**一：**

interface Compute {

int computer(int n, int m);

}

class Add implements Compute {

public int computer(int n, int m) {

return n + m;

}

}

class Subtract implements Compute {

public int computer(int n, int m) {

return n - m;

}

}

class Multiply implements Compute {

public int computer(int n, int m) {

return n \* m;

}

}

class Divide implements Compute {

public int computer(int n, int m) {

if (m == 0) {

throw new IllegalArgumentException("除数不能为零");

}

return n / m;

}

}

class UseCompute {

public void useCom(Compute com, int one, int two) {

int result = com.computer(one, two);

System.out.println(result);

}

}

public class Test {

public static void main(String[] args) {

UseCompute uc = new UseCompute();

uc.useCom(new Add(), 3, 5); // 输出8

uc.useCom(new Subtract(), 7, 2); // 输出5

uc.useCom(new Multiply(), 4, 6); // 输出24

uc.useCom(new Divide(), 10, 2); // 输出5

uc.useCom(new Divide(), 10, 0); // 抛出异常：除数不能为零

}

}

**二：**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("请输入分数：");

int score = scanner.nextInt();

try {

if (score >= 0 && score <= 100) {

System.out.println("您的分数是：" + score);

} else {

throw new IllegalArgumentException("分数必须在0-100之间");

}

} catch (IllegalArgumentException e) {

System.out.println(e.getMessage());

}

}

}

**三：**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int n = 0;

double sum = 0;

System.out.print("请输入N的值：");

while (true) {

try {

n = scanner.nextInt();

if (n < 0) {

throw new IllegalArgumentException("N必须是正数或者0，请重新输入：");

}

break;

} catch (IllegalArgumentException e) {

System.out.print(e.getMessage());

}

}

for (int i = 1; i <= n; i++) {

System.out.print("请输入第" + i + "个整数：");

int x = scanner.nextInt();

sum += x;

}

double average = sum / n;

System.out.println("这" + n + "个整数的平均值为：" + average);

}

}

**四：**

abstract class Employee {

private String name;

private String number;

private MyDate birthday;

public Employee(String name, String number, MyDate birthday) {

this.name = name;

this.number = number;

this.birthday = birthday;

}

public abstract double earnings();

@Override

public String toString() {

return "姓名：" + this.name + "，工号：" + this.number + "，出生日期：" + this.birthday.toString();

}

}

class MyDate {

private int year;

private int month;

private int day;

public MyDate(int year, int month, int day) {

this.year = year;

this.month = month;

this.day = day;

}

@Override

public String toString() {

return year + "年" + month + "月" + day + "日";

}

}

class SalariedEmployee extends Employee {

private double monthlySalary;

public SalariedEmployee(String name, String number, MyDate birthday, double monthlySalary) {

super(name, number, birthday);

this.monthlySalary = monthlySalary;

}

public double earnings() {

return monthlySalary;

}

}

class HourlyEmployee extends Employee {

private double wage;

private double hour;

public HourlyEmployee(String name, String number, MyDate birthday, double wage, double hour) {

super(name, number, birthday);

this.wage = wage;

this.hour = hour;

}

public double earnings() {

return wage \* hour;

}

}

public class Main {

public static void main(String[] args) {

Employee e1 = new SalariedEmployee("张三", "001", new MyDate(1990, 12, 1), 10000);

Employee e2 = new HourlyEmployee("李四", "002", new MyDate(1995, 6, 10), 50, 200);

System.out.println(e1.toString() + "，月薪为：" + e1.earnings());

System.out.println(e2.toString() + "，工资为：" + e2.earnings());

}

}

**五：**

public class Solution {

public int numMatchingSubseq(String S, String[] words) {

int count = 0;

for (String word : words) {

if (isSubsequence(word, S)) {

count++;

}

}

return count;

}

private boolean isSubsequence(String word, String s) {

if (word.length() > s.length()) {

return false;

}

int i = 0, j = 0;

while (i < word.length() && j < s.length()) {

if (word.charAt(i) == s.charAt(j)) {

i++;

}

j++;

}

return i == word.length();

}

}