ASSIGNMENT

B. LIKHITHA

Assignment 1: Student Encapsulation

Create a Student class:

• Private variables: id, name

• Public getters & setters

• Display details using getter methods.

Concepts: Data hiding, getter/setter

Student.java

Share Run Output

```
class Student {
    private int id;
    private String name;
    public void setId(int id) {
        this.id = id;}
    public void setName(String name) {
        this.name = name; }
    public int getId() {
        return id; }
    public String getName() {
        return name;}
    public static void main(String[] args) {
        Student s = new Student();
        s.setId(101);
        s.setName("Likhitha");
        System.out.println("ID: " + s.getId());
        System.out.println("Name: " + s.getName()); }}
```

```
ID: 101
Name: Likhitha

=== Code Execution Successful ===
```

Assignment 2: Bank Account

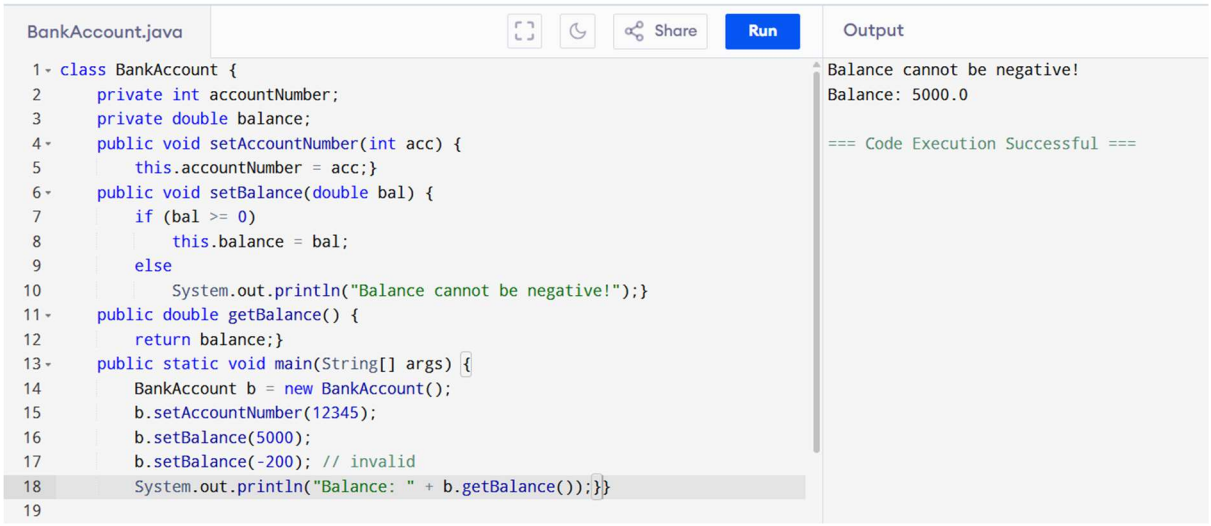
Create BankAccount class:

• Private: accountNumber, balance

• Setter validates balance (no negative value)

• Getter returns balance.

Concepts: Validation logic

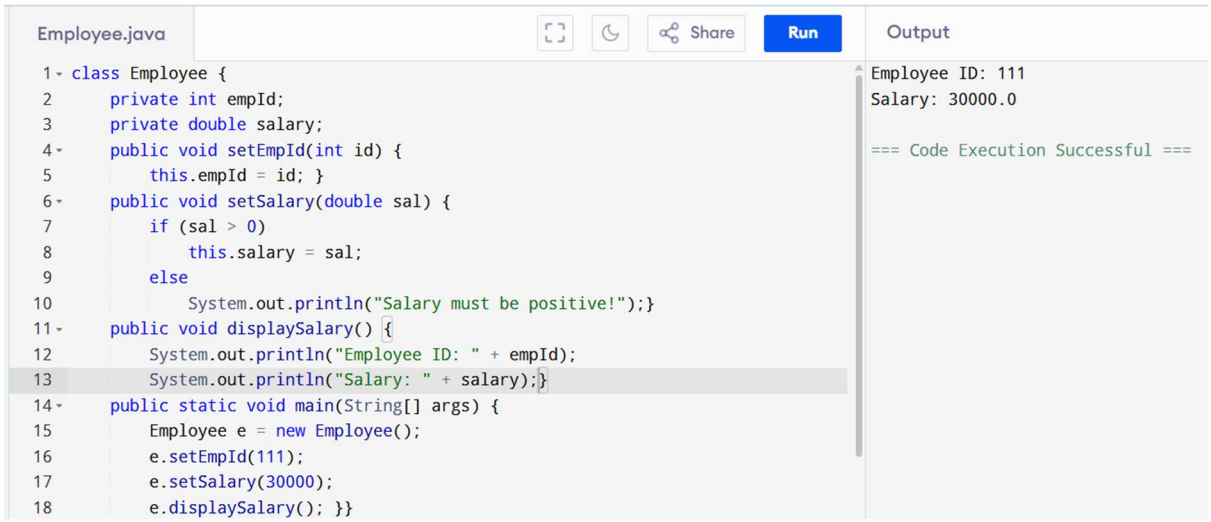


Assignment 3: Employee Salary

Create Employee class:

- Private: empId, salary
- Setter restricts salary > 0
- Method displaySalary()

Concepts: Controlled access



Assignment 4: Product Price

Create Product class:

