**Java code**

class Employee {

private String firstName;

private String lastName;

private double monthlySalary;

public Employee(String firstName, String lastName, double monthlySalary) {

this.firstName = firstName;

this.lastName = lastName;

if (monthlySalary > 0.0) {

this.monthlySalary = monthlySalary;

} else {

this.monthlySalary = 0.0;

}

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getFirstName() {

return firstName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public String getLastName() {

return lastName;

}

public void setMonthlySalary(double monthlySalary) {

if (monthlySalary > 0.0) {

this.monthlySalary = monthlySalary;

} else {

this.monthlySalary = 0.0;

}

}

public double getMonthlySalary() {

return monthlySalary;

}

public double getYearlySalary() {

return monthlySalary \* 12;

}

public void raiseSalary(double percentage) {

double raiseAmount = monthlySalary \* (percentage / 100);

monthlySalary += raiseAmount;

}

}

class EmployeeTest {

public static void main(String[] args) {

Employee emp1 = new Employee("John", "Doe", 5000.0);

Employee emp2 = new Employee("Jane", "Smith", -2000.0);

System.out.println("Employee 1:");

System.out.println("Full Name: " + emp1.getFirstName() + " " + emp1.getLastName());

System.out.println("Yearly Salary: " + emp1.getYearlySalary());

System.out.println("\nEmployee 2:");

System.out.println("Full Name: " + emp2.getFirstName() + " " + emp2.getLastName());

System.out.println("Yearly Salary: " + emp2.getYearlySalary());

emp1.raiseSalary(10);

emp2.raiseSalary(10);

System.out.println("\nAfter 10% raise:");

System.out.println("Employee 1 Yearly Salary: " + emp1.getYearlySalary());

System.out.println("Employee 2 Yearly Salary: " + emp2.getYearlySalary());

}

}

class SavingsAccount {

private static double annualInterestRate;

private double savingsBalance;

public SavingsAccount(double savingsBalance) {

this.savingsBalance = savingsBalance;

}

public void calculateMonthlyInterest() {

double monthlyInterest = (savingsBalance \* annualInterestRate) / 12;

savingsBalance += monthlyInterest;

}

public static void modifyInterestRate(double newInterestRate) {

annualInterestRate = newInterestRate;

}

public double getSavingsBalance() {

return savingsBalance;

}

}

class SavingsAccountTest {

public static void main(String[] args) {

SavingsAccount saver1 = new SavingsAccount(2000.0);

SavingsAccount saver2 = new SavingsAccount(3000.0);

SavingsAccount.modifyInterestRate(0.05);

System.out.println("Saver 1 - Initial Balance: $" + saver1.getSavingsBalance());

System.out.println("Saver 2 - Initial Balance: $" + saver2.getSavingsBalance());

saver1.calculateMonthlyInterest();

saver2.calculateMonthlyInterest();

System.out.println("\nSaver 1 - Balance after one month's interest: $" + saver1.getSavingsBalance());

System.out.println("Saver 2 - Balance after one month's interest: $" + saver2.getSavingsBalance());

SavingsAccount.modifyInterestRate(0.06);

saver1.calculateMonthlyInterest();

saver2.calculateMonthlyInterest();

System.out.println("\nSaver 1 - Balance after two months' interest: $" + saver1.getSavingsBalance());

System.out.println("Saver 2 - Balance after two months' interest: $" + saver2.getSavingsBalance());

}

}

abstract class Shape {

protected double area;

protected double perimeter;

public abstract void calculateArea();

public abstract void calculatePerimeter();

public void display() {

System.out.println("Area: " + area);

System.out.println("Perimeter: " + perimeter);

}

}

class Triangle extends Shape {

private double base;

private double height;

public Triangle(double base, double height) {

this.base = base;

this.height = height;

}

public void calculateArea() {

area = 0.5 \* base \* height;

}

public void calculatePerimeter() {

// Implement perimeter calculation for triangle

}

}

class Rectangle extends Shape {

private double length;

private double width;

public Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

public void calculateArea() {

area = length \* width;

}

public void calculatePerimeter() {

perimeter = 2 \* (length + width);

}

}

class Circle extends Shape {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

public void calculateArea() {

area = Math.PI \* radius \* radius;

}

public void calculatePerimeter() {

perimeter = 2 \* Math.PI \* radius;

}

}

class Cube extends Shape {

private double side;

public Cube(double side) {

this.side = side;

}

public void calculateArea() {

area = 6 \* side \* side;

}

public void calculatePerimeter() {

// Implement perimeter calculation for cube

}

}

class Square extends Shape {

private double side;

public Square(double side) {

this.side = side;

}

public void calculateArea() {

area = side \* side;

