Exception handling Lab Questions

**Sanjana Kumbhar**

**PGDAC (Aug2024)**

Q 1 Take two integers and as input, you have to compute x/y . If x and y are not integers or if is zero , exception will occur and you have to report it. Read sample Input/Output to know what to report in case of exceptions.

**package** assign4;

**import** java.util.\*;

**public** **class** ExcepOne {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** x, y;

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter value of x");

**while**(**true**){

**try** {

x= sc.nextInt();

**break**;

}**catch**(InputMismatchException e) {

System.***out***.println("Input mismatched enter valid input");

sc.next();

}

}

System.***out***.println("Enter value of y");

**while**(**true**){

**try** {

y= sc.nextInt();

**if**(y==0) **throw** **new** ArithmeticException();

**break**;

}**catch**(InputMismatchException e) {

System.***out***.println("Input mismatched enter valid input");

sc.next();

}**catch**(ArithmeticException e) {

System.***out***.println("Divide by zero");

}

}

System.***out***.println("Result: " + (x / y));

}

}

Output-

Enter value of x

22

Enter value of y

2

Result: 11

Enter value of x

0

Enter value of y

2

Result: 0

Enter value of x

2

Enter value of y

0

Divide by zero

Enter value of x

w

Input mismatched enter valid input

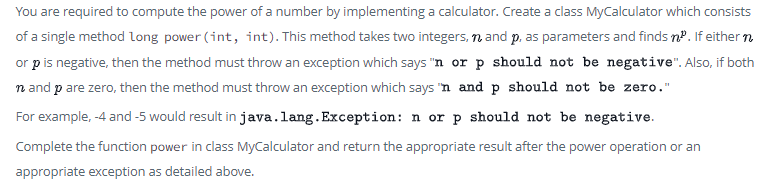
Enter value of x

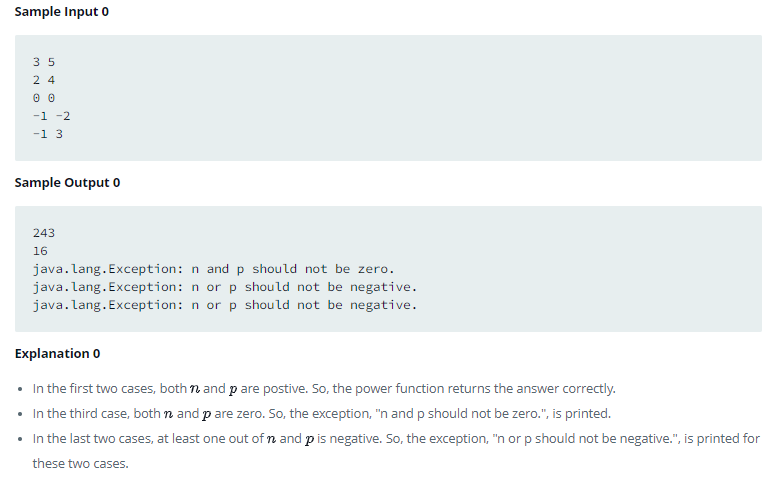
2

Enter value of y

e

Input mismatched enter valid input

Q 2 



**package** assign4;

**import** java.util.\*;

**public** **class** ExcepOne {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** x, y;

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter value of x");

**while**(**true**){

**try** {

x= sc.nextInt();

**break**;

}**catch**(InputMismatchException e) {

System.***out***.println("Input mismatched enter valid input");

sc.next();

}

}

System.***out***.println("Enter value of y");

**while**(**true**){

**try** {

y= sc.nextInt();

**if**(y==0) **throw** **new** ArithmeticException();

**break**;

}**catch**(InputMismatchException e) {

System.***out***.println("Input mismatched enter valid input");

sc.next();

}**catch**(ArithmeticException e) {

System.***out***.println("Divide by zero");

}

}

System.***out***.println("Result: " + (x / y));

sc.close();

}

}

Output-

4.0

java.lang.Exception: n and p should not be zero

0.0

java.lang.Exception: n and p should not be zero

0.0

java.lang.Exception: n and p should not be negative

0.0

java.lang.Exception: n and p should not be negative

0.0

Q 3 Write a program for user defined Exception that checks the external and internal marks if the internal marks is greater than 40 it raise the exception internal mark is exceed, if the external mark is greater than 60 exception is raised and display the message the external marks is exceed, create the above exception and use it in your program.