- Private: price
- Setter allows price only between 100 – 100000

Concepts: Business rule encapsulation

Product.java Share Run Output

```
class Product {
    private double price;
    public void setPrice(double price) {
        if (price >= 100 && price <= 100000)
            this.price = price;
        else
            System.out.println("Price must be between 100 and 100000!");
    }
    public double getPrice() {
        return price;
    }
    public static void main(String[] args) {
        Product p = new Product();
        p.setPrice(500);
        System.out.println("Price: " + p.getPrice());
    }
}
```

```
Price: 500.0

=== Code Execution Successful ===
```

Assignment 5: Login Credentials

Create User class:

- Private: email, password
- Password setter validates length ≥ 8

Concepts: Security with encapsulation

User.java Share Run Output

```
class User {
    private String email;
    private String password;
    public void setEmail(String email) {
        this.email = email;
    }
    public void setPassword(String password) {
        if (password.length() >= 8)
            this.password = password;
        else
            System.out.println("Password must be at least 8 characters!");
    }
    public static void main(String[] args) {
        User u = new User();
        u.setEmail("abc@gmail.com");
        u.setPassword("12345");    // invalid
        u.setPassword("12345678"); // valid
    }
}
```

```
Password must be at least 8 characters!

=== Code Execution Successful ===
```

Assignment 6: Customer Profile

Create Customer class:

- Private: name, age
- Setter restricts age ≥ 18
- Getter returns formatted data

Concepts: Validation inside setters

Customer.java

Share Run

Output

```
class Customer {
    private String name;
    private int age;
    public void setName(String name) {
        this.name = name;  }
    public void setAge(int age) {
        if (age >= 18)
            this.age = age;
        else
            System.out.println("Age must be 18 or above!");
    }
    public String getProfile() {
        return "Name: " + name + ", Age: " + age;
    }
    public static void main(String[] args) {
        Customer c = new Customer();
        c.setName("Likhitha");
        c.setAge(17);
        System.out.println(c.getProfile()); }}
```

```
Age must be 18 or above!
Name: Likhitha, Age: 0

=== Code Execution Successful ===
```

Assignment 7: Mobile Phone

Create Mobile class:

• Private: brand, price

• Getter returns price with GST

Concepts: Logic in getter

Mobile.java

Share Run

Output

```
class Mobile {
    private String brand;
    private double price;
    public void setBrand(String brand) {
        this.brand = brand;
    }
    public void setPrice(double price) {
        this.price = price;
    }
    public double getPriceWithGST() {
        return price + (price * 0.18); // 18% GST
    }
    public static void main(String[] args) {
        Mobile m = new Mobile();
        m.setBrand("Samsung");
        m.setPrice(10000);
        System.out.println("Price with GST: " + m.getPriceWithGST());
    }
}
```

```
Price with GST: 11800.0

=== Code Execution Successful ===
```

Assignment 8: ATM System

Create ATMAccount class:

• Private: pin, balance

• setPin() validates 4 digits

• withdraw(amount) checks balance

Concepts: Secure operations

ATMAccount.java

Share Run

```java
class ATMAccount {
    private int pin;
    private double balance;
    public void setPin(int pin) {
        if (String.valueOf(pin).length() == 4)
            this.pin = pin;
        else
            System.out.println("PIN must be 4 digits!");
    }
    public void deposit(double amount) {
        balance += amount;
    }
    public void withdraw(double amount) {
        if (amount <= balance)
            balance -= amount;
        else
            System.out.println("Insufficient balance!");
    }
```

ATMAccount.java

Share Run Output

```java
    public double getBalance() {
        return balance;
    }

    public static void main(String[] args) {
        ATMAccount a = new ATMAccount();
        a.setPin(1234);
        a.deposit(5000);
        a.withdraw(2000);

        System.out.println("Balance: " + a.getBalance());
    }
}
```

```
Balance: 3000.0

=== Code Execution Successful ===
```

Assignment 9: College Admission

Create Admission class:

• Private: marks

• Setter assigns grade internally

Concepts: Internal processing

Admission.java

Share Run Output

```java
class Admission {
    private int marks;
    private String grade;
    public void setMarks(int marks) {
        this.marks = marks;
        if (marks >= 90) grade = "A";
        else if (marks >= 75) grade = "B";
        else if (marks >= 50) grade = "C";
        else grade = "D";
    }
    public String getGrade() {
        return grade;
    }
    public static void main(String[] args) {
        Admission a = new Admission();
        a.setMarks(88);
        System.out.println("Grade: " + a.getGrade());
    }
}
```

```
Grade: B

=== Code Execution Successful ===
```

Assignment 10: Insurance Policy

Create Policy class:

- Private: policyId, premium
- Premium calculated internally based on age

Concepts: Business rule hiding

Policy.java

Share Run Output

```java
class Policy {
    private int policyId;
    private double premium;
    public void setPolicy(int id, int age) {
        this.policyId = id;
        if (age < 30)
            premium = 3000;
        else if (age < 45)
            premium = 4500;
        else
            premium = 6000; }
    public double getPremium() {
        return premium;
    }
    public static void main(String[] args) {
        Policy p = new Policy();
        p.setPolicy(101, 25);
        System.out.println("Premium: " + p.getPremium());
    }}
```

```
Premium: 3000.0

=== Code Execution Successful ===
```