ASSIGNMENT

B. LIKHITHA

Assignment 1: Student Class

Create a Student class with:

• Variables: id, name

• Method: displayDetails()

• Create object in main() and print details

Concepts: Class, Object

Student.java

```java
class Student {
    int id;
    String name;

    void displayDetails() {
        System.out.println("Student ID: " + id);
        System.out.println("Student Name: " + name);
    }

    public static void main(String[] args) {
        Student s = new Student();
        s.id = 101;
        s.name = "Likhitha";
        s.displayDetails();
    }
}
```

Output

```
Student ID: 101
Student Name: Likhitha

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

Assignment 2: Employee Salary

Create an Employee class:

• Variables: empId, empName, salary

• Initialize values using **default constructor**

• Display employee details.

Concepts: Default Constructor

Employee.java

```java
class Employee {
    int empId;
    String empName;
    double salary;

    Employee() {
        empId = 1;
        empName = "John";
        salary = 25000;
    }
    void display() {
        System.out.println(empId + " " + empName + " " + salary);
    }
    public static void main(String[] args) {
        Employee e = new Employee();
        e.display();
    }
}
```

Output

```
1 John 25000.0

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

Assignment 3: Car Information

Create a Car class:

• Variables: brand, model, price

• Use **parameterized constructor**

• Create 2 objects with different data.

Concepts: Parameterized Constructor

Car.java

Share Run

Output

```
class Car {
    String brand;
    String model;
    double price;
    Car(String b, String m, double p) {
        brand = b;
        model = m;
        price = p;
    }
    void show() {
        System.out.println(brand + " " + model + " Rs." + price);
    }
    public static void main(String[] args) {
        Car c1 = new Car("BMW", "X5", 9000000);
        Car c2 = new Car("Audi", "A6", 6000000);

        c1.show();
        c2.show();
    }
}
```

```
BMW X5 Rs.9000000.0
Audi A6 Rs.6000000.0

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

Assignment 4: Book Details

Create a Book class:

• Default constructor sets values

• Parameterized constructor sets custom values

• Display details of both.

Concepts: Constructor overloading.

Book.java

Share Run

Output

```
class Book {
    String title, author;
    int price;
    Book() { // Default constructor
        title = "Unknown";
        author = "Unknown";
        price = 0;
    }    Book(String t, String a, int p) { // Parameterized
        title = t;
        author = a;
        price = p; }
    void display() {
        System.out.println(title + " - " + author + " - Rs." + price);
    }
    public static void main(String[] args) {
        Book b1 = new Book();
        Book b2 = new Book("Java Basics", "James", 499);
        b1.display();
        b2.display();
    }}
```

```
Unknown - Unknown - Rs.0
Java Basics - James - Rs.499

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

Assignment 5: Bank Account

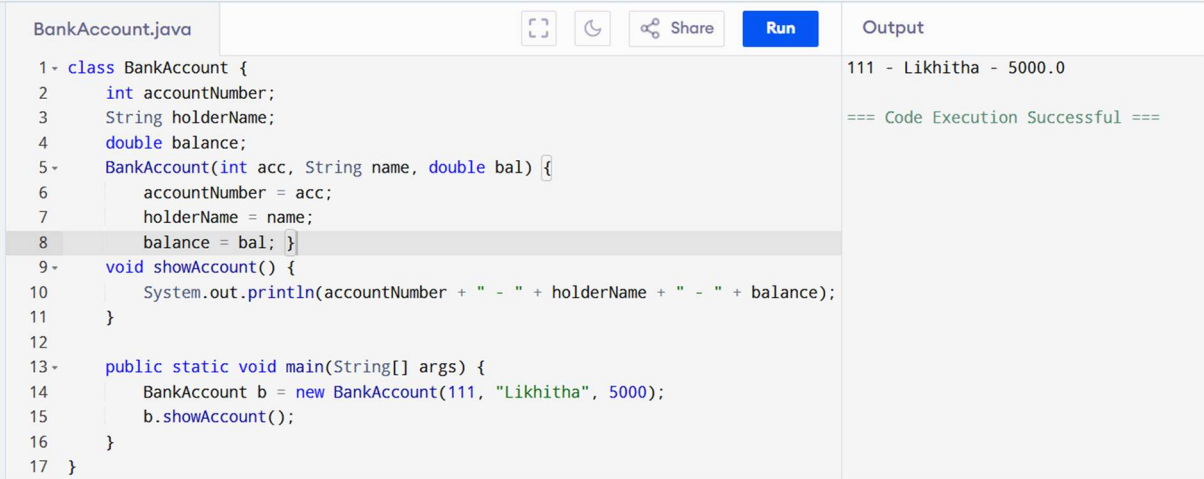
Create BankAccount class:

• Variables: accountNumber, holderName, balance

• Constructor to initialize data

• Method showAccount() prints details.

