ASSIGNMENT

B. LIKHITHA

Assignment 1: Calculator Overloading

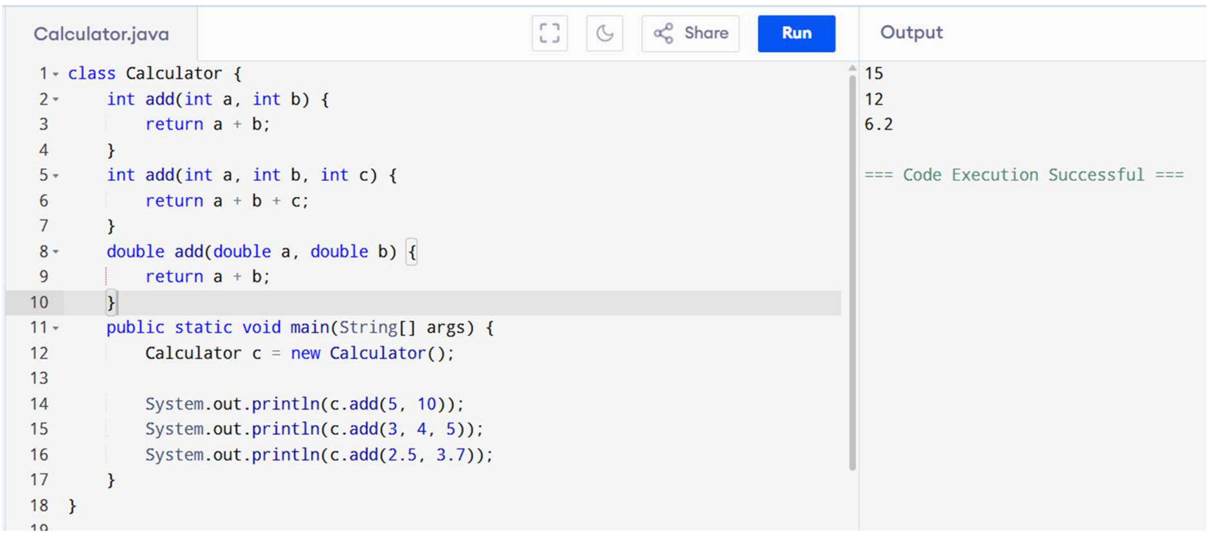
Create Calculator class:

• add(int a, int b)

• add(int a, int b, int c)

• add(double a, double b)

Concepts: Compile-time polymorphism



```java
class Calculator {
    int add(int a, int b) {
        return a + b;
    }
    int add(int a, int b, int c) {
        return a + b + c;
    }
    double add(double a, double b) {
        return a + b;
    }
    public static void main(String[] args) {
        Calculator c = new Calculator();

        System.out.println(c.add(5, 10));
        System.out.println(c.add(3, 4, 5));
        System.out.println(c.add(2.5, 3.7));
    }
}
```

```
15
12
6.2

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

Assignment 2: Area Calculator

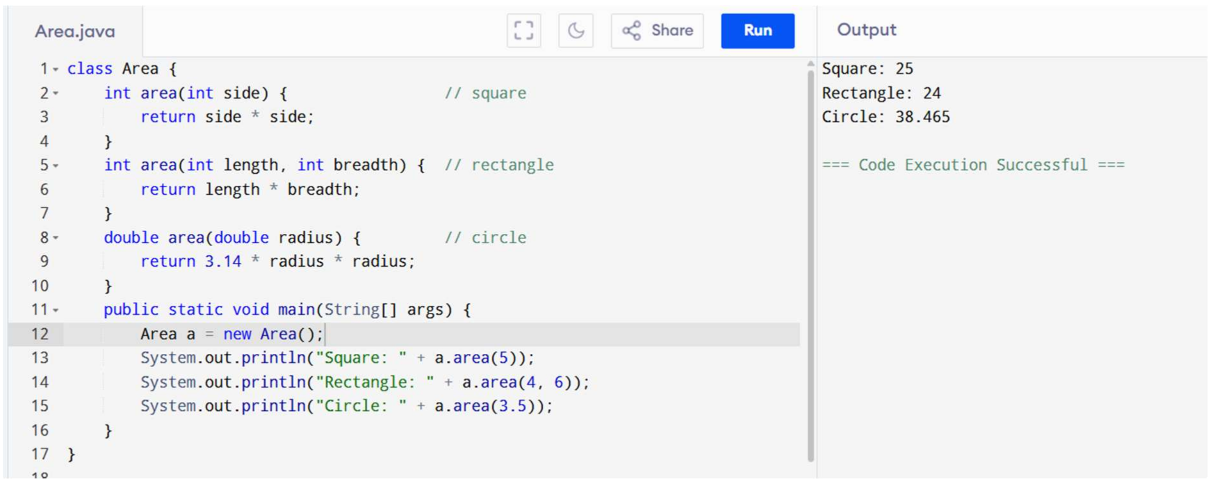
Create Area class:

• area(int side) → square

• area(int length, int breadth) → rectangle

• area(double radius) → circle

Concepts: Method signature change



