Encapsulation :

1. Student with Grade Validation & Configuration ?

Program :

```
package Day5_Encapsulation;
public class Student {
   private String name;
   private int rollNumber;
   private int marks;
   public Student(String name, int rollNumber, int marks) {
      this.name = name;
      this.rollNumber = rollNumber;
      this.marks = (marks >= 0 && marks <= 100) ? marks : 0;
   }
   public String getName() {
      return name;
   }
   public int getRollNumber() {
      return rollNumber;
   }
   public int getMarks() {
      return marks;
   }
   public void displayDetails() {
      System.out.println("Name: " + name);
      System.out.println("Roll Number: " + rollNumber);
      System.out.println("Marks: " + marks);
   }
   public void inputMarks(int newMarks) {
      if (newMarks >= marks && newMarks <= 100) {
         marks = newMarks;
      }
```

```
    }
    public static void main(String[] args) {
        Student s1 = new Student("Dev", 101, 95);
        s1.displayDetails();

        Student s2 = new Student("Yoga", 102, 150);
        s2.displayDetails();

        s1.inputMarks(98);
        s1.displayDetails();

        s2.inputMarks(80);
        s2.displayDetails();
    }
}
```

Output :
```
Name: Dev
Roll Number: 101
Marks: 95
Name: Yoga
Roll Number: 102
Marks: 0
Name: Dev
Roll Number: 101
Marks: 98
Name: Yoga
Roll Number: 102
Marks: 80
```

2. Rectangle Enforced Positive Dimensions ?

Program :

```
package Day5_Encapsulation;
```

```
public class Rectangle {
 private double width;
 private double height;
 public Rectangle(double width, double height) {
    this.width = (width > 0) ? width : 1;
    this.height = (height > 0) ? height : 1;
 }
 public void setWidth(double width) {
    if (width > 0) this.width = width;
 }
 public void setHeight(double height) {
    if (height > 0) this.height = height;
 }
 public double getArea() {
    return width * height;
 }
 public double getPerimeter() {
    return 2 * (width + height);
 }
 public void displayDetails() {
    System.out.println("Width: " + width);
    System.out.println("Height: " + height);
    System.out.println("Area: " + getArea());
    System.out.println("Perimeter: " + getPerimeter());
 }
 public static void main(String[] args) {
    Rectangle r1 = new Rectangle(5, 10);
    r1.displayDetails();

    Rectangle r2 = new Rectangle(-4, 0);
    r2.displayDetails();
    r2.setWidth(7);
    r2.setHeight(3);
    r2.displayDetails();
 }
}
```

Output :
```
Width: 5.0
Height: 10.0
Area: 50.0
Perimeter: 30.0
Width: 1.0
Height: 1.0
Area: 1.0
Perimeter: 4.0
Width: 7.0
Height: 3.0
Area: 21.0
Perimeter: 20.0
```

3. Advanced: Bank Account with Deposit/Withdraw Logic ?

Program :

```
package Day5_Encapsulation;
import java.util.ArrayList;
import java.util.List;
class BankAccount {
 private String accountNumber;
 private String accountHolder;
 private double balance;
 private List<String> transactions = new ArrayList<>();
 public BankAccount(String accountNumber, String accountHolder, double initialBalance) {
    this.accountNumber = accountNumber;
    this.accountHolder = accountHolder;
    this.balance = initialBalance;
    transactions.add("Account created with balance: " + initialBalance);
 }
 public void deposit(double amount) {
    if (amount > 0) {
        balance += amount;
        transactions.add("Deposited: " + amount);
    }
 }
 public boolean withdraw(double amount) {
    if (amount > 0 && amount <= balance) {
        balance -= amount;
        transactions.add("Withdraw: " + amount);
        return true;
    }
    return false;
 }
 public double getBalance() {
    return balance;
 }
 public String getLastTransaction() {
    return transactions.get(transactions.size() - 1);
 }
 public String toString() {
    String maskedAccount = "****" + accountNumber.substring(accountNumber.length() - 4);
    return "Account: " + maskedAccount + ", Holder: " + accountHolder + ", Balance: " + balance;
 }
}
public class BankAccountTest {
 public static void main(String[] args) {
    BankAccount acc = new BankAccount("1234567890", "Dev", 500);
    acc.deposit(200);
    System.out.println(acc);
    acc.withdraw(100);
    System.out.println("Last Transaction: " + acc.getLastTransaction());
    System.out.println(acc);
 }
}
```

