

# ASSIGNMENT

**B. LIKHITHA**

## Assignment 1: Calculator Overloading

Create Calculator class:

- add(int a, int b)
- add(int a, int b, int c)
- add(double a, double b)

Concepts: Compile-time polymorphism



```
Calculator.java
1- class Calculator {
2-     int add(int a, int b) {
3         return a + b;
4     }
5-     int add(int a, int b, int c) {
6         return a + b + c;
7     }
8-     double add(double a, double b) {
9         return a + b;
10    }
11-     public static void main(String[] args) {
12         Calculator c = new Calculator();
13
14         System.out.println(c.add(5, 10));
15         System.out.println(c.add(3, 4, 5));
16         System.out.println(c.add(2.5, 3.7));
17     }
18 }
19
```

Output

```
15
12
6.2

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

## Assignment 2: Area Calculator

Create Area class:

- area(int side) → square
- area(int length, int breadth) → rectangle
- area(double radius) → circle

Concepts: Method signature change

```
Area.java
1- class Area {
2-     int area(int side) {           // square
3-         return side * side;
4-     }
5-     int area(int length, int breadth) { // rectangle
6-         return length * breadth;
7-     }
8-     double area(double radius) {     // circle
9-         return 3.14 * radius * radius;
10-    }
11-    public static void main(String[] args) {
12-        Area a = new Area();
13-        System.out.println("Square: " + a.area(5));
14-        System.out.println("Rectangle: " + a.area(4, 6));
15-        System.out.println("Circle: " + a.area(3.5));
16-    }
17- }
```

Output

```
Square: 25
Rectangle: 24
Circle: 38.465

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

## Assignment 3: Print Data

Create Printer class:

- print(int)
- print(String)
- print(int, String)

Concepts: Overloaded methods

```
Printer.java
1- class Printer {
2-     void print(int num) {
3-         System.out.println("Int: " + num);
4-     }
5-     void print(String text) {
6-         System.out.println("String: " + text);
7-     }
8-     void print(int num, String text) {
9-         System.out.println("Int & String: " + num + ", " + text);
10-    }
11-    public static void main(String[] args) {
12-        Printer p = new Printer();
13-        p.print(10);
14-        p.print("Hello");
15-        p.print(20, "World");
16-    }
17- }
```

Output

```
Int: 10
String: Hello
Int & String: 20, World

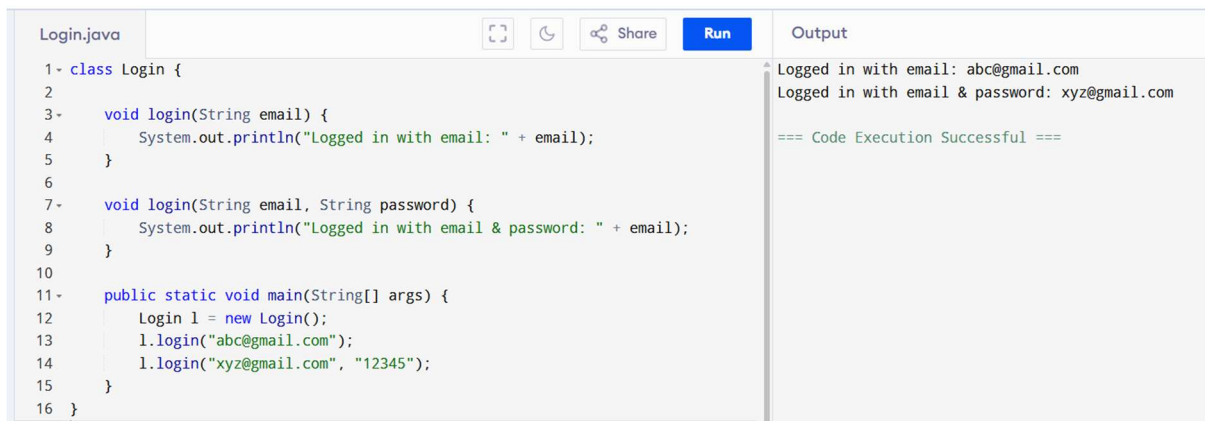
=== Code Execution Successful ===
```

## Assignment 4: Login System

Create Login class:

- login(String email)
- login(String email, String password)

## Concepts: Overloading for flexibility



```
1- class Login {
2
3-   void login(String email) {
4       System.out.println("Logged in with email: " + email);
5   }
6
7-   void login(String email, String password) {
8       System.out.println("Logged in with email & password: " + email);
9   }
10
11-   public static void main(String[] args) {
12       Login l = new Login();
13       l.login("abc@gmail.com");
14       l.login("xyz@gmail.com", "12345");
15   }
16 }
```