```java
class Area {
    int area(int side) {             // square
        return side * side;
    }
    int area(int length, int breadth) {  // rectangle
        return length * breadth;
    }
    double area(double radius) {         // circle
        return 3.14 * radius * radius;
    }
    public static void main(String[] args) {
        Area a = new Area();
        System.out.println("Square: " + a.area(5));
        System.out.println("Rectangle: " + a.area(4, 6));
        System.out.println("Circle: " + a.area(3.5));
    }
}
```

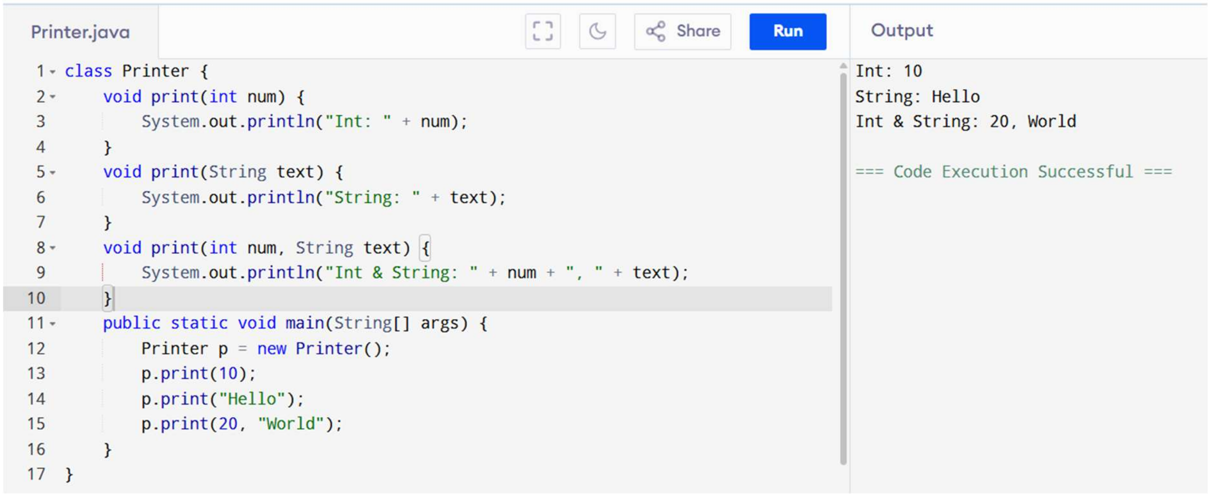
```
Square: 25
Rectangle: 24
Circle: 38.465

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

Assignment 3: Print Data

Create Printer class:

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

Concepts: Overloaded methods



```java
class Printer {
    void print(int num) {
        System.out.println("Int: " + num);
    }
    void print(String text) {
        System.out.println("String: " + text);
    }
    void print(int num, String text) {
        System.out.println("Int & String: " + num + ", " + text);
    }
    public static void main(String[] args) {
        Printer p = new Printer();
        p.print(10);
        p.print("Hello");
        p.print(20, "World");
    }
}
```

