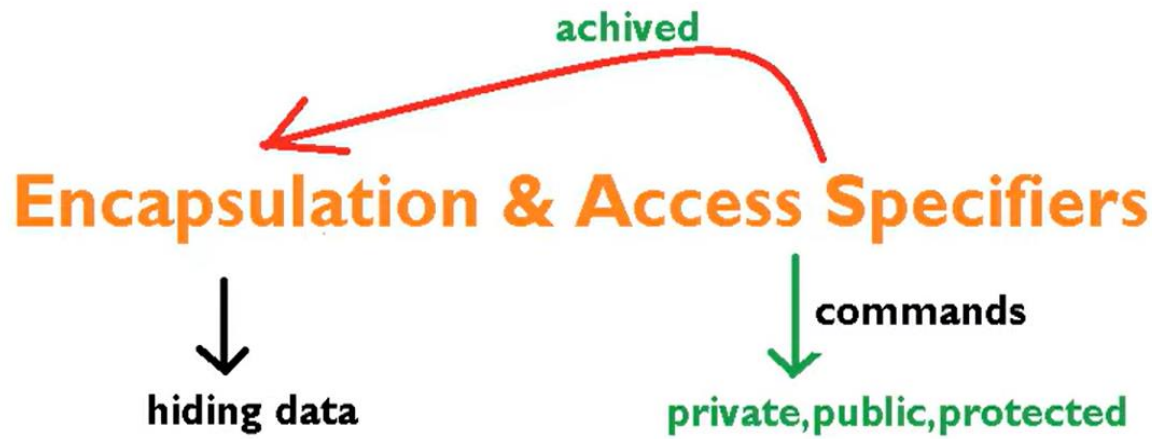
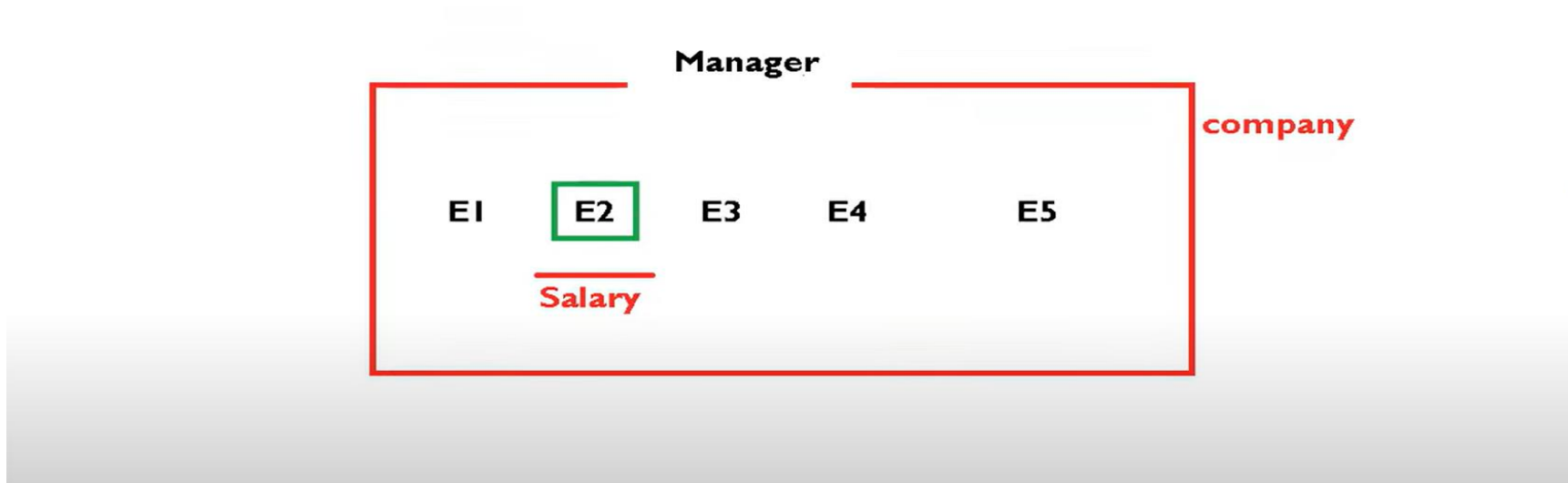
Encapsulation