}

public void calculatePerimeter() {

perimeter = 4 \* side;

}

}

public class Test {

public static void main(String[] args) {

Shape[] shapes = new Shape[5];

shapes[0] = new Triangle(5, 10);

shapes[1] = new Rectangle(4, 6);

shapes[2] = new Circle(3);

shapes[3] = new Cube(2);

shapes[4] = new Square(5);

for (Shape shape : shapes) {

shape.calculateArea();

shape.calculatePerimeter();

shape.display();

System.out.println();

}

}

}

import java.io.\*;

public class ExceptionHandlingExample {

public static void main(String[] args) {

try {

// Predefined compile time exception - FileNotFoundException

FileReader fileReader = new FileReader("nonexistent.txt");

// Predefined runtime exception - ArrayIndexOutOfBoundsException

int[] array = {1, 2, 3};

System.out.println(array[5]);

// Custom runtime exception - IllegalArgumentException

divideByZero(10, 0);

// Custom compile time exception - ClassNotFoundException

Class.forName("nonexistent.Class");

// Predefined runtime exception - NullPointerException

String str = null;

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

} catch (FileNotFoundException e) {

System.out.println("File not found: " + e.getMessage());

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Array index out of bounds: " + e.getMessage());

} catch (IllegalArgumentException e) {

System.out.println("Illegal argument: " + e.getMessage());

} catch (ClassNotFoundException e) {

System.out.println("Class not found: " + e.getMessage());

} catch (NullPointerException e) {

System.out.println("Null pointer exception: " + e.getMessage());

} finally {

System.out.println("Finally block executed");

}

}

public static void divideByZero(int numerator, int denominator) {

if (denominator == 0) {

throw new IllegalArgumentException("Denominator cannot be zero");

}

int result = numerator / denominator;

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

}

}

class LowBalanceException extends Exception {}

class NegativeNumberException extends Exception {}

class PasswordMismatchException extends Exception {}

class BankAccount {

private double balance;

public BankAccount(double balance) {

this.balance = balance;

}

public double getBalance() {

return balance;

}

public void balanceEnquiry() {

System.out.println("Account Balance: " + balance);

}

public void deposit(double amount) throws NegativeNumberException {

if (amount < 0) throw new NegativeNumberException();

balance += amount;

System.out.println("Deposit Successful. New balance: " + balance);

}

public void withdraw(double amount, String password) throws LowBalanceException, PasswordMismatchException, NegativeNumberException {

if (amount < 0) throw new NegativeNumberException();

if (!password.equals("password")) throw new PasswordMismatchException();

if (amount > balance) throw new LowBalanceException();

balance -= amount;

System.out.println("Withdrawal Successful. New balance: " + balance);

}

public void transfer(BankAccount recipient, double amount, String password) throws LowBalanceException, PasswordMismatchException, NegativeNumberException {

withdraw(amount, password);

recipient.deposit(amount);

System.out.println("Transfer Successful.");

}

}

public class Main {

public static void main(String[] args) {

BankAccount[] accounts = {

new BankAccount(1000.0),

new BankAccount(2000.0)

};

try {

accounts[0].balanceEnquiry();

accounts[0].deposit(500.0);

accounts[0].withdraw(200.0, "password");

accounts[0].transfer(accounts[1], 300.0, "password");

accounts[1].balanceEnquiry();

accounts[1].withdraw(500.0, "password");

} catch (NegativeNumberException | LowBalanceException | PasswordMismatchException e) {

System.out.println("Exception occurred.");

}

}

}

interface Stack {

int size = 5; // Variable size

void push(String item); // Abstract methods

String pop();

void display();

boolean overflow();

boolean underflow();

}

class IntegerStack implements Stack {

private int[] stack;

private int top;

public IntegerStack() {

stack = new int[size];

top = -1;

}

public void push(String item) {

int number = Integer.parseInt(item);

if (!overflow()) {

stack[++top] = number;

System.out.println("Pushed " + number + " into the IntegerStack.");

} else {

System.out.println("IntegerStack overflow.");

}

}

public String pop() {

if (!underflow()) {

int number = stack[top--];

return String.valueOf(number);

} else {

System.out.println("IntegerStack underflow.");

return null;

}

}

public void display() {

System.out.print("IntegerStack: ");

for (int i = top; i >= 0; i--) {

System.out.print(stack[i] + " ");

}

System.out.println();

}

public boolean overflow() {

return top == size - 1;

}

public boolean underflow() {

return top == -1;

}

}

class StringStack implements Stack {

private String[] stack;

private int top;

public StringStack() {

stack = new String[size];

top = -1;

}

public void push(String item) {

if (!overflow()) {

stack[++top] = item;