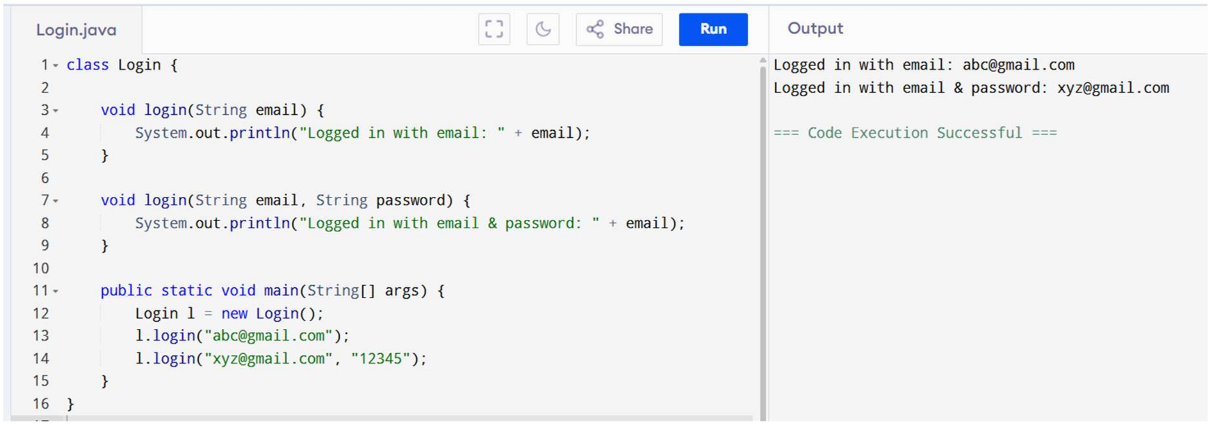
```
Int: 10
String: Hello
Int & String: 20, World

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

Assignment 4: Login System

Create Login class:

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

Concepts: Overloading for flexibility



```java
class Login {

    void login(String email) {
        System.out.println("Logged in with email: " + email);
    }

    void login(String email, String password) {
        System.out.println("Logged in with email & password: " + email);
    }

    public static void main(String[] args) {
        Login l = new Login();
        l.login("abc@gmail.com");
        l.login("xyz@gmail.com", "12345");
    }
}
```

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

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

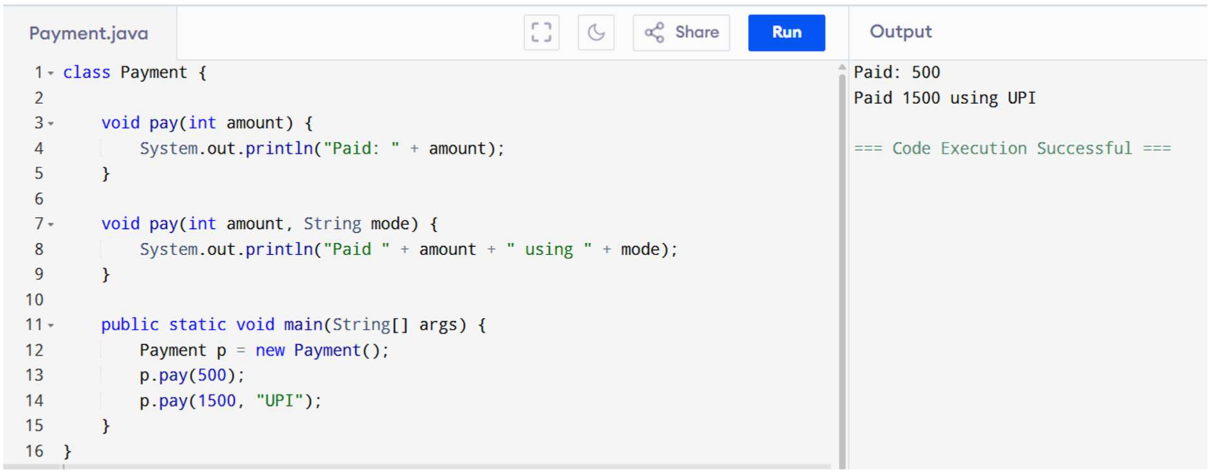
Assignment 5: Payment Calculation

Create Payment class:

• pay(int amount)

• pay(int amount, String mode)

Concepts: Same method, different params



