ENCAPSULATION

1.Student with Grade Validation & Configuration

package day5\_Assign;

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;

if (marks >= 0 && marks <= 100)

{

this.marks = marks;

}

else

{

this.marks = 0;

}

}

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

public int getRollNumber()

{

return rollNumber;

}

public void setRollNumber(int rollNumber)

{

this.rollNumber = 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 class Day5\_assig1 {

public static void main(String[] args)

{

Student s1 = new Student("riya", 1234, 90);

s1.displayDetails();

System.***out***.println();

Student s2 = new Student("varma", 5678, -10);

s2.displayDetails();

System.***out***.println();

Student s3 = new Student("RiyaVarma", 0, 70);

s3.setName("riyavarma");

s3.setRollNumber(9012);

s3.displayDetails();

}

}

2.Rectangle Enforced Positive Dimensions

package day5\_Assign;

public class Rectangle

{

private double width;

private double height;

public Rectangle(double width, double height)

{

if (width > 0)

{

this.width = width;

}

else

{

this.width = 1;

}

if (height > 0) {

this.height = height;

}

else

{

this.height = 1;

}

}

public double getWidth()

{

return width;

}

public void setWidth(double width)

{

if (width > 0)

{

this.width = width;

}

else

{

this.width = 1;

}

}

public double getHeight()

{

return height;

}

public void setHeight(double height)

{

if (height > 0)

{

this.height = height;

}

else

{

this.height = 1;

}

}

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 class Day5\_2Q {

public static void main(String[] args) {

Rectangle r1 = new Rectangle(5, 10);

r1.displayDetails();

System.***out***.println();

Rectangle r2 = new Rectangle(-3, 8);

r2.displayDetails();

System.***out***.println();

Rectangle r3 = new Rectangle(7, -4);

r3.setWidth(-2);

r3.setHeight(6);

r3.displayDetails();

}

}

3.Advanced: Bank Account with Deposit/Withdraw Logic

package day5\_Assign;

public class Transaction {

private String type;

private double amount;

private Transaction next;

public Transaction(String type, double amount)

{

this.type = type;

this.amount = amount;

this.next = null;

}

public String getType()

{

return type;

}

public double getAmount()

{

return amount;

}

public Transaction getNext()

{

return next;

}

public void setNext(Transaction next)

{

this.next = next;

}

public String toString()

{

return type + ": " + amount;

}

}

package day5\_Assign;

public class BankAccount {

private String accountNumber;

private String accountHolder;

private double balance;

private Transaction lastTransaction;

public BankAccount(String accountNumber,String accountHolder,double initialBalance)

{

this.accountNumber = accountNumber;

this.accountHolder = accountHolder;

this.balance = initialBalance;

if (initialBalance > 0) {

lastTransaction = new Transaction("Initial deposit", initialBalance);

}

}

public void deposit(double amount) {

if (amount > 0) {

balance += amount;

Transaction newTransaction = new Transaction("Deposit", amount);

newTransaction.setNext(lastTransaction);

lastTransaction = newTransaction;

System.***out***.println("Deposited: " + amount);

} else {

System.***out***.println("Invalid deposit amount");

}

}

public boolean withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

Transaction newTransaction = new Transaction("Withdrawal", amount);

newTransaction.setNext(lastTransaction);

lastTransaction = newTransaction;

System.***out***.println("Withdrawn: " + amount);

return true;

}

else {

System.***out***.println("invalid withdrawal amount.");

return false;

}

}

public double getBalance() {

return balance;

}

public Transaction getLastTransaction() {

return lastTransaction;

}

public String toString() {

String maskedAccountNumber = "XXXX-XXXX-" + accountNumber.substring(accountNumber.length() - 4);

return "Account Number: " + maskedAccountNumber + "\nAccount Holder: " + accountHolder + "\nBalance: " + balance;

}

}

package day5\_Assign;

public class Day5\_3Q {

public static void main(String[] args)