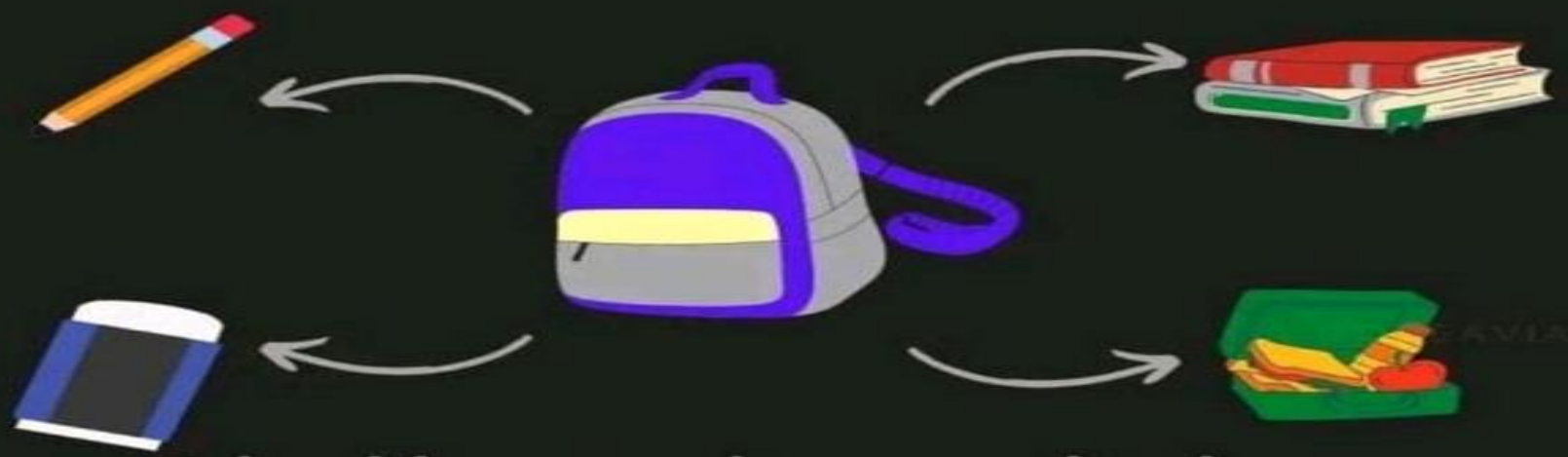
Encapsulation

- The meaning of Encapsulation, is to make sure that "sensitive" data is hidden from users.
- encapsulation describes bundling data and methods that work on that data within one unit.
- encapsulation refers to the bundling of data with the mechanisms or methods that operate on the data. It may also refer to the limiting of direct access to some of that data
- As one example, encapsulation can be used to hide the values or state of a structured data object inside a class, preventing direct access to them by clients in a way that could expose hidden implementation details or violate state invariance maintained by the methods.


achived
Encapsulation & Access Specifiers
commands
hiding data
private,public,protected