```java
class Payment {

    void pay(int amount) {
        System.out.println("Paid: " + amount);
    }

    void pay(int amount, String mode) {
        System.out.println("Paid " + amount + " using " + mode);
    }

    public static void main(String[] args) {
        Payment p = new Payment();
        p.pay(500);
        p.pay(1500, "UPI");
    }
}
```

```
Paid: 500
Paid 1500 using UPI

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

Assignment 6: Shape Drawing

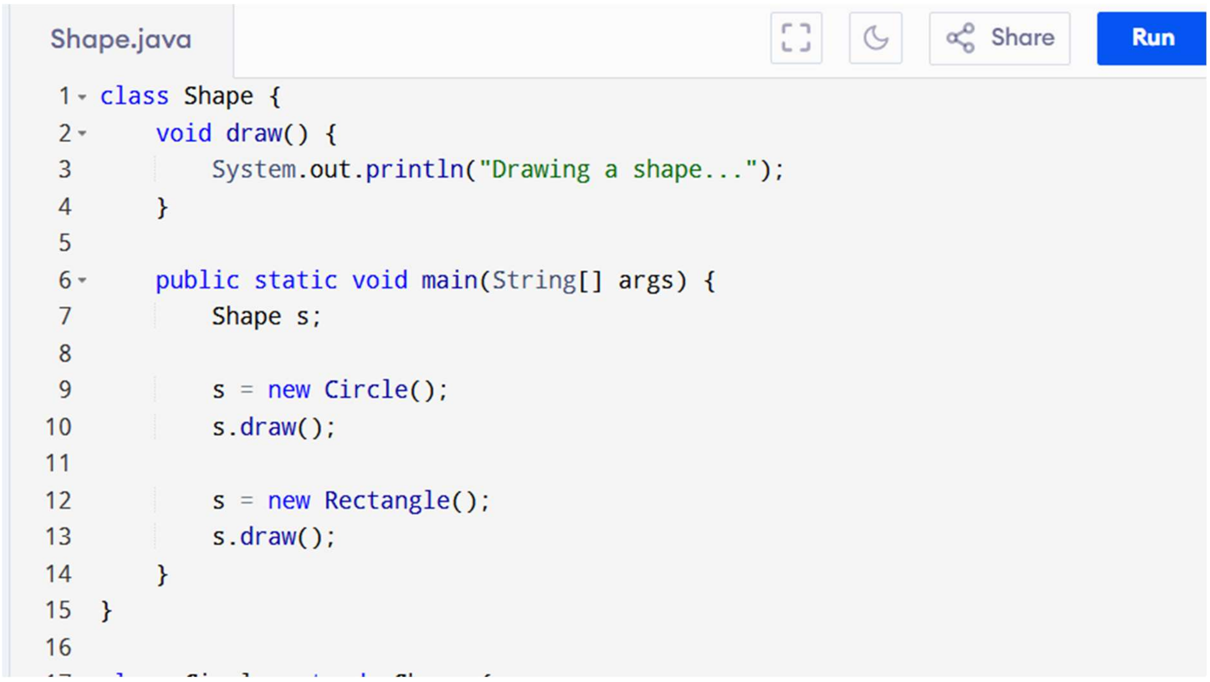
Create:

• Shape → draw()