**package** assign4;

**class** InternalException **extends** Exception{

InternalException(){

**super**("Internal mark is exceed");

}

}

**class** ExternalException **extends** Exception{

ExternalException(){

**super**("external marks is exceed");

}

}

**public** **class** Marks {

**int** emarks=0;

**int** imarks=0;

**void** check(**int** i, **int** e ) {

emarks=e;

imarks=i;

**try** {

**if**(imarks>40) {

**throw** **new** InternalException();

}

**if**(emarks>60) {

**throw** **new** ExternalException();

}

}**catch**(Exception d) {

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

}

System.***out***.println("Internal marks -"+imarks+" External Marks -"+emarks);

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Marks m1 = **new** Marks();

m1.check(30, 50);

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

Marks m2 = **new** Marks();

m2.check(50, 50);

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

Marks m3 = **new** Marks();

m3.check(30, 80);

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

Marks m4 = **new** Marks();

m4.check(50, 90);

}

}

Output-

Internal marks -30 External Marks -50

-------------------------------------------

assign4.InternalException: Internal mark is exceed

Internal marks -50 External Marks -50

-------------------------------------------

assign4.ExternalException: external marks is exceed

Internal marks -30 External Marks -80

-------------------------------------------

assign4.InternalException: Internal mark is exceed

Internal marks -50 External Marks -90

Q 4 Create a class Student with attributes roll no, name, age and course. Initialize values through parameterized constructor. If age of student is not in between 15 and 21 then generate user-defined exception “AgeNotWithinRangeException”. If name contains numbers or special symbols raise exception” NameNotValidException ”.

**package** assign4;

**class** AgeNotWithinRangeException **extends** Exception{

AgeNotWithinRangeException(){

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

}

}

**class** NameNotValidException **extends** Exception{

NameNotValidException(){

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

}

}

**public** **class** Student {

**int** roll\_no;

String name;

**int** age;

String course;

Student(**int** r,String n, **int** a,String c){

roll\_no=r;

name=n;

age=a;

course=c;

}

**void** CheckDetails() {

**try** {

**if**(age<15 || age>21) {

**throw** **new** AgeNotWithinRangeException();

}

**if**(!name.matches("[a-zA-Z]+")) {

**throw** **new** NameNotValidException();

}

}**catch**(Exception d) {

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

}

System.***out***.println("name -"+name+" roll number -"+roll\_no+" age-"+age+" course -"+course);

}

**public** **static** **void** main(String[] args) {

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

Student s1= **new** Student(1,"sam",20,"PGDAC");

s1.CheckDetails();

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

Student s2= **new** Student(1,"tom",24,"PGDAC");

s2.CheckDetails();

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

Student s3= **new** Student(1,"ram1@",24,"PGDAC");

s3.CheckDetails();

}

}

Output-

-------------------------------------------

name -sam roll number -1 age-20 course -PGDAC

-------------------------------------------

AgeNotWithinRangeException

assign4.AgeNotWithinRangeException

name -tom roll number -1 age-24 course -PGDAC

-------------------------------------------

AgeNotWithinRangeException

assign4.AgeNotWithinRangeException

name -ram1@ roll number -1 age-24 course -PGDAC

Q 5 Write a program to check all the three number entered by command line argument are greater than 10 , then print sum of those numbers . If any number is less then 10 then throw user defined exception “MyException” which print message number iis lesser then 10 “

**package** assign4;

**class** MyException1 **extends** Exception{

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

MyException1(String message){

**super**(message);

}

}

**public** **class** NumCheck1 {

**public** **static** **void** main(String[] args) {

**try** {

**int** i;

**int** sum=0;

**for**(String s:args) {

i=Integer.*parseInt*(s);

**if**(i>10) {

sum=sum+i;

}**else** {

**throw** **new** MyException1("Number is less than 10");

}

System.***out***.println("sum is "+sum);

}

}**catch**(MyException1 e){

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

}

}

}

Input-20 30 6

Output- Number is less than 10

Input – 20 30 40

Output- sum is 90

Q 6 create a program to demonstrate constructor channing .

