

# 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



The screenshot shows a Java IDE with a file named Student.java. The code defines a Student class with two attributes, id and name, and a displayDetails() method. The main method creates a Student object, initializes its attributes, and calls displayDetails(). The output window shows the execution results: Student ID: 101 and Student Name: Likhitha, followed by a success message.

```
Student.java
1- class Student {
2-     int id;
3-     String name;
4-
5-     void displayDetails() {
6-         System.out.println("Student ID: " + id);
7-         System.out.println("Student Name: " + name);
8-     }
9-
10-    public static void main(String[] args) {
11-        Student s = new Student();
12-        s.id = 101;
13-        s.name = "Likhitha";
14-        s.displayDetails();
15-    }
16- }
```

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



The screenshot shows a Java IDE with a file named Employee.java. The code defines an Employee class with three attributes: empId (int), empName (String), and salary (double). It includes a default constructor that initializes these attributes with values 1, "John", and 25000.0 respectively. A display() method is also defined. The main method creates an Employee object using the default constructor and calls the display() method. The output window shows the execution results: 1 John 25000.0, followed by a success message.

```
Employee.java
1- class Employee {
2-     int empId;
3-     String empName;
4-     double salary;
5-
6-     Employee() {
7-         empId = 1;
8-         empName = "John";
9-         salary = 25000;
10-    }
11-    void display() {
12-        System.out.println(empId + " " + empName + " " + salary);
13-    }
14-    public static void main(String[] args) {
15-        Employee e = new Employee();
16-        e.display();
17-    }
18- }
19- }
```

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
1- class Car {
2   String brand;
3   String model;
4   double price;
5- Car(String b, String m, double p) {
6   brand = b;
7   model = m;
8   price = p;
9 }
10- void show() {
11   System.out.println(brand + " " + model + " Rs." + price);
12 }
13- public static void main(String[] args) {
14   Car c1 = new Car("BMW", "X5", 9000000);
15   Car c2 = new Car("Audi", "A6", 6000000);
16
17   c1.show();
18   c2.show();
19 }
20 }
```

Output

```
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
1- class Book {
2   String title, author;
3   int price;
4- Book() { // Default constructor
5   title = "Unknown";
6   author = "Unknown";
7   price = 0;
8- } Book(String t, String a, int p) { // Parameterized
9   title = t;
10  author = a;
11  price = p; }
12- void display() {
13   System.out.println(title + " - " + author + " - Rs." + price);
14 }
15- public static void main(String[] args) {
16   Book b1 = new Book();
17   Book b2 = new Book("Java Basics", "James", 499);
18   b1.display();
19   b2.display();
20 }}
```

Output

```
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

<pre>BankAccount.java 1- class BankAccount { 2-     int accountNumber; 3-     String holderName; 4-     double balance; 5-     BankAccount(int acc, String name, double bal) { 6-         accountNumber = acc; 7-         holderName = name; 8-         balance = bal; } 9-     void showAccount() { 10-         System.out.println(accountNumber + " - " + holderName + " - " + balance); 11-     } 12- 13-     public static void main(String[] args) { 14-         BankAccount b = new BankAccount(111, "Likhitha", 5000); 15-         b.showAccount(); 16-     } 17- }</pre>	<p>Output</p> <pre>111 - Likhitha - 5000.0 === Code Execution Successful ===</pre>
--	--

## Assignment 6: Product Management

Create a Product class:

- Variables: productId, productName, price
- Constructor uses this keyword
- Display product info.

**Concepts:** this keyword

<pre>Product.java 1- class Product { 2-     int productId; 3-     String productName; 4-     double price; 5- 6-     Product(int productId, String productName, double price) { 7-         this.productId = productId; 8-         this.productName = productName; 9-         this.price = price; 10-     } 11- 12-     void show() { 13-         System.out.println(productId + " " + productName + " " + price); 14-     } 15- 16-     public static void main(String[] args) { 17-         Product p = new Product(10, "Laptop", 55000); 18-         p.show(); 19-     } 20- }</pre>	<p>Output</p> <pre>10 Laptop 55000.0 === Code Execution Successful ===</pre>
--	--

## 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

```
User.java
1- class User {
2     String email;
3     String password;
4-     User(String email) {
5         this.email = email;
6         this.password = "Not Set"; }
7-     User(String email, String password) {
8         this.email = email;
9         this.password = password; }
10-    void show() {
11        System.out.println(email + " " + password);
12    }
13-    public static void main(String[] args) {
14        User u1 = new User("abc@gmail.com");
15        User u2 = new User("xyz@gmail.com", "12345");
16        u1.show();
17        u2.show();
18    }
19 }
```