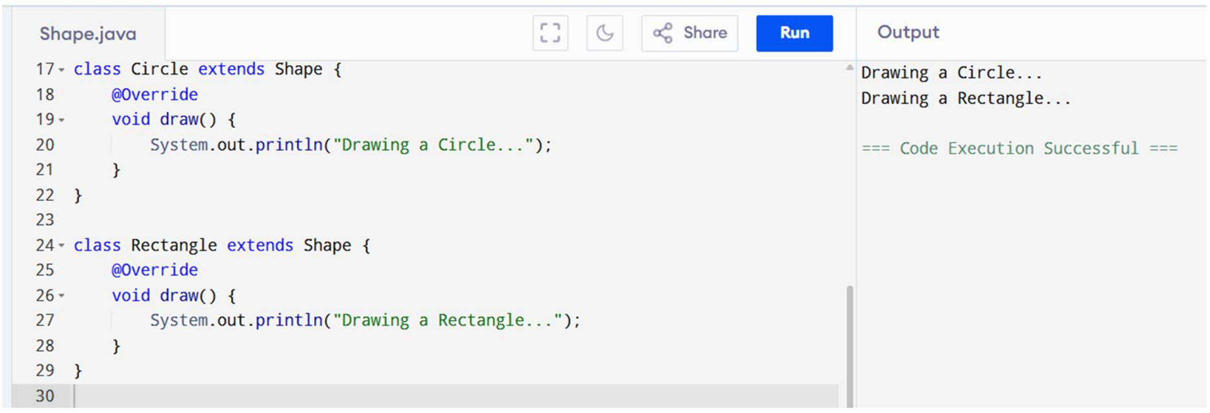
• Circle, Rectangle override draw()

• Use Shape reference

Concepts: Runtime polymorphism



```java
class Shape {
    void draw() {
        System.out.println("Drawing a shape...");
    }

    public static void main(String[] args) {
        Shape s;

        s = new Circle();
        s.draw();

        s = new Rectangle();
        s.draw();
    }
}

```



```java
class Circle extends Shape {
    @Override
    void draw() {
        System.out.println("Drawing a Circle...");
    }
}

class Rectangle extends Shape {
    @Override
    void draw() {
        System.out.println("Drawing a Rectangle...");
    }
}

