**Experiment No-9**

**Experiment Name:** Demonstration of Encapsulation in Java

**Task No-1:** Student Class with Encapsulation - Create a Student class with private fields: name, id, and cgpa. Use setter methods to set values and getter methods to display them.

**Objective:**

- To understand and implement encapsulation in Java using private variables and public setter/getter - methods.
- To demonstrate how data can be protected and accessed safely in an object-oriented program.

**Problem analysis:**

- Input Variables: name (String), id (String), cgpa (double)
- Processing:
    - Create Student class with private fields.
    - Use setter methods to assign values.
    - Use getter methods to retrieve and display values.
- Output Variables: Student name, ID, and CGPA.
- Concept Used: Encapsulation (data hiding + access methods).

**Algorithm:**

1. START
2. Define a class named Student
    - Declare private fields: name (String), id (String), cgpa (double)
    - Define public setter methods:
        - setName(String n): assign n to name
        - setId(String i): assign i to id
        - setCgpa(double c): assign c to cgpa
    - Define public getter methods:
        - getName(): return name
        - getId(): return id
        - getCgpa(): return cgpa
3. Define a public class TestCode
    - In main():
        a. Create Student object s1
        b. Use setters to assign: name = "Atik", id = "E25046", cgpa = 0.00
        c. Use getters to retrieve and print all values
4. END

**Source code:**

```
class Student {
    private String name;
    private String id;
    private double cgpa;
    public void setName(String n) {
        name = n;
    }
    public void setId(String i) {
        id = i;
    }
    public void setCgpa(double c) {
        cgpa = c;
    }
    public String getName() {
        return name;
    }
    public String getId() {
        return id;
    }
    public double getCgpa() {
        return cgpa;
    }
}
public class StudentInfoemation {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.setName("Atik");
        s1.setId("E251046");
        s1.setCgpa(3.00);
        System.out.println("Student Info:");
        System.out.println("Name: " + s1.getName());
        System.out.println("ID: " + s1.getId());
        System.out.println("CGPA: " + s1.getCgpa());
    }
}
```

**Output:**

```
PROBLEMS 3    OUTPUT    DEBUG CONSOLE    TERMINAL    PORT

Microsoft Windows [Version 10.0.19045.6466]
(c) Microsoft Corporation. All rights reserved.

D:\java lab report VS code> cmd /C "C:\Users\Dell\.
64720 -XX:+ShowCodeDetailsInExceptionMessages -cp "
va lab report VS code_a595819c\bin" StudentInfoemat
Student Info:
Name: Atik
ID: E251046
CGPA: 3.0

D:\java lab report VS code>
```

**Task No-2:** BankAccount Class with Encapsulation- Create a BankAccount class with private fields: accountHolder, accountNumber, and balance. Use methods to set values and show account info securely.

**Objective:**

- To demonstrate encapsulation in Java by hiding bank account details using private variables and accessing them through public methods.

**Problem Analysis:**

- Input Variables: accountHolder (String), accountNumber (String), balance (double)
- Processing:
  - Create BankAccount class with private fields
  - Use setter methods to assign account details
  - Use a method to display account info securely
- Output Variables: Account Holder, Account Number, Balance
- Concept Used: Encapsulation (data hiding + access through methods)

**Algorithm:**

1. Start
2. Create a class named BankAccount
3. Declare private fields: accountHolder, accountNumber, balance
4. Create setter methods to assign values
5. Create a method to display account information
6. In the main class, create a BankAccount object
7. Use setters to assign account details
8. Call the display method to show details
9. End

**Source code:**

```
class BankAccount {
    private String accountHolder;
    private String accountNumber;
    private String bankName;
    private String branchName;
    private double balance;
    public void setAccountHolder(String h) {
        accountHolder = h;
    }
    public void setAccountNumber(String num) {
        accountNumber = num;
    }
    public void setBankName(String bank) {
        bankName = bank;
    }
    public void setBranchName(String branch) {
        branchName = branch;
    }
    public void setBalance(double b) {
        balance = b;
    }
    public void showAccountDetails() {
```

```
        System.out.println("------------------------------------------------------------");
        System.out.println("            ACCOUNT DETAILS ");
        System.out.println("------------------------------------------------------------");
        System.out.println("Account Holder   : " + accountHolder);
        System.out.println("Account Number   : " + accountNumber);
        System.out.println("Bank             : " + bankName);
        System.out.println("Branch           : " + branchName);
        System.out.println();
        System.out.printf("Current Balance  : %, .2f BDT\n\n", balance);
    }
    public void showTransaction(String date, String txnId, String type, double amount) {
        System.out.println("------------------------------------------------------------");
        System.out.println("            TRANSACTION DETAILS ");
        System.out.println("------------------------------------------------------------");
        System.out.printf("| %s | %s | %s | %, .2f BDT |\n", date, txnId, type, amount);
    }
}
public class BankDetails {
    public static void main(String[] args) {
        BankAccount acc = new BankAccount();
        acc.setAccountHolder("Md. Atikur Rahaman");
        acc.setAccountNumber("2025 1225 1046");
        acc.setBankName("Islami Bank Bangladesh PLC");
        acc.setBranchName("Chawkbazar Branch");
        acc.setBalance(12540.00);
        acc.showAccountDetails();
        acc.showTransaction("06-Dec-2025", "IBBL-TXN-98421356", "Deposit", 12540.00);
    }
}
```

**Output:**

```
D:\java lab report VS code>java BankDetails
------------------------------------------------------------
            ACCOUNT DETAILS
------------------------------------------------------------
Account Holder   : Md. Atikur Rahaman
Account Number   : 2025 1225 1046
Bank             : Islami Bank Bangladesh PLC
Branch           : Chawkbazar Branch

Current Balance  :  12,540.00 BDT

------------------------------------------------------------
            TRANSACTION DETAILS
------------------------------------------------------------
| 06-Dec-2025 | IBBL-TXN-98421356 | Deposit |  12,540.00 BDT |

D:\java lab report VS code>
```

**Discussion:**

Encapsulation is a fundamental concept of Object-Oriented Programming that involves hiding data (declaring fields as private) and providing controlled access through public methods (setters and getters). In Task 1, the Student class demonstrates encapsulation by protecting the student's name, id, and cgpa, allowing safe assignment and retrieval via methods. Similarly, in Task 2, the BankAccount class secures sensitive banking information such as accountHolder, accountNumber, and balance. Using setter methods ensures values are assigned safely, while display methods or getters allow controlled viewing of information. Both tasks highlight the benefits of encapsulation: data security, prevention of unauthorized access, ease of maintenance, and flexible future enhancements, such as adding validation or formatting for output (like bank statements).