by Kunal Sir

Assignment: Encapsulation

Part A: MCQ's

1. Which statement best defines encapsulation?
    - A. Binding one class to another
    - B. Wrapping data and methods into a single unit
    - C. Hiding logic using inheritance
    - D. Achieving polymorphism

    Answer: B. Wrapping data and methods into a single unit

2. What is the main purpose of encapsulation in Java?
    - A. Code reusability
    - B. Faster execution
    - C. Data hiding and security
    - D. Multiple inheritance

    Answer: C. Data hiding and security

3. Which access modifier is **most restrictive**?
    - A. public
    - B. protected
    - C. default
    - D. private

    Answer: D. private

4. If all instance variables of a class are declared `private`, the class is called:
    - A. Immutable class
    - B. Secure class
    - C. Tightly encapsulated class
    - D. Abstract class

    Answer: C. Tightly encapsulated class

5. Which of the following breaks encapsulation?
    - A. Private variables
    - B. Getter and setter methods
    - C. Public data members
    - D. Using access modifiers

    Answer: C. Public data members

6. Can encapsulation be achieved without using getters and setters?
    - A. No, never
    - B. Yes, by using public variables
    - C. Yes, by using private variables with controlled methods
    - D. Only using inheritance

    Answer: C. Yes, by using private variables with controlled methods

7. Why are instance variables usually kept `private`?
    - A. To improve performance
    - B. To prevent direct modification
    - C. To support polymorphism
    - D. To reduce memory usage

    Answer: B. To prevent direct modification

8. Which access modifier allows access **only within the same package**?
    - A. private
    - B. protected
    - C. public
    - D. default

    Answer: D. default

9. In encapsulation, validation logic is usually written in:
    - A. Constructors only
    - B. Getter methods
    - C. Setter methods
    - D. Main method

    Answer: C. Setter methods

10. What happens if a setter method is removed?
    - A. Variable becomes public
    - B. Variable becomes read-only
    - C. Variable becomes write-only
    - D. Program will not compile

    Answer: B. Variable becomes read-only

11. Which real-world example best represents encapsulation?
    - A. Library book
    - B. ATM machine
    - C. Classroom
    - D. Internet

    Answer: B. ATM machine

12. Which keyword is mandatory to achieve data hiding?
    - A. static
    - B. final
    - C. private
    - D. protected

    Answer: C. private

13. Can a class be encapsulated if variables are `protected`?
    - A. Yes, fully
    - B. Partially
    - C. No
    - D. Only in same package

    Answer: B. Partially

14. Which OOP concept is **directly supported** by encapsulation?
    - A. Abstraction
    - B. Data hiding
    - C. Inheritance
    - D. Polymorphism

    Answer: B. Data hiding

15. If a variable is private, who can access it?
    - A. Any class in same package
    - B. Child class
    - C. Same class only
    - D. Any class

    Answer: C. Same class only

16. What is the role of getters?
    - A. Modify data
    - B. Validate data
    - C. Read data safely
    - D. Initialize data

    Answer: C. Read data safely

17. Which access modifier gives maximum accessibility?
    - A. private
    - B. protected
    - C. default
    - D. public

    Answer: D. public

18. Encapsulation improves maintainability because:
    - A. Data is duplicated
    - B. Changes are localized
    - C. Code becomes longer
    - D. Execution becomes faster

    Answer: B. Changes are localized

19. Which scenario violates encapsulation rules?
    - A. Public getter, private variable
    - B. Private variable, no setter
    - C. Public variable without validation
    - D. Private variable with setter

    Answer: C. Public variable without validation

20. What will happen if we expose all variables as public?
    - A. Better security
    - B. No impact
    - C. Loss of control over data
    - D. Faster development

    Answer: C. Loss of control over data

Part B: Problem Statement

1. Bank Account

Problem Statement:
Create a class called `BankAccount`. The account balance should not be changed directly. The balance must be private. Money should be added or removed only using methods.

Student Task:
Use a private variable for balance and methods to deposit and withdraw money.

Sample Input:
Deposit = 5000
Withdraw = 2000

Sample Output:
Balance = 3000

```
class BankAccount {
    private int balance;

    public void deposit(int amount) {
        balance = balance + amount;
    }

    public void withdraw(int amount) {
        balance = balance - amount;
    }

    public int getBalance() {
        return balance;
    }

    public static void main(String[] args) {
        BankAccount b = new BankAccount();
        b.deposit(5000);
        b.withdraw(2000);
        System.out.println("Balance = " + b.getBalance());
    }
}
```

2. Student Marks

Problem Statement:
Create a class called `Student`. Marks should not be accessed directly. Marks must be between 0 and 100.

Student Task:
Keep marks private and set them using a method.