```

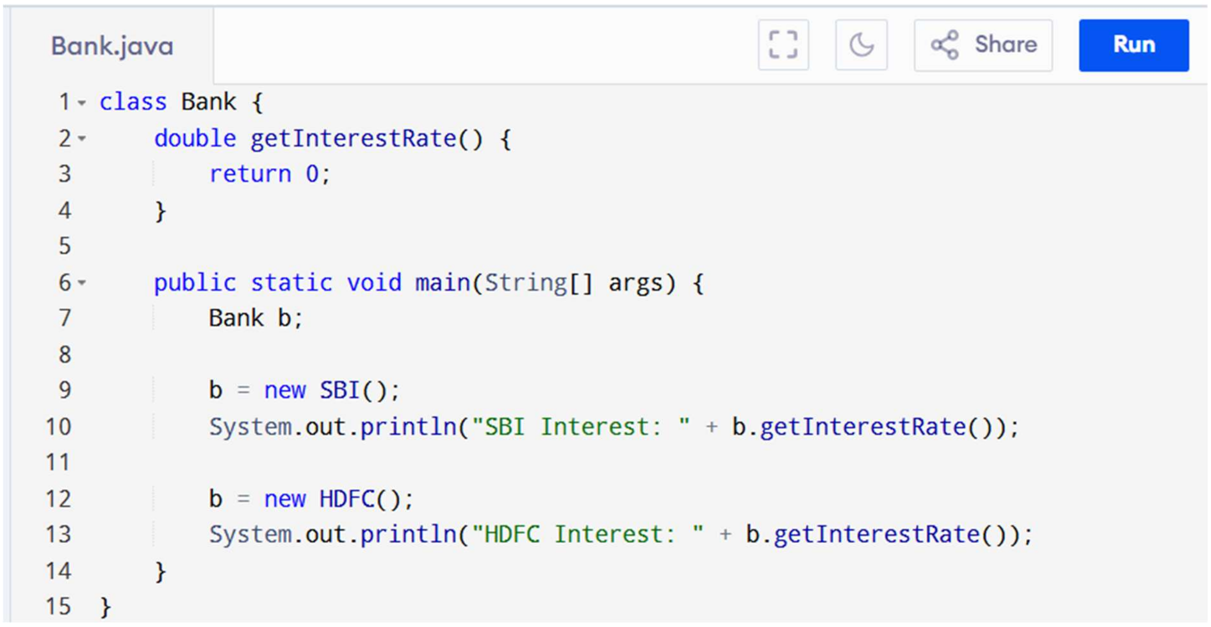
```
Drawing a Circle...
Drawing a Rectangle...

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

Assignment 7: Bank Interest

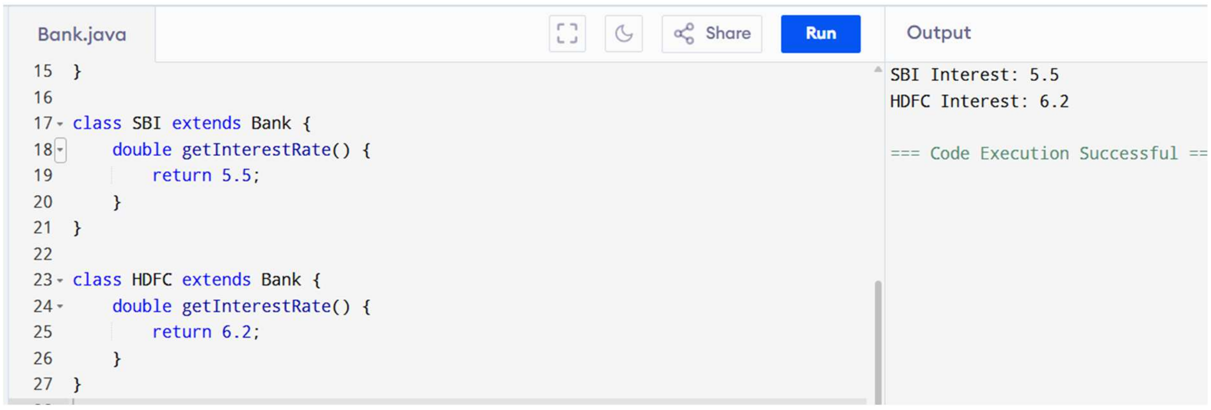
Create:

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

Concepts: Dynamic method dispatch



```java
class Bank {
    double getInterestRate() {
        return 0;
    }

    public static void main(String[] args) {
        Bank b;

        b = new SBI();
        System.out.println("SBI Interest: " + b.getInterestRate());

        b = new HDFC();
        System.out.println("HDFC Interest: " + b.getInterestRate());
    }
}
```



```java
}

class SBI extends Bank {
    double getInterestRate() {
        return 5.5;
    }
}

class HDFC extends Bank {
    double getInterestRate() {
        return 6.2;
    }
}
```

