Day 4 :  
**exception handling :**

package day4;

public class task2 {

public static void main(String args[]) {

int a;

System.***out***.println("before error");

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

int b=a/0;

System.***out***.println("after error");

}

}

The local variable a may not have been initialized

The above code catches none initialize during the compile time and those can be handle by the experts and then deployed however there are instance when error occurs during the run time which must me execution after deploying .

package day4;

public class task2 {

public static void main(String args[]) {

int a=10;

System.***out***.println("before error");

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

int b=a/0;

System.***out***.println("after error");

}

Exception in thread "main" java.lang.ArithmeticException:

The above code arithmetic exception is only during the run time and not compile time ( before statement is executed )

This kind of error handle by the programmer using exception handling mechanism

package day4;

public class task2 {

public static void main(String args[]) {

int a=10;

System.***out***.println("before error");

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

try {

int b=a/0; }

catch(ArithmeticException e){

System.***out***.println("please provide the valid number not 0");

}

System.***out***.println("after error");

}}

before error

please provide the valid number not 0

after error

the above code since the run time exception is handle using try block the line after the exception also run fine without break…

package day4;

class Second{

void firstcall()

{

secondcall();

void secondcall() {

System.***out***.println("second call block");

int b=5/0;

}

}

public class task3 {

public static void main(String args[]) {

Second s = new Second();

s.firstcall();

}

}

second call block

Exception in thread "main" java.lang.ArithmeticException: / by zero

at javatraining055/day4.Second.secondcall(task3.java:10)

at javatraining055/day4.Second.firstcall(task3.java:6)

at javatraining055/day4.task3.main(task3.java:18)

exception handling mechanism always takes place in an order of stack trace ( LIFO )

when the error is spotted is immediatly report to the higher body that is the current activation record in the stack trace if the current activation record is not capable handling the error is report to the next higher level body in the hierarchy.

When the none of the higher body able to resolve the exception to report to the main methos which will pass to the JVM to take action .

package day4;

import java.util.Scanner;

import java.util.InputMisMatchException;

class Second{

void firstcall()

{

secondcall();

}

void secondcall() {

int c;

System.***out***.println("second call block");

try {

int b=5/0;

c=Scanner.nextInt();

}

catch(ArithmeticException e)

{

e.printStackTrace();

}

catch(InputMisMatchException in)

{

in.printStackTrace();

}

}

}

public class task4 {

public static void main(String args[]) {

Second s = new Second();

s.firstcall();

}

A try may have any no of catch block associated but any given point of time the try block handle or uses only one catch block

The choosen catch will depends on the first exception that is call by the try block

package day4;

import java.util.Scanner;

import java.util.\*;

class Second1{

int arr[]= {1,45,23,4};

void firstcall()

{

secondcall();

}

void secondcall() {

int c;

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

System.***out***.println("second call block");

try {

int b=5/6;

c=s.nextInt();

System.***out***.println(arr[9]);

}

catch(ArithmeticException e)

{

e.printStackTrace();

}

catch(InputMismatchException in)

{

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

}

catch(Exception exe)

{

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

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

}

}

}

public class task4 {

public static void main(String args[]) {

Second1 s = new Second1();

s.firstcall();

}

}

Array bounded index is raised this is from the Exception genral catch block :  
Index 9 out of bounds for length 4

exception

**\*\*\*if the multiple catch is occur in the same class use the 🡪 starting with specific exception to finally generic .**

**Ex :**

catch(InputMismatchException in)

{

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

}

catch(Exception exe)

{

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

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

}

**Finally :**

finally {

System.***out***.println("finally statement "); // this is executed when error occur or not

}

**RETURN :**

try {

int b=5/0;

c=s.nextInt();

//System.out.println(arr[9]);

}

catch(ArithmeticException e)

{

e.printStackTrace();

return 1;

}

**Its return 1 because the arithmetic exception is occur**

package day4;

import java.util.InputMismatchException;

import java.util.Scanner;

class MyException extends Exception{

public MyException(String msg)

{

super(msg);

}

}

class First{

int arr[]= {1,45,23,4};

void firstcall()

{

secondcall();

}

void secondcall() {

int c;

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

System.***out***.println("second call block");

try {

int b=5/8;

c=s.nextInt();

//System.out.println(arr[9]);

throw new MyException("customer exception");

}

catch(ArithmeticException e)

{

e.printStackTrace();

}

catch(InputMismatchException in)

{

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

}

catch(Exception exe)

{

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

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

}

finally {

System.***out***.println("finally statement "); // this is executed when error occur or not

}

}

}

public class task5 {

public static void main(String args[])

{

First f = new First();

f.firstcall();

}

}

second call block

12

day4.MyException: customer exception

exception

finally statement

The user define exception class should extends the inbuild Exception class.

class Stock

{

static int *stockvalue*=10;

public static void upStock(int order)

{

try {

if(*stockvalue*<3)

{

throw new ArithmeticException();

}

}

catch(ArithmeticException e){

System.***out***.println("please restock the inventry ");

}

if(order>*stockvalue*)

{

System.***out***.println("out of stock");

}

}

}

public class task6 {

public static void main(String args[])

{

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

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

int order=s.nextInt();

Stock st = new Stock();

Stock.*upStock*(order);

}