Output

```
Logged in with email: abc@gmail.com
Logged in with email & password: xyz@gmail.com

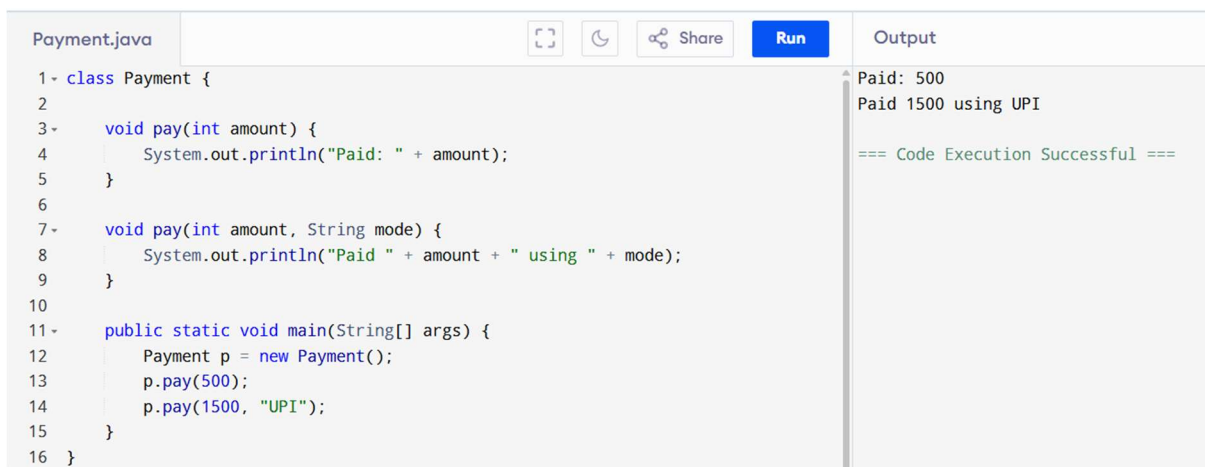
=== Code Execution Successful ===
```

## Assignment 5: Payment Calculation

Create Payment class:

- pay(int amount)
- pay(int amount, String mode)

Concepts: Same method, different params



```
1- class Payment {
2
3-   void pay(int amount) {
4       System.out.println("Paid: " + amount);
5   }
6
7-   void pay(int amount, String mode) {
8       System.out.println("Paid " + amount + " using " + mode);
9   }
10
11-   public static void main(String[] args) {
12       Payment p = new Payment();
13       p.pay(500);
14       p.pay(1500, "UPI");
15   }
16 }
```

Output

```
Paid: 500
Paid 1500 using UPI

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

## Assignment 6: Shape Drawing

Create:

- Shape → draw()
- Circle, Rectangle override draw()
- Use Shape reference

## Concepts: Runtime polymorphism

The screenshot shows a Java IDE with two files: Shape.java and a second file (likely Circle.java or Rectangle.java). The Shape.java file contains a base class Shape with a draw() method and a main() method. The second file contains two subclasses, Circle and Rectangle, both extending Shape and overriding the draw() method. The output window shows the results of running the code: "Drawing a Circle..." and "Drawing a Rectangle...", followed by "=== Code Execution Successful ===".

```
Shape.java
1- class Shape {
2-     void draw() {
3-         System.out.println("Drawing a shape...");
4-     }
5-
6-     public static void main(String[] args) {
7-         Shape s;
8-
9-         s = new Circle();
10        s.draw();
11
12        s = new Rectangle();
13        s.draw();
14    }
15 }
16

Shape.java
17- class Circle extends Shape {
18     @Override
19     void draw() {
20         System.out.println("Drawing a Circle...");
21     }
22 }
23
24- class Rectangle extends Shape {
25     @Override
26     void draw() {
27         System.out.println("Drawing a Rectangle...");
28     }
29 }
30
```

Output

```
^
Drawing a Circle...
Drawing a Rectangle...

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

## Assignment 7: Bank Interest

Create:

- Bank → getInterestRate()
- SBI, HDFC override method

Concepts: Dynamic method dispatch

```
Bank.java
1 class Bank {
2     double getInterestRate() {
3         return 0;
4     }
5
6     public static void main(String[] args) {
7         Bank b;
8
9         b = new SBI();
10        System.out.println("SBI Interest: " + b.getInterestRate());
11
12        b = new HDFC();
13        System.out.println("HDFC Interest: " + b.getInterestRate());
14    }
15 }
```

```
Bank.java
15 }
16
17 class SBI extends Bank {
18     double getInterestRate() {
19         return 5.5;
20     }
21 }
22
23 class HDFC extends Bank {
24     double getInterestRate() {
25         return 6.2;
26     }
27 }
```

Output

```
SBI Interest: 5.5
HDFC Interest: 6.2

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

## Assignment 8: Notification System

Create:

- Notification → send()
- EmailNotification, SMSNotification override send()

Concepts: Real-time example

Notification.java



Share

Run

```
1- class Notification {
2-     void send() {
3-         System.out.println("Sending notification...");
4-     }
5-
6-     public static void main(String[] args) {
7-         Notification n;
8-
9-         n = new EmailNotification();
10-        n.send();
11-
12-        n = new SMSNotification();
13-        n.send();
14-    }
15- }
```

Notification.java



Share

Run

Output

```
16
17- class EmailNotification extends Notification {
18-     @Override
19-     void send() {
20-         System.out.println("Sending Email Notification...");
21-     }
22- }
23
24- class SMSNotification extends Notification {
25-     @Override
26-     void send() {
27-         System.out.println("Sending SMS Notification...");
28-     }
29- }
```

^ Sending Email Notification...  
Sending SMS Notification...  
  
=== Code Execution Successful ===