```
SBI Interest: 5.5
HDFC Interest: 6.2

=== Code Execution Successful ==
```

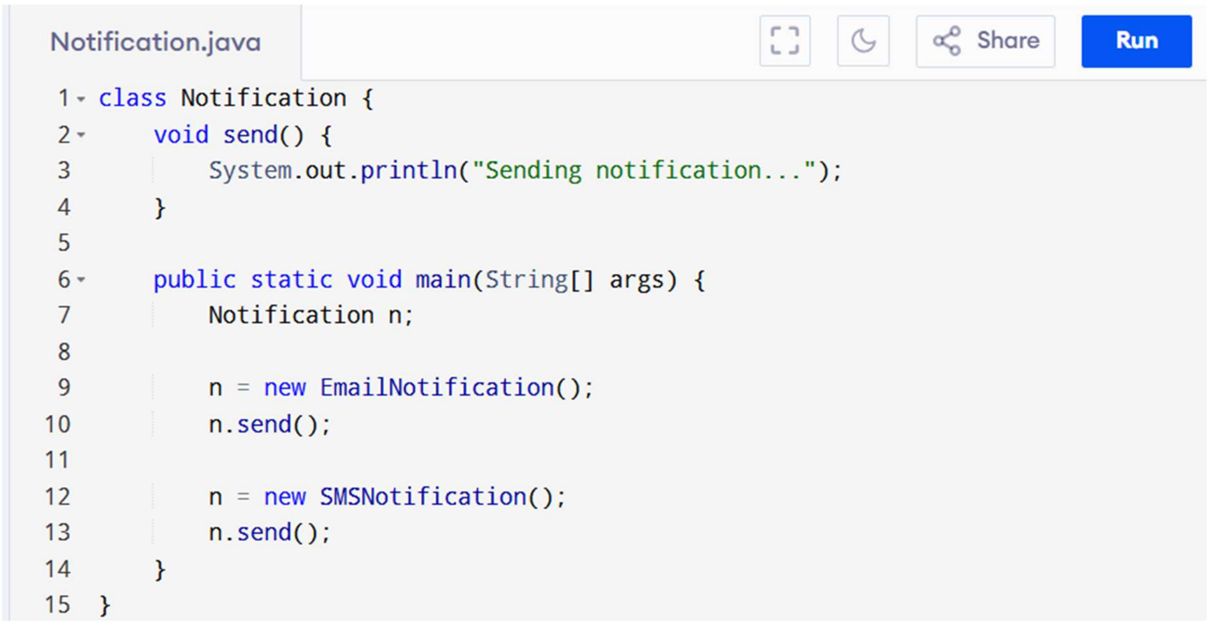
Assignment 8: Notification System

Create:

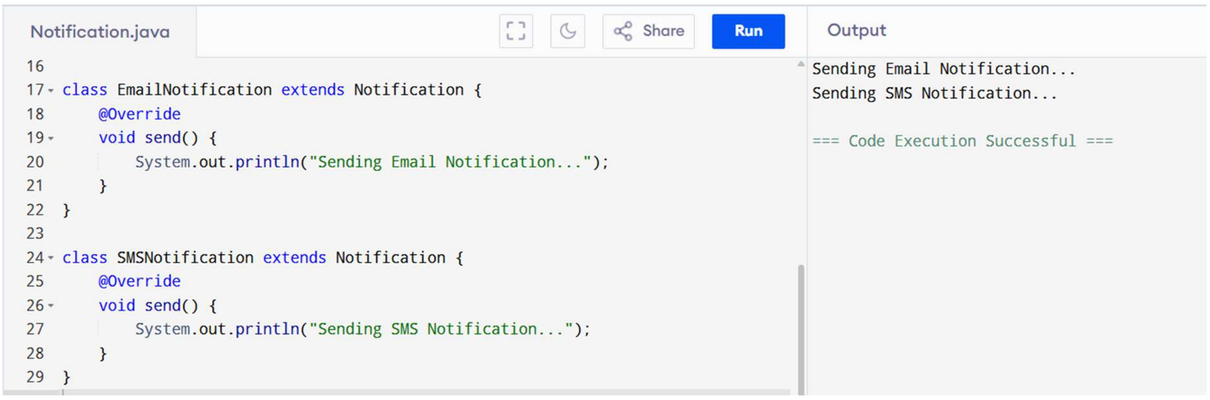
• Notification → send()

• EmailNotification, SMSNotification override send()

Concepts: Real-time example



```java
class Notification {
    void send() {
        System.out.println("Sending notification...");
    }

    public static void main(String[] args) {
        Notification n;

        n = new EmailNotification();
        n.send();

        n = new SMSNotification();
        n.send();
    }
}
```



```java

class EmailNotification extends Notification {
    @Override
    void send() {
        System.out.println("Sending Email Notification...");
    }
}

class SMSNotification extends Notification {
    @Override
    void send() {
        System.out.println("Sending SMS Notification...");
    }
}
```

```
Sending Email Notification...
Sending SMS Notification...

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