Output : Account: ****7890, Holder: Dev, Balance: 700.0

Last Transaction: Withdraw: 100.0

Account: ****7890, Holder: Dev, Balance: 600.0

4. Inner Class Encapsulation: Secure Locker ?

Program :

```
package Day5_Encapsulation;
import java.util.Scanner;
class Locker {
    private String password;
    private boolean isLocked;
    public Locker(String password) {
        this.password = password;
        this.isLocked = true;
    }
    public void unlock(String inputPassword) {
        if (isLocked) {
            if (password.equals(inputPassword)) {
                isLocked = false;
                System.out.println("Locker unlocked successfully!");
            } else {
                System.out.println("Incorrect password. Locker remains locked.");
            }
        } else {
            System.out.println("Locker is already unlocked.");
        }
    }
    public void lock() {
        if (!isLocked) {
            isLocked = true;
            System.out.println("Locker locked successfully!");
        } else {
            System.out.println("Locker is already locked.");
        }
    }
    public void status() {
        System.out.println("Locker is currently " + (isLocked ? "LOCKED" : "UNLOCKED"));
    }
}
public class LockerTest {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Locker myLocker = new Locker("1234");
        myLocker.status();
        System.out.print("Enter password to unlock: ");
        String input = sc.nextLine();
        myLocker.unlock(input);
        myLocker.status();
        myLocker.lock();
        myLocker.status();
    }
}
```

```
Output : Locker is currently LOCKED
Enter password to unlock: 1234
```

```
Locker unlocked successfully!
Locker is currently UNLOCKED
Locker locked successfully!
Locker is currently LOCKED
```

5. Builder Pattern & Encapsulation: Immutable Product ?

Program :

```java
package Day5_Encapsulation;
class Product {
    private String name;
    private double price;
    private String category;
    private String description;
    private Product(Builder builder) {
        this.name = builder.name;
        this.price = builder.price;
        this.category = builder.category;
        this.description = builder.description;
    }
    public void display() {
        System.out.println("Product: " + name);
        System.out.println("Price: " + price);
        System.out.println("Category: " + category);
        System.out.println("Description: " + description);
    }
    public static class Builder {
        private String name;
        private double price;
        private String category;
        private String description;
        public Builder setName(String name) {
            this.name = name;
            return this;
        }
        public Builder setPrice(double price) {
            this.price = price;
            return this;
        }
        public Builder setCategory(String category) {
            this.category = category;
            return this;
        }
        public Builder setDescription(String description) {
            this.description = description;
            return this;
        }
        public Product build() {
            return new Product(this);
        }
    }
}
public class ProductBuilder {
    public static void main(String[] args) {
        Product laptop = new Product.Builder()
                .setName("Dell Inspiron")
                .setPrice(55000.00)
                .setCategory("Electronics")
                .setDescription("Powerful laptop with 16GB RAM and 512GB SSD")
                .build();

        laptop.display();
```

```
   }
}
```

Output : Product: Dell Inspiron

Price: 55000.0

Category: Electronics

Description: Powerful laptop with 16GB RAM and 512GB SSD

Interface :

1. Reverse CharSequence: Custom BackwardSequence ?

Program :

```
package Day5_Interface;
public class ReverseChar implements CharSequence {
   private String reversed;
   public ReverseChar(String original) {
      this.reversed = new StringBuilder(original).reverse().toString();
   }
   public int length() {
      return reversed.length();
   }
   public char charAt(int index) {
      return reversed.charAt(index);
   }
   public CharSequence subSequence(int start, int end) {
      return reversed.substring(start, end);
   }
   public String toString() {
      return reversed;
   }
   public static void main(String[] args) {
      ReverseChar bs = new ReverseChar("hello");
      System.out.println("Reversed: " + bs);
```

```
        System.out.println("Length: " + bs.length());
        System.out.println("Char at 1: " + bs.charAt(1));
        System.out.println("SubSequence(1,4): " + bs.subSequence(1, 4));
    }
}
```

```
Output : Reversed: olleh
Length: 5
Char at 1: l
SubSequence(1,4): lle
```

