1-> Create 1st class, where a class has one method that takes 2 integers as parameters and returns an integer denoting their sum.

Create 2nd class that inherits from a superclass of the first class.

Again 3rd class and here print the output of the first class method with the help of 2nd class object.

class First{

public int sum(int a,int b){

return a+b;

}

}

class Second extends First{

}

public class Inher{

public static void main(String[] args){

Second s = new Second();

int c = s.sum(2,4);

System.out.println(c);

}

}

2-> Create an interface that contains a method, where the method return type is int and the method has one int type parameter. You need to write another class that implements the interface.

And your function just takes an integer as input and return the sum of all its divisors. For example divisors of 8 are 1, 2, 4 and 8, so divisor addition should return 15. The value of n will be at most 1000.

interface Iface{

int sodiv(int n);

}

public class Inter implements Iface{

public int sodiv(int n){

int sum = 0;

int i;

for(i=1;i\*i<=n;i++){

if(n%i==0){

sum = sum+i+(n/i);

}

}

if(i\*i==n) sum = sum - i;

return sum;

}

public static void main(String[] args){

Inter a = new Inter();

System.out.println(a.sodiv(8));

}

}

3-> Write a program to sort the given array of String in descending order. Your string value should be the same during the output. Like- .12 printed as .12 and 0.12 printed as 0.12

Ex-> String[] str = {“-100”,”50”,”.12”,”0.12”,”0”,”000.000”};

import java.util.\*;

class Pair {

int x;

Float y;

public Pair(int x, Float y) {

this.x = x;

this.y = y;

}

}

public class Sort{

public static void main(String[] args){

String[] arr = {"-100","50",".12","0.12","0","000.000"};

Pair[] newarr = new Pair[6];

for(int i=0;i<6;i++){

try {

float j=Float.parseFloat(arr[i]);

newarr[i] = new Pair(i,j);

} catch (NumberFormatException e) {

System.out.println(e);

}

}

Comparator<Pair> comparator = new Comparator<>() {

//@Override

public int compare(Pair p1, Pair p2) {

return (p2.y>p1.y)?1:-1;

}

};

Arrays.sort(newarr, comparator);

for(int i=0;i<6;i++){

System.out.println(arr[newarr[i].x]);

}

}

}

4-> You will be given two integers a and b as input, you have to compute a/b.

If a and b are not 32 bit signed integers or if b is zero,

exception will occur and you have to report it.

Read sample Input/Output to know what to report in case of exceptions.

Sample Input 0:

10

3

Sample Output 0:

3

Sample Input 1:

10

Hello

Sample Output 1:

java.util.InputMismatchException

Sample Input 2:

10

0

Sample Output 2:

java.lang.ArithmeticException: / by zero

Sample Input 3:

23.323

0

Sample Output 3:

java.util.InputMismatchException

Sample Input 4:

2147483648

55

Sample Output 4:

java.util.InputMismatchException

import java.util.\*;

public class Divide{

public static void div(){

Scanner sc = new Scanner(System.in);

try{

System.out.print("Please Enter the Integer value of a: ");

int a = sc.nextInt();

System.out.print("Please Enter the Integer value of b: ");

int b = sc.nextInt();

int c = a/b;

System.out.println(c);

}

catch(InputMismatchException e){

System.out.println(e);

}

catch(ArithmeticException e){

System.out.println(e);

}

}

public static void main(String[] args){

div();

}

}

5->

Write a Java program to create a class known as "SBIBankAccount"

with methods called deposit() and withdraw().

And create opening account methos openAccount.

create sub classes for premiumAccount and jundhanAccount and override the deposit,

withdraw, and openAccount in sub class.

Where if you open premium account should be start from 5K and your jundhanAccount

start from 0 balance.

where withdraw() method to prevent withdrawals if the account balance falls

below 5K for the premium account and prevent withdrawals if the account balance falls

below 0 for the jundhanAccount.

import java.util.\*;

class SBIBankAccount{

protected int balance;

SBIBankAccount(int val){

openAccount(val);

}

private void openAccount(int val){

this.balance = val;

}

public int getBalance(){

return this.balance;

}

public void deposit(int val){}

public void withdraw(int val){}

}

class PremiumAccount extends SBIBankAccount{

PremiumAccount(int val){

super(val);

}

public void deposit(int val){

this.balance+=val;

System.out.println(val+" is deposited on your account");

}

public void withdraw(int val){

if(this.balance-val<5000){

int amt = this.balance-5000;

System.out.println("You can't withdraw "+amt+" above");

}

else{

this.balance-=val;

System.out.println(val+" is withdrawn from your account");

}

}

}

class JundhanAccount extends SBIBankAccount{

JundhanAccount(int val){

super(val);

}

public void deposit(int val){

this.balance+=val;

System.out.println(val+" is deposited on your account");

}

public void withdraw(int val){

if(this.balance-val<0){

int amt = this.balance-0;

System.out.println("You can't withdraw "+amt+" above");

}

else{

this.balance-=val;

System.out.println(val+" is withdrawn from your account");

}

}

}

public class Bank{

public static void main(String[] args){

System.out.println("1. PremiumAccount");

System.out.println("2. JundhanAccount");

System.out.print("Enter the type of Account you want: ");

Scanner sc = new Scanner(System.in);

int typeOfAccount = sc.nextInt();

System.out.print("Enter the Initial Amount you want to Deposit: ");

int val = sc.nextInt();

if(typeOfAccount==1){

if(val>=5000){

PremiumAccount b = new PremiumAccount(val);

System.out.println("Your Account Balance is: "+b.getBalance());

}

else{

System.out.println("Your Account should have a minimum deposit of 5000");

}

}

else{

JundhanAccount b = new JundhanAccount(val);

System.out.println("Your Account Balance is: "+b.getBalance());

}

}

}

16- Write a Java programming to create a banking system with three classes -

Bank, Account, SavingsAccount, and CurrentAccount. The bank should have a list of

accounts and methods for adding them. Accounts should be an interface with methods

to deposit, withdraw, calculate interest, and view balances. SavingsAccount and

CurrentAccount should implement the Account interface and have their own unique

methods.

interface Account{

public void deposit(int val);

public void withdraw(int val);

public Float interest();

public int getBalance();

}

class SavingsAccount implements Account{

private int balance;

public int minBalance = 2000;

SavingsAccount(int val){

this.balance = val;

}

public void deposit(int val){

this.balance+=val;

}

public void withdraw(int val){

if(this.balance-val>=this.minBalance){

this.balance-=val;

}

else{

System.out.println("Your Account should have a minimum balance of "+this.minBalance);

}

}

public Float interest(){

return 0.0f;

}

public int getBalance(){

return this.balance;

}

}

class CurrentAccount implements Account{

private int balance;

public int minBalance = 10000;

public Float rate = 10.0f;

CurrentAccount(int val){

this.balance = val;

}

public void deposit(int val){

this.balance+=val;

}

public void withdraw(int val){

this.balance-=val;

}

public Float interest(){

int time = 12;

return ((this.minBalance-this.balance)\*rate)/time;

}

public int getBalance(){

return this.balance;

}

}

public class Banks{

public static void main(String[] args){

SavingsAccount a = new SavingsAccount(3000);

System.out.println("Your Account Balance is: "+a.getBalance());

}

}