Sample Input:
Marks = 85

Sample Output:
Marks saved successfully

```
class Student {
    private int marks;

    public void setMarks(int m) {
        if (m >= 0 && m <= 100) {
            marks = m;
        }
    }

    public int getMarks() {
        return marks;
    }

    public static void main(String[] args) {
        Student s = new Student();
        s.setMarks(85);
        System.out.println("Marks = " + s.getMarks());
    }
}
```

---

3. Employee Salary

Problem Statement:
Create a class called `Employee`. Salary should not be changed directly.

Student Task:
Make salary private and update it using a method.

Sample Input:
Salary = 30000

Sample Output:
Salary updated

```
class Employee {
    private int salary;

    public void setSalary(int s) {
        if (s > 0) {
            salary = s;
        }
    }

    public int getSalary() {
        return salary;
    }

    public static void main(String[] args) {
        Employee e = new Employee();
        e.setSalary(30000);
        System.out.println("Salary = " + e.getSalary());
    }
}
```

---

4. ATM System

Problem Statement:
Create a class called `ATM`. The cash balance should not be accessed directly.

Student Task:
Keep balance private and allow withdrawal using a method.

Sample Input:
Withdraw = 1000

Sample Output:
Cash withdrawn successfully

```
class ATM {
    private int balance = 5000;

    public void withdraw(int amount) {
        if (amount <= balance) {
            balance = balance - amount;
            System.out.println("Cash withdrawn");
        } else {
            System.out.println("Insufficient balance");
        }
    }

    public static void main(String[] args) {
        ATM a = new ATM();
        a.withdraw(1000);
    }
}
```

---

5. Mobile Volume

Problem Statement:
Create a class called `Mobile`. Volume should stay between 0 and 100.

Student Task:
Make volume private and change it using methods.

Sample Input:
Increase volume by 20

Sample Output:
Current volume = 70

```
class Mobile {
    private int volume = 50;

    public void increaseVolume(int v) {
        if (volume + v <= 100) {
            volume = volume + v;
        }
    }

    public int getVolume() {
        return volume;
    }

    public static void main(String[] args) {
        Mobile m = new Mobile();
        m.increaseVolume(20);
        System.out.println("Volume = " + m.getVolume());
    }
}
```

---

6. Library Book

Problem Statement:
Create a class called `Book`. A book should be issued only once.

Student Task:
Keep book status private and update it using methods.

Sample Input:
Issue book

Sample Output:
Book issued

```
class Book {
    private boolean available = true;

    public void issueBook() {
        if (available) {
            available = false;
            System.out.println("Book issued");
        } else {
            System.out.println("Book not available");
        }
    }

    public static void main(String[] args) {
        Book b = new Book();
        b.issueBook();
    }
}
```

---

7. Shopping Cart

Problem Statement:
Create a class called `Cart`. Total amount should be calculated automatically.

Student Task:
Make total amount private and update it using methods.

Sample Input:
Add item price = 1200

Sample Output:
Total amount = 1200

```
class Cart {
    private int totalAmount;

    public void addItem(int price) {
        totalAmount = totalAmount + price;
    }

    public int getTotalAmount() {
        return totalAmount;
    }

    public static void main(String[] args) {
        Cart c = new Cart();
        c.addItem(1200);
        System.out.println("Total Amount = " + c.getTotalAmount());
    }
}
```

---

8. User Password

Problem Statement:
Create a class called `User`. Password should not be visible.

Student Task:
Keep password private and check it using a method.

Sample Input:
Enter password = abc@123

Sample Output:
Login successful

```
class User {
    private String password = "abc@123";

    public void checkPassword(String p) {
        if (password.equals(p)) {
            System.out.println("Login successful");
        } else {
            System.out.println("Wrong password");
        }
    }

    public static void main(String[] args) {
        User u = new User();
        u.checkPassword("abc@123");
    }
}
```

---

9. Vehicle Speed

Problem Statement:
Create a class called `Vehicle`. Speed should not cross a limit.

Student Task:
Make speed private and change it using methods.

Sample Input:
Increase speed by 30

Sample Output:
Speed = 90

```
class Vehicle {
    private int speed = 60;

    public void increaseSpeed(int s) {
        if (speed + s <= 120) {
            speed = speed + s;
        }
    }

    public int getSpeed() {
        return speed;
    }

    public static void main(String[] args) {
        Vehicle v = new Vehicle();
        v.increaseSpeed(30);
        System.out.println("Speed = " + v.getSpeed());
    }
}
```

---

10. Online Exam

Problem Statement:
Create a class called `Exam`. Marks should be updated only after checking answers.

Student Task:
Keep marks private and update them using a method.

Sample Input:
Marks = 78

Sample Output:
Marks saved

```
class Exam {
    private int marks;

    public void setMarks(int m) {
        marks = m;
    }

    public int getMarks() {
        return marks;
    }

    public static void main(String[] args) {
        Exam e = new Exam();
        e.setMarks(78);
        System.out.println("Marks = " + e.getMarks());
    }
}
```