**package** assign4;

**class** costruct1{

costruct1(){

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

}

costruct1(**int** a){

System.***out***.println("value of a is "+a);

}

}

**class** costruct2 **extends** costruct1 {

costruct2(){

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

}

costruct2(**int** b){

**super**(5);

System.***out***.println("value of b is"+b);

}

}

**public** **class** ConstChaining {

**public** **static** **void** main(String[] args) {

**new** costruct2();

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

**new** costruct2(20);

}

}

Output-

hello

world

---------

value of a is 5

value of b is20

Q 7 Write a program that define interface Admission having abstract method registration

Create another class Student in separate file having method Addstudent ()

1. In Addstudent create local class Mtech student that inherits Admission interface
2. In same method also create anonymous class that also inherits Admission interface

In both above classes implement registration method

In main call AddStudent method of student class.

**interface** Admission{

**void** registration();

}

**class** Student1{

**void** Addstudent() {

System.***out***.println("add student");

**class** Mtech **implements** Admission{

**public** **void** registration() {

System.***out***.println("student1 -registration");

}

}

Mtech m1=**new** Mtech();

m1.registration();

**new** Admission(){

**public** **void** registration() {

System.***out***.println("student1 -registration2");

}

}.registration();

}

}

**public** **class** Adm {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Student1 s1=**new** Student1();

s1.Addstudent();

}

}

Output-

add student

student1 -registration

student1 -registration2

Q 8 Implement a Java program to read an integer from the user and calculate its square root. Handle the InputMismatchException if the user enters a non-integer value.

**package** assign4;

**import** java.util.Scanner;

**public** **class** SquareRoot {

**double** num=0;

**void** sqrt() {

**try** {

System.***out***.println("Enter number");

Scanner sc = **new** Scanner(System.***in***);

num = sc.nextDouble();

**double** sqrt= Math.*sqrt*(num);

System.***out***.println("square root is "+sqrt);

}**catch**(NumberFormatException e) {

System.***out***.println("sorry ! enter corret number");

}

}

**public** **static** **void** main(String[] args) {

SquareRoot s1= **new** SquareRoot();

s1.sqrt();

}

}

Output-

Enter number

16

square root is 4.0

------------------------------------------

Enter number

a

Exception in thread "main" java.util.InputMismatchException

at java.base/java.util.Scanner.throwFor(Scanner.java:947)

at java.base/java.util.Scanner.next(Scanner.java:1602)

at java.base/java.util.Scanner.nextDouble(Scanner.java:2573)

at assign4.SquareRoot.sqrt(SquareRoot.java:10)

at assign4.SquareRoot.main(SquareRoot.java:19)

Q 9 Write a Java program to read an integer array from the user and calculate the average of its elements. Handle the NumberFormatException if the user enters a non-integer value.

**package** assign4;

**import** java.util.\*;

**public** **class** avg {

**int** a,len;

**int**[] arr1;

**void** input(){

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Length of array");

len = sc.nextInt();

arr1=**new** **int**[len];

System.***out***.println("Enter elements of array");

**for**(**int** i=0;i<len;i++) {

**try** {

arr1[i]=sc.nextInt();

}**catch**(InputMismatchException e) {

System.***out***.println("Invalid input. Please enter an integer.");

sc.next(); // Clear the invalid input

}

}

}

**void** average() {

**int** avg=0;

**for**(**int** i=0;i<len;i++) {

avg+=arr1[i];

}

System.***out***.print(avg);

}

**public** **static** **void** main(String[] args) {

avg a= **new** avg();

a.input();

a.average();

}

}

Output-

Length of array

4

Enter elements of array

2 3 f 4

Invalid input. Please enter an integer.

9

Q 10 Develop a Java program to read a string from the user and convert it into an integer. Handle the NumberFormatException if the string cannot be converted to an integer.