Concepts: Object initialization



Assignment 6: Product Management

Create a Product class:

• Variables: productId, productName, price

• Constructor uses this keyword

• Display product info.

Concepts: this keyword

Assignment 7: User Login System

Create User class:

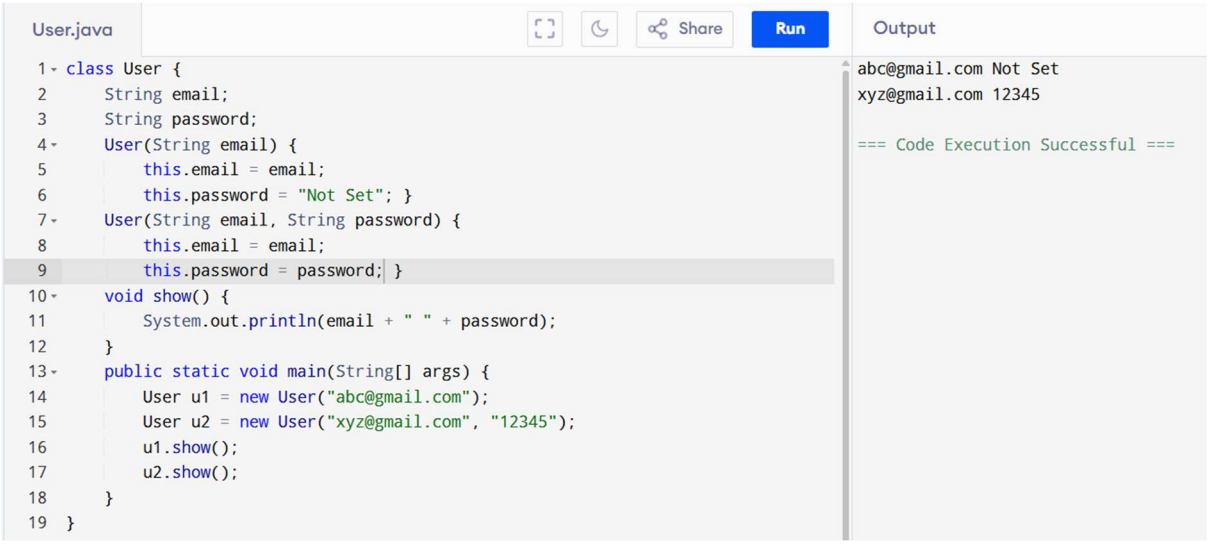
• Variables: email, password

• One constructor takes email only

• Another takes email & password

• Display user credentials.

Concepts: Constructor overloading



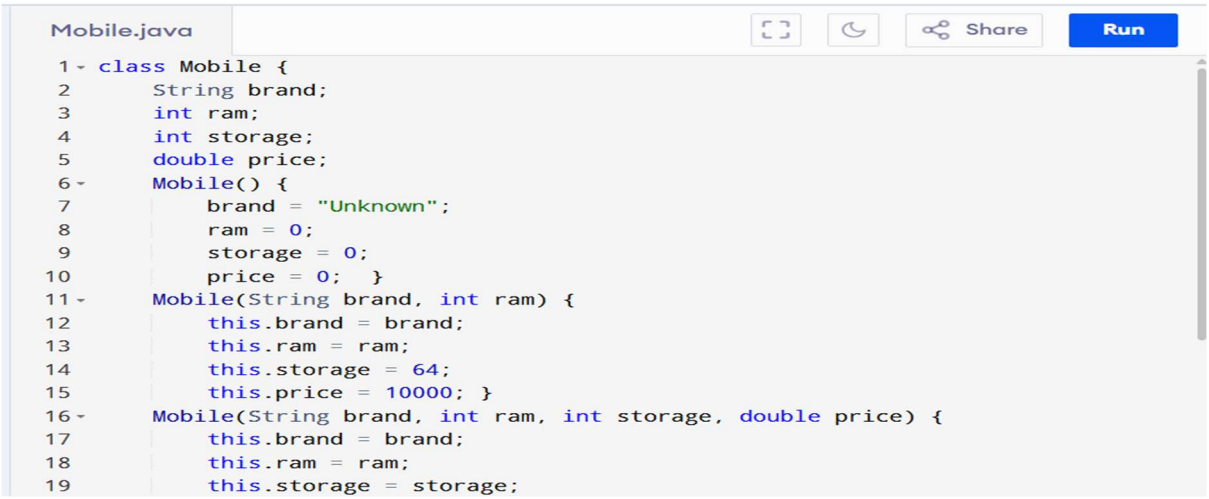
Assignment 8: Mobile Store

Create Mobile class:

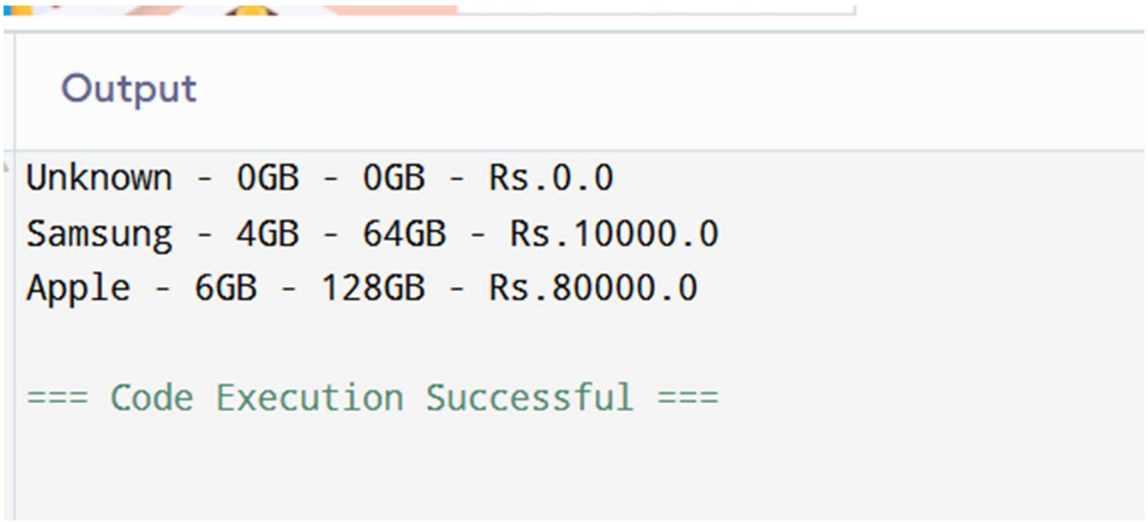
• Fields: brand, ram, storage, price

• Create multiple objects using different constructors

• Print mobile configuration.**Concepts:** Object reuse & overloading



```
        this.price = price; }
    void display() {
        System.out.println(brand + " - " + ram + "GB - " + storage + "GB - Rs."
            + price); }
    public static void main(String[] args) {
        Mobile m1 = new Mobile();
        Mobile m2 = new Mobile("Samsung", 4);
        Mobile m3 = new Mobile("Apple", 6, 128, 80000);
        m1.display();
        m2.display();
        m3.display();
    }}
```



Assignment 9: Library Book System

Create LibraryBook class:

• Fields: bookId, title, author

• Constructor assigns values

• Method isAvailable() returns true/false

Concepts: Object behavior

Assignment 10: College Admission

Create StudentAdmission class:

• Default constructor → general admission

• Parameterized constructor → merit admission

• Display admission type.

Concepts: Constructor logic

StudentAdmission.java — Share — Run — Output

```
class StudentAdmission {
    String admissionType;
    StudentAdmission() {
        admissionType = "General Admission";
    }
    StudentAdmission(String type) {
        admissionType = type;
    }
    void show() {
        System.out.println("Admission Type: " + admissionType);
    }
    public static void main(String[] args) {
        StudentAdmission s1 = new StudentAdmission();
        StudentAdmission s2 = new StudentAdmission("Merit Admission");

        s1.show();
        s2.show();
    }
}
```

```
Admission Type: General Admission
Admission Type: Merit Admission

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

Assignment 11: Constructor Chaining

Create Person class:

• Constructor with name

• Constructor with name & age

• Use this() to chain constructors.

Concepts: Constructor chaining

Person.java — Share — Run — Output

```
class Person {
    String name;
    int age;
    Person(String name) {
        this.name = name;
        this.age = 0;
    }
    Person(String name, int age) {
        this(name);
        this.age = age;
    }
    void show() {
        System.out.println(name + " " + age);
    }
    public static void main(String[] args) {
        Person p = new Person("Likhitha", 20);
        p.show();
    }
}
```

```
Likhitha 20

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

Assignment 12: E-Commerce Order

Create Order class:

• Fields: orderId, customerName, amount

• Multiple constructors

• Calculate tax inside constructor.

Concepts: Business logic in constructors

Order.java

Share Run

```java
class Order {
    int orderId;
    String customerName;
    double amount;
    double tax;
    Order(int orderId) {
        this.orderId = orderId;
        this.customerName = "Unknown";
        this.amount = 0;
        this.tax = 0;
    }
```

Order.java

Share Run

Output

```java
    Order(int orderId, String name, double amount) {
        this.orderId = orderId;
        this.customerName = name;
        this.amount = amount;
        this.tax = amount * 0.18;  // 18% GST
    }
    void display() {
        System.out.println(orderId + " " + customerName + " Amount: " + amount
            + " Tax: " + tax);
    }
    public static void main(String[] args) {
        Order o1 = new Order(101);
        Order o2 = new Order(102, "Likhitha", 5000);
        o1.display();
        o2.display();
    }}
```

```
101 Unknown Amount: 0.0 Tax: 0.0
102 Likhitha Amount: 5000.0 Tax: 900.0

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