Output

```
abc@gmail.com Not Set
xyz@gmail.com 12345

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

## 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

```
Mobile.java
1- class Mobile {
2     String brand;
3     int ram;
4     int storage;
5     double price;
6-     Mobile() {
7         brand = "Unknown";
8         ram = 0;
9         storage = 0;
10        price = 0; }
11-     Mobile(String brand, int ram) {
12         this.brand = brand;
13         this.ram = ram;
14         this.storage = 64;
15         this.price = 10000; }
16-     Mobile(String brand, int ram, int storage, double price) {
17         this.brand = brand;
18         this.ram = ram;
19         this.storage = storage;
```

```

20     this.price = price; }
21 void display() {
22     System.out.println(brand + " - " + ram + "GB - " + storage + "GB - Rs."
        + price); }
23 public static void main(String[] args) {
24     Mobile m1 = new Mobile();
25     Mobile m2 = new Mobile("Samsung", 4);
26     Mobile m3 = new Mobile("Apple", 6, 128, 80000);
27     m1.display();
28     m2.display();
29     m3.display();
30 }

```

## Output

```

Unknown - 0GB - 0GB - Rs.0.0
Samsung - 4GB - 64GB - Rs.10000.0
Apple - 6GB - 128GB - Rs.80000.0

```

=== Code Execution Successful ===

## Assignment 9: Library Book System

Create LibraryBook class:

- Fields: bookId, title, author
- Constructor assigns values
- Method isAvailable() returns true/false

**Concepts:** Object behavior

LibraryBook.java	Run	Output
<pre> 1- class LibraryBook { 2-     int bookId; 3-     String title; 4-     String author; 5-     LibraryBook(int id, String t, String a) { 6-         bookId = id; 7-         title = t; 8-         author = a; 9-     } 10-     boolean isAvailable() { 11-         return true; // For now, assume available 12-     } 13-     public static void main(String[] args) { 14-         LibraryBook b = new LibraryBook(1, "Java", "James"); 15-         System.out.println("Available: " + b.isAvailable()); 16-     } 17- } </pre>	Run	<pre> Available: true  === Code Execution Successful === </pre>

## Assignment 10: College Admission

Create StudentAdmission class:

- Default constructor → general admission
- Parameterized constructor → merit admission
- Display admission type.

**Concepts:** Constructor logic

```
StudentAdmission.java
1- class StudentAdmission {
2-     String admissionType;
3-     StudentAdmission() {
4-         admissionType = "General Admission";
5-     }
6-     StudentAdmission(String type) {
7-         admissionType = type;
8-     }
9-     void show() {
10-         System.out.println("Admission Type: " + admissionType);
11-     }
12-     public static void main(String[] args) {
13-         StudentAdmission s1 = new StudentAdmission();
14-         StudentAdmission s2 = new StudentAdmission("Merit Admission");
15-
16-         s1.show();
17-         s2.show();
18-     }
19- }
```

Output

```
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
1- class Person {
2-     String name;
3-     int age;
4-     Person(String name) {
5-         this.name = name;
6-         this.age = 0;
7-     }
8-     Person(String name, int age) {
9-         this(name);
10-         this.age = age;
11-     }
12-     void show() {
13-         System.out.println(name + " " + age);
14-     }
15-     public static void main(String[] args) {
16-         Person p = new Person("Likhitha", 20);
17-         p.show();
18-     }
19- }
```

Output

```
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

```
1 class Order {
2     int orderId;
3     String customerName;
4     double amount;
5     double tax;
6     Order(int orderId) {
7         this.orderId = orderId;
8         this.customerName = "Unknown";
9         this.amount = 0;
10        this.tax = 0;
11    }
}
```

Order.java

Share

Run

```
12 Order(int orderId, String name, double amount) {
13     this.orderId = orderId;
14     this.customerName = name;
15     this.amount = amount;
16     this.tax = amount * 0.18; // 18% GST
17 }
18 void display() {
19     System.out.println(orderId + " " + customerName + " Amount: " + amount
20         + " Tax: " + tax);
21 }
22 public static void main(String[] args) {
23     Order o1 = new Order(101);
24     Order o2 = new Order(102, "Likhitha", 5000);
25     o1.display();
26     o2.display();
27 }
```

Output

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

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