{

BankAccount a = new BankAccount("1234567890", "John Doe", 1000);

System.***out***.println(a.toString());

a.deposit(500);

System.***out***.println("Balance: " + a.getBalance());

a.withdraw(200);

System.***out***.println("Balance: " + a.getBalance());

System.***out***.println("Last transaction: " + a.getLastTransaction());

}

}

4.Inner Class Encapsulation: Secure Locker

package day5\_Assign;

public class Locker {

private int id;

private boolean locked;

private String code;

private SecurityMgr sm;

public Locker(int id,String code){

this.id=id;

this.code=code;

this.sm=new SecurityMgr();

this.locked=true;

}

public void lock(){

locked=true;

}

public boolean unlock(String c){

return sm.chk(c);

}

public boolean isLocked(){

return locked;

}

private class SecurityMgr{

public boolean chk(String c){

if(c.equals(code)){

locked=false;

return true;

}

return false;

}

}

public static void main(String[] args){

Locker l=new Locker(101,"1234");

System.***out***.println(l.isLocked());

System.***out***.println(l.unlock("9999"));

System.***out***.println(l.isLocked());

System.***out***.println(l.unlock("1234"));

System.***out***.println(l.isLocked());

}

}

5.Builder Pattern & Encapsulation: Immutable Product

package day5\_Assign;

public class Product {

private final String name;

private final String code;

private final double price;

private final String category;

private Product(Builder b){

this.name=b.name;

this.code=b.code;

this.price=b.price;

this.category=b.category;

}

public String getName(){

return name;

}

public String getCode(){

return code;

}

public double getPrice(){

return price;

}

public String getCategory(){

return category;

}

public static class Builder{

private String name;

private String code;

private double price;

private String category;

public Builder withName(String name){

this.name=name;

return this;

}

public Builder withCode(String code){

this.code=code;

return this;

}

public Builder withPrice(double price){

if(price<0)throw new IllegalArgumentException("neg price");

this.price=price;

return this;

}

public Builder withCategory(String category){

this.category=category;

return this;

}

public Product build(){

return new Product(this);

}

}

public static void main(String[] args){

Product p=new Product.Builder().withName("Pen").withCode("P001").withPrice(10.5).withCategory("Stationery").build();

System.***out***.println(p.getName()+" "+p.getCode()+" "+p.getPrice()+" "+p.getCategory());

}

}

INTERFACE

1. Reverse CharSequence: Custom BackwardSequence

package day5\_Assign;

public class BackwardSequence implements CharSequence{

private String seq;

public BackwardSequence(String seq){

this.seq=new StringBuilder(seq).reverse().toString();

}

public int length(){

return seq.length();

}

public char charAt(int index){

return seq.charAt(index);

}

public CharSequence subSequence(int start,int end){

return seq.subSequence(start,end);

}

public String toString(){

return seq;

}

public static void main(String[] args){

BackwardSequence bs=new BackwardSequence("hello");

System.out.println(bs.toString());

System.out.println(bs.length());

System.out.println(bs.charAt(0));

System.out.println(bs.subSequence(0,3));

}

}

2. Moveable Shapes Simulation

package day5\_Assign;

public interface Movable {

void moveUp();

void moveDown();

void moveLeft();

void moveRight();

}

package day5\_Assign;

public class MovablePoint implements Movable

{

private 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 + ")";

}

public int getXSpeed()

{

return xSpeed;

}

public int getYSpeed()

{

return ySpeed;

}

}

package day5\_Assign;

public class MovableCircle implements Movable

{

private MovablePoint center;

private int radius;

public MovableCircle(MovablePoint center, int radius)

{

this.center = center; this.radius = radius;

}

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;

}

}

package day5\_Assign;

public class MovableRectangle implements Movable

{

private MovablePoint topLeft, bottomRight;

public MovableRectangle(MovablePoint tl, MovablePoint br)

{

if (tl.getXSpeed() != br.getXSpeed() || tl.getYSpeed() != br.getYSpeed()) {

throw new IllegalArgumentException("Points must have same speed");

