Encapsulation

Assignment Question

Assignment Questions

1. **What is Encapsulation in Java? Why is it called Data hiding?**

   Ans :

- **Encapsulation in Java is a fundamental concept of object-oriented programming that involves bundling data (attributes) and methods (functions) that operate on the data into a single unit, known as a class.**
- **The primary purpose of encapsulation is to restrict direct access to the internal state (data) of an object and provide controlled access through methods.**
- **It promotes the idea of hiding the implementation details of an object's data while providing well-defined interfaces for interacting with the object.**
- **Data Hiding: Encapsulation is often referred to as "data hiding" because it restricts direct access to an object's internal data from outside the class. This helps maintain the integrity and consistency of the data by preventing unintended or unauthorized modifications. Access to the data is only allowed through methods provided by the class.**

2. **What are the important features of Encapsulation?**

   Ans :

- **Data hiding: Encapsulation hides the data members of a class from outside access. This means that the data cannot be accessed directly by other classes. This can be useful to protect the data from being modified by unauthorized code or to ensure that the data is used in a consistent way.**
- **Reusability: Encapsulated code can be reused in other projects. This is because the data members of the class are hidden from outside access. This means that the code does not need to be modified to be reused in another project.**
- **Flexibility: Encapsulation provides flexibility by allowing you to change the implementation of a class without affecting other parts of your code. This is because other parts of your code only interact with the public interface of the class, which remains unchanged.**

3. **What are getter and setter methods in Java Explain with an example**

Ans :

- **Getter and setter methods, also known as accessor and mutator methods, are a common implementation of encapsulation in Java.**
- **They are used to control access to an object's attributes (instance variables) by providing controlled read (get) and write (set) access to those attributes. Getters retrieve the values of attributes, and setters modify the values of attributes, typically with added logic for validation and consistency.**
- **Getter and setter methods are special methods in Java that are used to access and modify the data members of a class. Getter methods are used to get the value of a data member, and setter methods are used to set the value of a data member.**
- **The names of getter and setter methods are prefixed with the keywords get and set, respectively. The name of the data member is used as the rest of the method name. For example, if the data member is called name, then the getter method would be called getName() and the setter method would be called setName().**
- **Getter and setter methods are typically declared as public methods. This means that they can be accessed by any code that has access to the class. However, it is also possible to declare getter and setter methods as private or protected. This will restrict access to the methods to the class in which they are declared or to subclasses of that class**

```
    class House{
    private String Name;
    private String Colour;
    public void setName(String Name){
     this.Name = Name;
     this.Colour = Colour;
    }
    public String getName(){
       return Name;
    }
    public void setColour(String Colour){
     this.Colour = Colour;
    }
    public String getColour(){
       return Colour;
    }
}
public class JavaDemo {
    public static void main (String [] args){
```

```
            House obj = new House ();
            obj.setName ("Antilia");
            obj.setColour ("blue");
            String HouseA = obj.getName();
            String HouseB = obj.getColour();
            System.out.println (HouseA);
            System.out.println (HouseB);
        }
    }
```

Output :

```
    Antilia
    blue
```

4. What is the use of this keyword explain with an example

Ans :

- **This keyword in Java is a reference to the current object within an instance method or constructor.**
- **It is used to differentiate between instance variables (fields) and method parameters or local variables that have the same name. This allows you to access instance variables and methods of the current object.**

```
    class Teacher{
        private String Name;
        private String Subject;
        public void setName (String Name){
            this.Name = Name;
        }
        public void setSubject (String Subject){
            this.Subject = Subject;
        }
        public void show (){
            System.out.println (Name + " " + Subject);
        }
    }
    public class DemoJava {
        public static void main(String[] args){
            Teacher obj = new Teacher ();
```

```
            obj.setName("Varshab");
            obj.setSubject ("Math");
            obj.show();
        }
    }
```

Output :

```
    Varshab Math
```

5. What is the advantage of Encapsulation?

Ans :

- **Data Hiding: Encapsulation hides the internal state (data) of an object from external code. This prevents unauthorized access to or modification of an object's data, ensuring data integrity and security.**
- **Controlled Access: Access to an object's data is controlled through well-defined getter and setter methods. This allows you to enforce rules, validations, and business logic before allowing changes to the data, ensuring data consistency and correctness.**
- **Flexibility and Maintainability: Encapsulation allows you to change the internal implementation of a class without affecting the external code that uses the class. This enhances code maintainability, promotes backward compatibility, and makes it easier to evolve and improve the class over time.**
- **Abstraction: Encapsulation promotes abstraction by separating the essential properties and behaviors of an object from its implementation details. This simplifies the usage of objects, as users interact with the object through a well-defined interface, rather than needing to understand its internal complexities.**
- **Code Isolation: Encapsulation helps isolate different parts of the codebase. Changes made within the encapsulated class won't necessarily impact the rest of the program, reducing the chances of unintended side effects and making it easier to test and debug code.**
- **Enhanced Security: By hiding the implementation details and exposing only controlled methods, you reduce the risk of unauthorized access or tampering with sensitive data. This is especially important in applications that handle user authentication, financial transactions, or other security-critical operations.**
- **Modularity: Encapsulation encourages modular design by encapsulating data and behavior within individual classes. This makes it easier to manage and organize the codebase, as each class focuses on a specific responsibility.**
- **Reduced Complexity: Encapsulation simplifies the complexity of interacting with objects by providing well-defined methods and interfaces. This abstraction reduces the cognitive load on developers, making it easier to work with and reason about code.**

- **Promotion of Best Practices: Encapsulation discourages the direct manipulation of data from outside the class. Instead, it promotes the use of getter and setter methods, which can enforce best practices, such as validation and error handling.**

- **Code Reusability: Encapsulated classes with well-defined interfaces can be reused in different parts of an application or in other projects, as long as the external contract (i.e., the method signatures) remains the same.**

6. How to achieve encapsulation in Java? Give an example.

Ans :

- **Make Data Members Private: Declare class attributes (data members) as private. This restricts direct access to these attributes from outside the class.**

- **Provide Getter Methods: Create public getter methods to access the values of private attributes. Getter methods should return the values of the attributes without allowing direct access to them.**

- **Provide Setter Methods: Create public setter methods to modify the values of private attributes. Setter methods should include validation logic, if necessary, to ensure that the new values meet specific criteria.**

```
    class Tree {
     private String Fruit;
     private int Number;
     public void setData(){
          Fruit = "Mango";
          Number = 18;
     }
     public void show (){
          System.out.println (Fruit + " " + Number);
     }
}
     public class Demojp {
     public static void main (String [] args){
          Tree obj = new Tree();
          obj.setData();
          obj.show();
     }
}
```

Output :

```
  Mango 18
```