Encapsulation

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

Ans:

Encapsulation in Java is a process of wrapping code and data together into a single unit, for example, a capsule which is mixed of several medicines.

We can create a fully encapsulated class in Java by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it.

It called data hiding because in encapsulation the data in a class is hidden by other classes by using the concept of data hiding which is achieved by making the member or method of a class private.

2. What are the important features of Encapsulation?

Ans:

- The functionality is defined *in one place* and not in multiple places.
- It is defined in a logical place – the place where the data is kept.
- Data inside our object is not modified unexpectedly by external code in a completely different part of our program.
- When we use a method, we only need to know what result the method will produce – we don't need to know details about the object's internals in order to use it. We could switch to using another object which is completely different on the inside, and not have to change any code because both objects have the same interface.

3. What are getter and setter methods in Java? Explain with an Example.

Ans: In Java, Getter and Setter are methods used to protect your data and make your code more secure. Getter and Setter make the programmer convenient in setting and getting the value for a particular data type.

Getter in Java: Getter returns the value (accessors), it returns the value of data type int, String, double, float, etc. For the program's convenience, the getter starts with the word "get" followed by the variable name.

Setter in Java: While Setter sets or updates the value (mutators). It sets the value for any variable used in a class's programs. and starts with the word "set" followed by the variable name.

Example:

```
package Encapsulation;

public class GetterSetter {
   private int salary;

   // a setter method that assign a
   // value to the salary variable
   void setSalary(int s)
   {
      if(s < 0 )
      {
         s = -s;
      }

      this.salary = s;
   }


   // a getter method to retrieve
   // the salary
   int getSalary()
   {
      return this.salary;
   }

   public void storeSalaryDB(int salary)
   {
```

```
        // code for storing the salary in the database
        System.out.println("Salary Stored in database");
    }

    // main method
    public static void main(String args[])
    {
        // creating an object of the class GetterSetter
        GetterSetter obj = new GetterSetter();

        obj.setSalary(-50000);

        int salary = obj.getSalary();

        // storing salary in database
        obj.storeSalaryDB(salary);

    }
}
```

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

Ans:

Here is given the 6 usage of java this keyword.

1. this can be used to refer current class instance variable.
2. this can be used to invoke current class method (implicitly)
3. this can be used to invoke current class constructor.
4. this can be passed as an argument in the method call.
5. this can be passed as argument in the constructor call.
6. this can be used to return the current class instance from the method.

Example:

```
package Encapsulation;

class Student1{
```

```
    private String name;
    private int age;

    void setData1(int age){
        this.age = age;  // assign value in instance variable age by using this
keyword
        // shadowing problem solve (this.age = age)
    }
    void setData2(String name){
        this.name = name;
    }
    public void show(){
        System.out.println(name +" "+age);
    }
}

public class ShadowingSolu {
    public static void main(String[] args) {
        Student1 s1 = new Student1();
        s1.setData1(22);
        s1.setData2("Rajan");
        s1.show();
    }

}
```

5. What is the advantage of Encapsulation?

Ans: By providing only a setter or getter method, you can make the class read-only or write-only. In other words, you can skip the getter or setter methods.

It provides you the control over the data. Suppose you want to set the value of id which should be greater than 100 only, you can write the logic inside the setter method. You can write the logic not to store the negative numbers in the setter methods.

It is a way to achieve data hiding in Java because other class will not be able to access the data through the private data members.

The encapsulate class is easy to test. So, it is better for unit testing.

The standard IDE's are providing the facility to generate the getters and setters. So, it is easy and fast to create an encapsulated class in Java.

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

Ans:

Encapsulation can be achieved by Declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.

Or

In Java, encapsulation is achieved by declaring the instance variables of a class as private, which means they can only be accessed within the class. To allow outside access to the instance variables, public methods called getters and setters are defined, which are used to retrieve and modify the values of the instance variables, respectively.

Example:

```
package Encapsulation;

class Person{

   private String name;
   private int age;

   public void setName(String name){
      this.name = name;
   }
   public String getName(){
      return name;
   }

   public void setAge(int age){
```

```
        this.age = age;
    }
    public int getAge(){
        return age;
    }
}
public class EncapsulationEx {
    public static void main(String[] args) {
        Person p = new Person();

        p.setName("Rajan");
        p.setAge(20);

        System.out.println("Name : "+p.getName());
        System.out.println("Age : "+p.getAge());
    }
}
```