System.out.println("Pushed \"" + item + "\" into the StringStack.");

} else {

System.out.println("StringStack overflow.");

}

}

public String pop() {

if (!underflow()) {

return stack[top--];

} else {

System.out.println("StringStack underflow.");

return null;

}

}

public void display() {

System.out.print("StringStack: ");

for (int i = top; i >= 0; i--) {

System.out.print(stack[i] + " ");

}

System.out.println();

}

public boolean overflow() {

return top == size - 1;

}

public boolean underflow() {

return top == -1;

}

}

class DoubleStack implements Stack {

private double[] stack;

private int top;

public DoubleStack() {

stack = new double[size];

top = -1;

}

public void push(String item) {

double number = Double.parseDouble(item);

if (!overflow()) {

stack[++top] = number;

System.out.println("Pushed " + number + " into the DoubleStack.");

} else {

System.out.println("DoubleStack overflow.");

}

}

public String pop() {

if (!underflow()) {

double number = stack[top--];

return String.valueOf(number);

} else {

System.out.println("DoubleStack underflow.");

return null;

}

}

public void display() {

System.out.print("DoubleStack: ");

for (int i = top; i >= 0; i--) {

System.out.print(stack[i] + " ");

}

System.out.println();

}

public boolean overflow() {

return top == size - 1;

}

public boolean underflow() {

return top == -1;

}

}

public class Main {

public static void main(String[] args) {

IntegerStack intStack = new IntegerStack();

StringStack strStack = new StringStack();

DoubleStack doubleStack = new DoubleStack();

intStack.push("10");

intStack.push("20");

intStack.push("30");

intStack.display();

System.out.println("Popped: " + intStack.pop());

intStack.display();

strStack.push("Hello");

strStack.push("World");

strStack.display();

System.out.println("Popped: " + strStack.pop());

strStack.display();

doubleStack.push("1.5");

doubleStack.push("2.75");

doubleStack.display();

System.out.println("Popped: " + doubleStack.pop());

doubleStack.display();

}

}

// Main package: math

package math;

import java.util.Arrays;

public class Statistics {

public static double mean(double[] numbers) {

double sum = 0.0;

for (double num : numbers) {

sum += num;

}

return sum / numbers.length;

}

public static double median(double[] numbers) {

Arrays.sort(numbers);

int middle = numbers.length / 2;

if (numbers.length % 2 == 0) {

double median1 = numbers[middle - 1];

double median2 = numbers[middle];

return (median1 + median2) / 2.0;

} else {

return numbers[middle];

}

}

public static double average(double[] numbers) {

double sum = 0.0;

for (double num : numbers) {

sum += num;

}

return sum / numbers.length;

}

public static double standardDeviation(double[] numbers) {

double mean = mean(numbers);

double sum = 0.0;

for (double num : numbers) {

sum += Math.pow(num - mean, 2);

}

return Math.sqrt(sum / numbers.length);

}

}

// Subpackage: convert

package math.convert;

public class NumberConverter {

public static String decimalToBinary(int decimal) {

return Integer.toBinaryString(decimal);

}

public static String decimalToOctal(int decimal) {

return Integer.toOctalString(decimal);

}

public static String decimalToHex(int decimal) {

return Integer.toHexString(decimal);

}

public static int binaryToDecimal(String binary) {

return Integer.parseInt(binary, 2);

}

public static int octalToDecimal(String octal) {

return Integer.parseInt(octal, 8);

}

public static int hexToDecimal(String hex) {

return Integer.parseInt(hex, 16);

}

}

// Application program

public class Main {

public static void main(String[] args) {

double[] numbers = { 2.5, 7.8, 1.3, 9.6, 5.4 };

System.out.println("Mean: " + math.Statistics.mean(numbers));

System.out.println("Median: " + math.Statistics.median(numbers));

System.out.println("Average: " + math.Statistics.average(numbers));

System.out.println("Standard Deviation: " + math.Statistics.standardDeviation(numbers));

int decimal = 42;

System.out.println("Decimal to Binary: " + math.convert.NumberConverter.decimalToBinary(decimal));

System.out.println("Decimal to Octal: " + math.convert.NumberConverter.decimalToOctal(decimal));

System.out.println("Decimal to Hex: " + math.convert.NumberConverter.decimalToHex(decimal));

String binary = "101010";

System.out.println("Binary to Decimal: " + math.convert.NumberConverter.binaryToDecimal(binary));

String octal = "52";

System.out.println("Octal to Decimal: " + math.convert.NumberConverter.octalToDecimal(octal));

String hex = "2A";

System.out.println("Hex to Decimal: " + math.convert.NumberConverter.hexToDecimal(hex));

}

}