Encapsulation in Java is one of the fundamental Object-Oriented Programming (OOP) concepts. It refers to the bundling of data (variables) and methods (functions) that operate on the data into a single unit or class, and restricting access to the internal details of that class. This helps protect the data from outside interference and misuse.

To achieve encapsulation in Java:

1. We declare the variables of a class as private.

2. We provide public getter and setter methods to access and update the value of a private variable.

Here’s an example demonstrating encapsulation in Java:

```
// Encapsulated class
class Student {
    // Private fields
    private String name;
    private int age;

    // Constructor
    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    // Getter for name
    public String getName() {
        return name;
    }

    // Setter for name
    public void setName(String name) {
        this.name = name;
    }

    // Getter for age
    public int getAge() {
        return age;
    }

    // Setter for age
    public void setAge(int age) {
        if (age > 0) { // Validation
            this.age = age;
        } else {
            System.out.println("Please enter a valid age.");
        }
    }
}
```

```
// Main class to test encapsulation
public class Main {
    public static void main(String[] args) {
        // Create a new Student object
        Student student = new Student("Alice", 20);

        // Access and modify private fields through getters and setters
        System.out.println("Name: " + student.getName());
        System.out.println("Age: " + student.getAge());

        // Changing the values using setters
        student.setName("Bob");
        student.setAge(22);

        System.out.println("Updated Name: " + student.getName());
        System.out.println("Updated Age: " + student.getAge());

        // Trying to set an invalid age
        student.setAge(-5);  // This will print a validation message
    }
}
```

Explanation:

1. Private Fields: The name and age fields in the Student class are private. This prevents direct access from outside the class.

2. Public Getters and Setters: We provide public getName(), setName(), getAge(), and setAge() methods, which allow controlled access to the name and age fields.

3. Validation in Setters: The setAge method includes a basic validation to ensure that age cannot be set to a negative value.

Output:

```
Name: Alice
Age: 20
Updated Name: Bob
Updated Age: 22
Please enter a valid age.
```

This is encapsulation in action, as it hides the internal details and allows controlled access to the properties through methods