2. Moveable Shapes Simulation ?

Program :

```
package Day5_Interface;
interface Movable {
    void moveUp();
    void moveDown();
    void moveLeft();
    void moveRight();
}
class MovablePoint implements Movable {
    int x, y, xSpeed, ySpeed;
    public MovablePoint(int x, int y, int xSpeed, int ySpeed) {
        this.x = x;
        this.y = y;
        this.xSpeed = xSpeed;
        this.ySpeed = ySpeed;
    }
    public void moveUp() {
        y -= ySpeed;
    }
```

```
  public void moveDown() {
    y += ySpeed;
  }
  public void moveLeft() {
    x -= xSpeed;
  }
  public void moveRight() {
    x += xSpeed;
  }
  public String toString() {
    return "(" + x + ", " + y + ")";
  }
}
class MovableCircle implements Movable {
  int radius;
  MovablePoint center;
  public MovableCircle(int radius, MovablePoint center) {
    this.radius = radius;
    this.center = center;
  }
  public void moveUp() {
    center.moveUp();
  }
  public void moveDown() {
    center.moveDown();
  }
  public void moveLeft() {
    center.moveLeft();
  }
  public void moveRight() {
    center.moveRight();
    }
    public String toString() {
        return "Circle center: " + center + " radius: " + radius;
    }
}
class MovableRectangle implements Movable {
    MovablePoint topLeft, bottomRight;
    public MovableRectangle(MovablePoint topLeft, MovablePoint bottomRight) {
        if (topLeft.xSpeed != bottomRight.xSpeed || topLeft.ySpeed != bottomRight.ySpeed) {
            throw new IllegalArgumentException("Points must have the same speed");
        }
        this.topLeft = topLeft;
        this.bottomRight = bottomRight;
    }
    public void moveUp() {
        topLeft.moveUp();
        bottomRight.moveUp();
    }
    public void moveDown() {
        topLeft.moveDown();
        bottomRight.moveDown();
    }
    public void moveLeft() {
        topLeft.moveLeft();
        bottomRight.moveLeft();
    }
    public void moveRight() {
        topLeft.moveRight();
        bottomRight.moveRight();
    }
    public String toString() {
        return "Rectangle [TopLeft: " + topLeft + ", BottomRight: " + bottomRight + "]";
    }
}
public class ShapeTest {
    public static void main(String[] args) {
        MovablePoint p1 = new MovablePoint(0, 0, 2, 2);
        MovableCircle c1 = new MovableCircle(5, p1);
        System.out.println(c1);
        c1.moveUp();
        c1.moveRight();
        System.out.println("After move: " + c1);

        MovableRectangle r1 = new MovableRectangle(
            new MovablePoint(0, 0, 1, 1),
            new MovablePoint(2, 2, 1, 1)
        );
        System.out.println(r1);
        r1.moveDown();
        r1.moveRight();
        System.out.println("After move: " + r1);
    }
}
```

Output :
```
Circle center: (0, 0) radius: 5
After move: Circle center: (2, -2) radius: 5
Rectangle [TopLeft: (0, 0), BottomRight: (2, 2)]
After move: Rectangle [TopLeft: (1, 1), BottomRight: (3, 3)]
```

3. Contract Programming: Printer Switch ?

Program :

```
package Day5_Interface;
```

```
interface Printer {
   void print(String document);
}
class LaserPrinter implements Printer {
   public void print(String document) {
      System.out.println("LaserPrinter printing: " + document);
   }
}
class InkjetPrinter implements Printer {
   public void print(String document) {
      System.out.println("InkjetPrinter printing: " + document);
   }
}
public class PrinterSwitch {
   public static void main(String[] args) {
      Printer p;

      p = new LaserPrinter();
      p.print("Report.pdf");

      p = new InkjetPrinter();
      p.print("Photo.jpg");
   }
}
```