}

this.topLeft = tl;

this.bottomRight = br;

}

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 top-left " + topLeft + " bottom-right " + bottomRight;

}

}

package day5\_Assign;

public class Q2 {

public static void main(String[] args) {

MovablePoint p = new MovablePoint(0, 0, 1, 1);

MovableCircle c = new MovableCircle(p, 5);

MovableRectangle r = new MovableRectangle(p, new MovablePoint(5, 5, 1, 1));

System.***out***.println(p);

System.***out***.println(c);

System.***out***.println(r);

p.moveRight();

p.moveDown();

System.***out***.println(p);

System.***out***.println(c);

System.***out***.println(r);

}

}

3. Contract Programming: Printer Switch

package day5\_Assign;

public interface Printer {

void print(String doc);

}

package day5\_Assign;

public class LaserPrinter implements Printer {

public void print(String doc) {

System.***out***.println("Printing " + doc + " using laser printer");

}

}

package day5\_Assign;

public class InkjetPrinter implements Printer {

public void print(String doc) {

System.***out***.println("Printing " + doc + " using inkjet printer");

}

}

package day5\_Assign;

public class Q3 {

public static void main(String[] args) {

Printer p = new LaserPrinter();

p.print("Doc1");

Printer p1 = new InkjetPrinter();

p1.print("Doc2");

}

}

4. Extended Interface Hierarchy

package day5\_Assign;

public interface BaseVehicle {

void start();

}

package day5\_Assign;

public interface AdvancedVehicle extends BaseVehicle {

void stop();

boolean refuel(int amount);

}

package day5\_Assign;

public class Car implements AdvancedVehicle {

private int fuelLvl;

public Car(int fuelLvl) {

this.fuelLvl = fuelLvl;

}

public void start() {

System.***out***.println("Car started");

}

public void stop() {

System.***out***.println("Car stopped");

}

public boolean refuel(int amount) {

fuelLvl+= amount;

return true;

}

}

package day5\_Assign;

public class Q4 {

public static void main(String[] args) {

AdvancedVehicle c = new Car(50);

c.start();

c.stop();

c.refuel(20);

BaseVehicle c1 = (BaseVehicle) c;

c1.start(); }}

5. Nested Interface for Callback Handling

package day5\_Assign;

import java.time.LocalDateTime;

import java.util.ArrayList;

import java.util.List;

public class TimeServer{

public interface Client{

void updateTime(LocalDateTime now);

}

private List<Client> clients=new ArrayList<>();

public void registerClient(Client c){

clients.add(c);

}

public void notifyClients(){

LocalDateTime now=LocalDateTime.*now*();

for(Client c:clients){

c.updateTime(now);

}

}

public static void main(String[] args){

TimeServer s=new TimeServer();

s.registerClient(now->System.***out***.println("Client1:"+now));

s.registerClient(now->System.***out***.println("Client2:"+now));

s.notifyClients();

}

}

6. Default and Static Methods in Interfaces

public interface Polygon{

double getArea();

default double getPerimeter(int...sides){

int sum=0;

for(int s:sides){

sum+=s;

}

return sum;

}

static String shapeInfo(){

return "This is a polygon";

}

}

public class Rectangle implements Polygon{

private int len;

private int wid;

public Rectangle(int len,int wid){

this.len=len;

this.wid=wid;

}

public double getArea(){

return len\*wid;

}

}

public class Triangle implements Polygon{

private int b;

private int h;

public Triangle(int b,int h){

this.b=b;

this.h=h;

}

public double getArea(){

return 0.5\*b\*h;

}

}

public class Main{

public static void main(String[] args){

Rectangle r=new Rectangle(4,5);

System.out.println(r.getArea());

System.out.println(r.getPerimeter(4,5,4,5));

Triangle t=new Triangle(3,4);

System.out.println(t.getArea());

System.out.println(t.getPerimeter(3,4,5));