Manager
company
E1
E2
E3
E4
E5
Salary

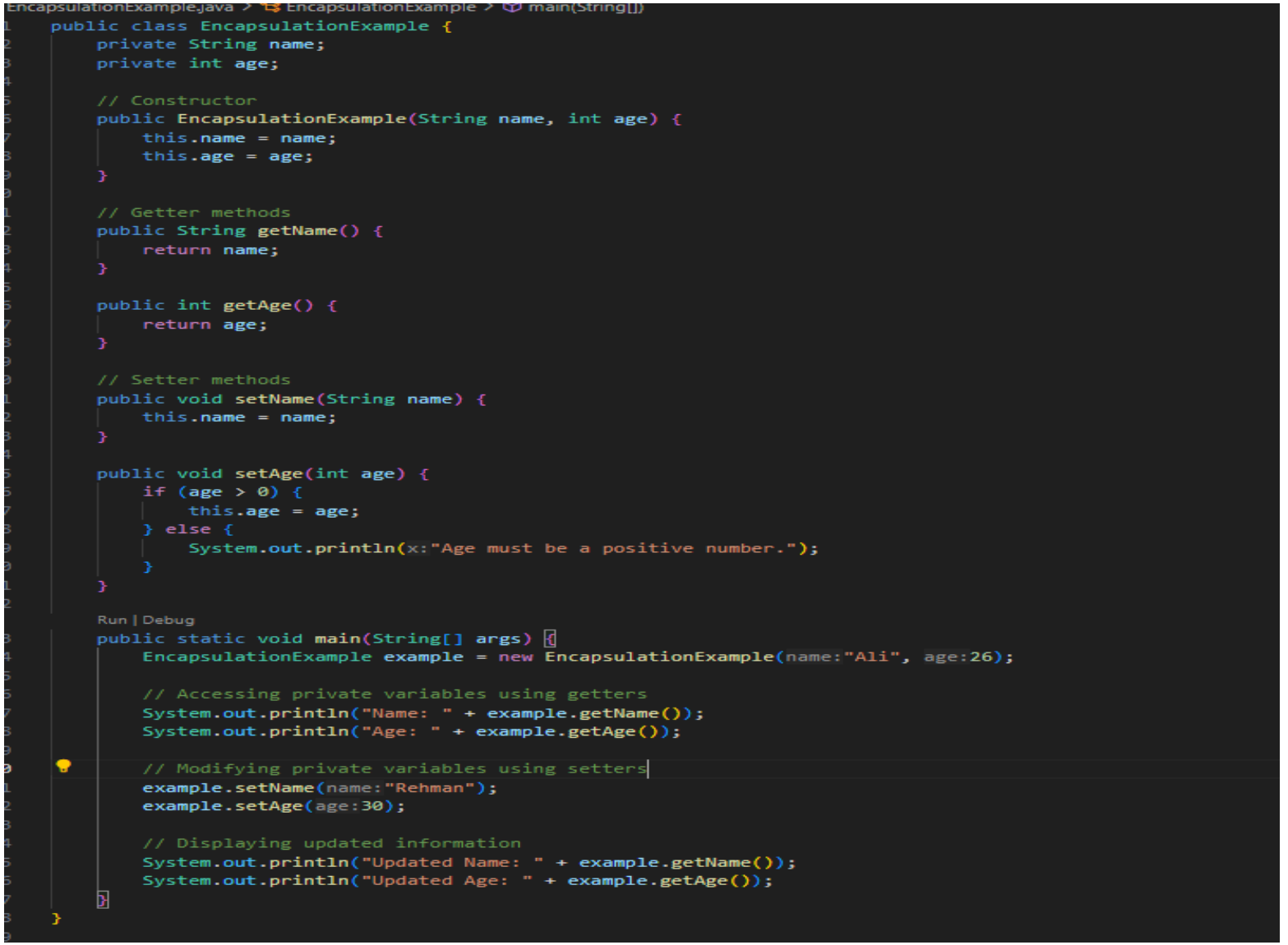
Encapsulation



School bag can keep our book,
pen, erasers, sharpner so on..

>>Swipe

In java



```java
public class EncapsulationExample {
    private String name;
    private int age;

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

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

    public int getAge() {
        return age;
    }

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

    public void setAge(int age) {
        if (age > 0) {
            this.age = age;
        } else {
            System.out.println(x:"Age must be a positive number.");
        }
    }

    Run | Debug
    public static void main(String[] args) {
        EncapsulationExample example = new EncapsulationExample(name:"Ali", age:26);

        // Accessing private variables using getters
        System.out.println("Name: " + example.getName());
        System.out.println("Age: " + example.getAge());

        // Modifying private variables using setters
        example.setName(name:"Rehman");
        example.setAge(age:30);

        // Displaying updated information
        System.out.println("Updated Name: " + example.getName());
        System.out.println("Updated Age: " + example.getAge());
    }
}
```

In Python..

```python
class EncapsulationExample:
    def __init__(self, name, age):
        self._name = name
        self._age = age

    # Getter methods
    def get_name(self):
        return self._name

    def get_age(self):
        return self._age

    # Setter methods
    def set_name(self, name):
        self._name = name

    def set_age(self, age):
        if age > 0:
            self._age = age
        else:
            print("Age must be a positive number.")

# Example usage
example = EncapsulationExample("Ali", 26)

# Accessing private variables using getters
print("Name:", example.get_name())
print("Age:", example.get_age())

# Modifying private variables using setters
example.set_name("Rehman")
example.set_age(30)

# Displaying updated information
print("Updated Name:", example.get_name())
print("Updated Age:", example.get_age())
```

```
Name: Ali
```

Why Encapsulation?

- Better control of class attributes and methods
- Class attributes can be made read-only (if you only use the get method), or write-only (if you only use the set method)
- Flexible: the programmer can change one part of the code without affecting other parts
- Increased security of data

Task…..!

- Task: Employee Information System
  - Design an Employee class with private attributes like employeeId, employeeName, and salary. Provide methods to access employee details and update the salary. Implement encapsulation to protect the integrity of employee data.
- Task: Student Enrollment System
  - Develop a Student class with private attributes such as studentId, studentName, and coursesEnrolled. Implement methods to display student information and add or remove courses. Apply encapsulation to control access to the student's data.
- Task: Product Inventory System
  - Create a Product class with private attributes such as productCode, productName, and quantityInStock. Utilize encapsulation to provide methods for updating stock, checking stock availability, and displaying product information.