**package** assign4;

//Develop a Java program to read a string from the user and convert it into an integer. Handle the NumberFormatException if the string cannot be converted to an integer.

**import** java.util.\*;

**public** **class** Convert {

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

String s;

System.***out***.println("Enter string : ");

String S=sc.nextLine();

**try** {

**int** a= Integer.*parseInt*(S);

}**catch**(NumberFormatException e) {

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

}

}

}

Output-

Enter string :

aa

NumberFormatException

Q 11 You are tasked with implementing a Java program to manage bank accounts. Each bank account has an account number, balance, and account holder name. The program should support deposit, withdrawal, and balance inquiry operations.

Input

createAccount 123 John 1000

deposit 123 500

withdraw 123 200

balance 123

output

Balance for account 123: 1300

**package** assign4.Bank;

**import** java.util.Scanner;

**public** **class** BankAccount {

**int** anum;

String name;

**double** blncA;

Scanner sc = **new** Scanner(System.***in***);

**void** createA(**int** a, String s,Double b) {

name = s;

anum=a;

blncA=b;

}

**void** depositA(**int** a, **double** depoA) {

**if**(a==anum) {

blncA+=depoA;

}**else** {

System.***out***.println("Account Number Mismatch !");

}

}

**void** withdrawA(**int** a,**double** withA) {

**if**(a==anum) {

blncA-=withA;

}**else** {

System.***out***.println("Account Number Mismatch !");

}

}

**void** balanceA(**int** a) {

**if**(a==anum) {

System.***out***.println(anum+" : "+blncA);

}**else** {

System.***out***.println("Account Number Mismatch !");

}

}

**public** **static** **void** main(String[] args) {

BankAccount a= **new** BankAccount();

a.createA(123, "John", 1000.000);

a.depositA(123, 500);

a.withdrawA(123, 200);

a.balanceA(123);

}

}

Output-

123 : 1300.0

—---------------------MCQ—------------

1 What is the purpose of the finally block in Java exception handling?

A) To catch exceptions and handle them appropriately.

B) To execute a block of code regardless of whether an exception is thrown or not.

C) To specify the exceptions that a method might throw.

D) To define custom exception classes.

Answer: B) To execute a block of code regardless of whether an exception is thrown or not.

Which of the following statements about checked and unchecked exceptions in Java is true?

A) Checked exceptions are subclasses of RuntimeException.

B) Unchecked exceptions must be explicitly caught or declared in the method signature using the throws keyword.

C) Checked exceptions are always caught at compile-time.

D) Unchecked exceptions are subclasses of Exception.

Answer: C) Checked exceptions are always caught at compile-time.

What happens if an exception is thrown inside a try block, but no catch block matches its type?

A) The program crashes.

B) The exception is caught by the finally block.

C) The program continues execution after the try block.

D) The exception is propagated up the call stack.

Answer: D) The exception is propagated up the call stack.

Which keyword is used to explicitly throw an exception in Java?

A) try

B) throw

C) catch

D) finally

Answer: B) throw

What does the throws keyword indicate in a method signature?

A) That the method must be called within a try block.

B) That the method throws an exception.

C) That the method handles exceptions internally.

D) That the method is overloaded.

Answer: B) That the method throws an exception.

What happens if a method declares that it throws a checked exception, but no exception of that type is thrown within the method?

A) The code will not compile.

B) The method will throw a NullPointerException.

C) The method will throw an ArrayIndexOutOfBoundsException.

D) The code will compile without any issues.

Answer: D) The code will compile without any issues.

Which of the following is NOT a valid way to handle exceptions in Java?

A) Using a try block followed by a catch block.

B) Using a try block followed by multiple catch blocks.

C) Using a try block followed by a finally block.

D) Using a try block followed by a throw statement.

Answer: D) Using a try block followed by a throw statement.

What does the printStackTrace() method do when called on an exception object in Java?

A) Prints the exception message to the console.

B) Prints the entire stack trace of the exception to the console.

C) Throws the exception to the console.

D) Clears the exception object.

Answer: B) Prints the entire stack trace of the exception to the console.

In Java, which keyword is used to create a custom exception class?

A) try

B) catch

C) throw

D) class

Answer: D) class