System.out.println(Polygon.shapeInfo());

}

}

**LAMBDA EXPRESSIONS :**

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

@FunctionalInterface

interface SumCalculator{

int sum(int a,int b);

}

public class Q1{

public static void main(String[] args){

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

System.out.println(add.sum(10,20));

}

}

2. Check If a String Is Empty

import java.util.function.Predicate;

public class Q2{

public static void main(String[] args){

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

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

System.out.println(isEmpty.test("hello"));

}

}

3. Filter Even or Odd Numbers

package day5\_Assign;

import java.util.Arrays;

import java.util.List;

import java.util.stream.Collectors;

public class Q3{

public static void main(String[] args){

List<Integer> numbers=Arrays.*asList*(1,2,3,4,5,6,7,8,9,10);

List<Integer> even=numbers.stream()

.filter(n->n%2==0)

.collect(Collectors.*toList*());

List<Integer> odd=numbers.stream()

.filter(n->n%2!=0)

.collect(Collectors.*toList*());

System.***out***.println(even);

System.***out***.println(odd);

}

}

4. Convert Strings to Uppercase/Lowercase

package day5\_Assign;

import java.util.Arrays;

import java.util.List;

import java.util.stream.Collectors;

public class Q4{

public static void main(String[] args){

List<String> strs=Arrays.*asList*("apple","banana","cherry");

List<String> upper=strs.stream()

.map(s->s.toUpperCase())

.collect(Collectors.*toList*());

List<String> lower=strs.stream()

.map(s->s.toLowerCase())

.collect(Collectors.*toList*());

System.***out***.println(upper);

System.***out***.println(lower);

}

}

5. Sort Strings by Length or Alphabetically

import java.util.Arrays;

import java.util.List;

import java.util.stream.Collectors;

public class Q5 {

public static void main(String[] args) {

List<String> strings = Arrays.asList("hello", "abc", "xyz", "longer string");

List<String> sortedByLength = strings.stream()

.sorted((s1, s2) -> Integer.compare(s1.length(), s2.length()))

.collect(Collectors.toList());

List<String> sortedAlphabetically = strings.stream()

.sorted()

.collect(Collectors.toList());

System.out.println("Sorted by length: " + sortedByLength); // [abc, xyz, hello, longer string]

System.out.println("Sorted alphabetically: " + sortedAlphabetically); // [abc, hello, longer string, xyz]

}

}

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

package day5\_Assign;

public class Q6{

public static void main(String[] args){

double[] nums={2.5,3.7,1.8,4.2,5.0};

double sum=0;

double max=Double.NEGATIVE\_INFINITY;

double min=Double.POSITIVE\_INFINITY;

for(double n:nums){

sum+=n;

if(n>max){

max=n;

}

if(n<min){

min=n;

}

}

double avg=sum/nums.length;

System.out.println(sum);

System.out.println(max);

System.out.println(avg);

}

}

7. Create similar lambdas for max/min.

@FunctionalInterface

interface MaxCal{

double max(double[] arr);

}

@FunctionalInterface

interface MinCal{

double min(double[] arr);

}

public class Q7{

public static void main(String[] args){

MaxCal maxCal=arr->{

double max=Double.NEGATIVE\_INFINITY;

for(double d:arr){

if(d>max){

max=d;

}

}

return max;

};

MinCal minCal=arr->{

double min=Double.POSITIVE\_INFINITY;

for(double d:arr){

if(d<min){

min=d;

}

}

return min;

};

double[] numbers={3.5,7.2,1.1,4.8,2.9};

System.out.println(maxCal.max(numbers));

System.out.println(minCal.min(numbers));

}

}

8. Calculate Factorial

@FunctionalInterface

interface FactorialCalculator{

long cal(int n);

}

public class Q8{

public static void main(String[] args){

FactorialCalculator factorial = n -> {

long res = 1;

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

res \*= i;

}

return res;

};

System.out.println(factorial.cal(5));

}

}