Output : LaserPrinter printing: Report.pdf

InkjetPrinter printing: Photo.jpg

4. Extended Interface Hierarchy ?

Program :

```
package Day5_Interface;
```

```
interface BaseVehicle {
   void start();
}
interface AdvancedVehicle extends BaseVehicle {
   void stop();
   boolean refuel(int amount);
}
class Car implements AdvancedVehicle {
   private int fuel;
   public Car(int fuel) {
      this.fuel = fuel;
   }
   public void start() {
      if (fuel > 0) {
         System.out.println("Car started.");
      } else {
         System.out.println("No fuel to start.");
      }
   }
   public void stop() {
      System.out.println("Car stopped.");
   }
   public boolean refuel(int amount) {
      fuel += amount;
      System.out.println("Refueled " + amount + " units. Current fuel: " + fuel);
      return true;
   }
}
public class InterfaceHeirarchy {
   public static void main(String[] args) {
      BaseVehicle base = new Car(5);
```

```
        base.start();
        AdvancedVehicle adv = (AdvancedVehicle) base;
        adv.refuel(10);
        adv.stop();
    }
}
```

```
Output : Car started.
Refueled 10 units. Current fuel: 15
Car stopped.
```

5. Nested Interface for Callback Handling ?

Program :

```
package Day5_Interface;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
class TimeServer {
    public static interface Client {
        void updateTime(LocalDateTime now);
    }
    private List<Client> clients = new ArrayList<>();
    public void registerClient(Client client) {
        clients.add(client);
    }
    public void notifyClients() {
        LocalDateTime now = LocalDateTime.now();
        for (Client c : clients) {
            c.updateTime(now);
        }
    }
}
```

```
class MobileClient implements TimeServer.Client {
    public void updateTime(LocalDateTime now) {
        System.out.println("MobileClient: Time updated to " + now);
    }
}
class DesktopClient implements TimeServer.Client {
    public void updateTime(LocalDateTime now) {
        System.out.println("DesktopClient: Time updated to " + now);
    }
}
public class NestedInterface {
    public static void main(String[] args) {
        TimeServer server = new TimeServer();
        server.registerClient(new MobileClient());
        server.registerClient(new DesktopClient());

        server.notifyClients();
    }
}
```

Output :
```
MobileClient: Time updated to 2025-08-10T14:31:28.102141100
DesktopClient: Time updated to 2025-08-10T14:31:28.102141100
```

6. Default and Static Methods in Interfaces ?

Program :

```
package Day5_Interface;
interface Polygon {
    double getArea();
    default double getPerimeter(int... sides) {
        int sum = 0;
        for (int side : sides) {
            sum += side;
        }
        return sum;
    }
    static String shapeInfo() {
        return "Polygon: A closed shape with straight sides.";
    }
}
class Rectangle implements Polygon {
    int length, width;
    public Rectangle(int length, int width) {
        this.length = length;
        this.width = width;
    }
    public double getArea() {
        return length * width;
    }
}
class Triangle implements Polygon {
    int base, height;
    public Triangle(int base, int height) {
        this.base = base;
        this.height = height;
    }
    public double getArea() {
        return 0.5 * base * height;
    }
}
public class InterfaceMethods {
    public static void main(String[] args) {
        Rectangle r = new Rectangle(10, 5);
        Triangle t = new Triangle(6, 4);
        System.out.println("Rectangle area: " + r.getArea());
        System.out.println("Rectangle perimeter: " + r.getPerimeter(10, 5, 10, 5));
        System.out.println("Triangle area: " + t.getArea());
        System.out.println("Triangle perimeter: " + t.getPerimeter(6, 4, 5));
        System.out.println(Polygon.shapeInfo());
    }
}
```

Output : Rectangle area: 50.0

```
Rectangle perimeter: 30.0
Triangle area: 12.0
Triangle perimeter: 15.0
Polygon: A closed shape with straight sides.
```

Lambda Expressions :

1. Sum of Two Integers ?

Program :

```
package Day5_LambdaExpressions;
interface Sum {
    int add(int a, int b);
}
public class Lambda {
    public static void main(String[] args) {
        Sum s = (x, y) -> x + y;
        int a = 15;
        int b = 25;
        System.out.println("Sum of " + a + " and " + b + " = " + s.add(a, b));
    }
}
```

Output : Sum of 15 and 25 = 40

2. Define a functional interface SumCalculator { int sum(int a, int b); } and a lambda expression to sum two integers.

Program :

```
package Day5_LambdaExpressions;

interface SumCalculator {

   int sum(int a, int b);

}

public class LambdaSum {

   public static void main(String[] args) {

      SumCalculator adder = (a, b) -> a + b;

      int result = adder.sum(10, 20);

      System.out.println("Sum: " + result);

   }

}
```

Outpt : Sum: 30

3. Check If a String Is Empty. Create a lambda (via a functional interface like Predicate<String>) that returns true if a given string is empty.

