiding: Encapsulation allows the hiding of internal details and provides access to data only through defined methods, ensuring data integrity and security.
2. Modularity: Encapsulation promotes modular design by encapsulating related data and behaviours within a single class, making it easier to understand, maintain, and update code.
3. Code reusability: Encapsulation enables code reusability by encapsulating functionalities into classes, allowing them to be easily reused in different parts of a program or in different programs.
4. Flexibility and maintainability: Encapsulation enables changes to the internal implementation of a class without affecting other parts of the program, leading to easier maintenance and updates.
5.Enhanced code organisation: Encapsulation provides a clear structure and organisation to code, making it more manageable and readable.

Q7.Is java a 100% Object oriented Programming language? If not, why ?

Ans)No, Java is not considered a 100% Object-Oriented Programming (OOP) language because:
1. Primitive Data Types: Java includes primitive data types (e.g., int, float, boolean) that are not objects and do not inherit from a common class.

2. Static Methods and Variables: Java allows the use of static methods and variables, which belong to the class itself rather than individual objects. Static members are not associated with any particular instance of the class.

3. Final Classes and Methods: Java allows the declaration of final classes and methods that cannot be inherited or overridden, breaking the inheritance hierarchy.

4. Procedural Programming Features: Java supports procedural programming features, such as control structures (if-else, for, while) and the use of static variables and methods outside of class instances.

5. Lack of Multiple Inheritance: Java does not support multiple inheritance, where a class can inherit from multiple parent classes. It uses interfaces to achieve multiple inheritance of behaviour.

Q8.What are the advantages of abstraction in java?
Ans)Advantages of abstraction in Java:
1. Simplified Code: Abstraction helps in simplifying complex code by providing a high-level view and hiding unnecessary details.

2. Code Reusability: Abstraction allows the creation of reusable components that can be used across different projects, promoting code reuse and reducing development time.

3. Security: Abstraction provides a level of security by hiding sensitive implementation details and exposing only necessary interfaces.

4. Flexibility: Abstraction allows for easy modification and maintenance of code, as changes can be made to the underlying implementation without affecting the code that uses the abstracted interface.

5. Enhances Maintainability: By reducing code complexity and dependency on implementation details, abstraction improves code maintainability and makes it easier to identify and fix issues.

Q9.What is an abstraction explained with an Example?

Ans)Abstraction is a fundamental concept in object-oriented programming that allows us to represent complex systems or entities in a simplified manner by focusing on essential characteristics and hiding unnecessary details. It provides a way to create a model or representation of a real-world entity or system.

For example, consider a car. We can abstract the car by defining its essential properties and behaviours, such as the ability to accelerate, brake, and steer. We don't need to know the intricate details of how the engine works or the internal mechanisms of the car. This abstraction allows us to work with the car as a high-level entity without getting bogged down by the complexities of its implementation. We can interact with the car using its defined interface, such as pressing the accelerator pedal or turning the steering wheel, without worrying about the internal workings of the car.

Q10.What is the final class in Java?

Ans)In Java, the final keyword can be applied to a class to make it a final class. A final class is a class that cannot be subclassed or extended. It means that no other class can inherit from a final class.

When a class is declared as final, it signifies that the design of the class is complete and should not be modified further. It ensures that the behaviour of the final class cannot be changed by any subclasses.

There are a few reasons why a class might be declared as final:

1. Security: By making a class final, it prevents any potential security vulnerabilities that could arise from subclassing and overriding certain behaviours.

2. Performance: A final class can be optimised by the compiler, as it knows that the class will not be extended and certain optimizations can be applied.