Program :

```
package Day5_LambdaExpressions;

import java.util.function.Predicate;

public class String_Empty {

        public static void main(String[] args) {

                Predicate<String> isEmpty = s -> s.isEmpty();

                String s1 = "";

                String s2 = "Hello";

                System.out.println(isEmpty.test(s1));

                System.out.println(isEmpty.test(s2));

        }

}
```

Output : true

false

4. Filter Even or Odd Numbers ?

Program :

```
package Day5_LambdaExpressions;

import java.util.function.Predicate;

public class LambdaEvenOdd {

	public static void main(String[] args) {

		int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

		Predicate<Integer> isEven = n -> n % 2 == 0;

		Predicate<Integer> isOdd = n -> n % 2 != 0;

		System.out.println("Even Numbers:");

		for (int n : numbers) {

			if (isEven.test(n)) {

				System.out.println(n);

			}

		}

		System.out.println("\nOdd Numbers:");

		for (int n : numbers) {

			if (isOdd.test(n)) {

				System.out.println(n);

			}

		}

	}

}
```

Output :
```
Even Numbers:

2

4

6

8

10


Odd Numbers:
```

```
1
3
5
7
9
```

5. Convert Strings to Uppercase/Lowercase ?

Program :

```
package Day5_LambdaExpressions;
import java.util.Arrays;
import java.util.List;
public class LambdaConvert {
   public static void main(String[] args) {
      List<String> words = Arrays.asList("Apple", "Banana", "Cherry");
      System.out.println("Uppercase:");
      words.stream()
          .map(word -> word.toUpperCase())
          .forEach(word -> System.out.println(word));
      System.out.println("Lowercase:");
      words.stream()
          .map(word -> word.toLowerCase())
          .forEach(word -> System.out.println(word));
   }
}
```

Output :

```
Uppercase:
APPLE
BANANA
CHERRY
Lowercase:
apple
banana
```

```
cherry
```

6. Sort Strings by Length or Alphabetically ?

Program :

```
package Day5_LambdaExpressions;
import java.util.Arrays;
import java.util.List;
public class LambdaSort {
  public static void main(String[] args) {
    List<String> names = Arrays.asList("Dev", "Yoga", "Muktha", "Sai");
    System.out.println("Sort by Length:");
    names.stream()
        .sorted((a, b) -> Integer.compare(a.length(), b.length()))
        .forEach(System.out::println);
  }
}
```

Output :

```
Sort by Length:
Dev
Sai
Yoga
Muktha
```

7. Aggregate Operations (Sum, Max, Average) on Double Arrays ?

Program :

```
package Day5_LambdaExpressions;
import java.util.Arrays;
public class LambdaAggregate {
  public static void main(String[] args) {
    double[] numbers = { 10.5, 20.3, 30.7, 40.2 };
```

```
        double sum = Arrays.stream(numbers).sum();
        double max = Arrays.stream(numbers).max().orElse(Double.NaN);
        double avg = Arrays.stream(numbers).average().orElse(Double.NaN);
        System.out.println("Sum: " + sum);
        System.out.println("Max: " + max);
        System.out.println("Average: " + avg);
    }
}
```

Output : Sum: 101.7

Max: 40.2

Average: 25.425

8. Create similar lambdas for max/min.

Program :

```
package Day5_LambdaExpressions;
interface MathOperation {
    int operate(int a, int b);
}
public class LambdaMaxMin {
    public static void main(String[] args) {
        MathOperation max = (a, b) -> a > b ? a : b;
        MathOperation min = (a, b) -> a < b ? a : b;

        System.out.println("Max: " + max.operate(10, 20));
        System.out.println("Min: " + min.operate(10, 20));
    }
}
```

Output : Max: 20

Min: 10

9. Calculate Factorial ?

Program :

```
package Day5_LambdaExpressions;

import java.util.function.IntFunction;

public class LambdaFactorial {

    public static void main(String[] args) {

        IntFunction<Long> factorial = n -> {

            long result = 1;

            for (int i = 2; i <= n; i++) {

                result *= i;

            }

            return result;

        };

        int number = 5;

        System.out.println("Factorial of " + number + " is: " + factorial.apply(number));

    }

}
```

Output : Factorial of 5 is: 120