# Exception Handling

Exception Handling is mainly used to handle the checked exceptions.if there occurs nay unchecked exception such as **NullPointerException**,it is programmers fault that he is not performing check up before the code being used.

**Q-1: in which case you will use try-catch and in which case throws**

**Q-2: Write the name of 5 checked exception**

**A:** 1-Exception

2-IOException

3-FileNotFoundException

4-ClassNotFoundException

5-NoSuchMethodException

**Q-3: Write the name of 5 unchecked exception and Explain wirh**

**A:** 1-ArithmeticException

2-NullPointerException

3-ArrayIndexOutOfBoundsException

4-NumberFormatException

5-ClassCastException

**Q-4: Write the name of 5 Error.**

**A:** 1-OutOfMemoryError

2-StackOverFlowError

3-VirtualMachineError

4-NoClassDefFoundError

5-NoSuchMethodError

**Q-5: Explain the Some Common Scenarios of Exception**

**A:** Some Common Scenarios of are as

1-> int a =50/0; // ArithmeticException

2->String s =null;

System.out.println(s.length()); // NullPointerException

3->String s = "abc";

int i =Integer.parseInt(s); // NumberFormatException

4->int a[] = new int[5];

a[10]=50; // ArrayIndexOutOfBoundsException

5- > TreeSet t1 = new TreeSet();

t1.add(12);

t1.add(“ABC”); // ClassCastException

**Q-6: Explain the following error**

**A:** **1-NoSuchMethodError->**

If main() method not found upto jdk1.6

**2-NoClassDefFoundError->**

If .class file is not there while executing

**3-StackOverFlowError->**

In recursive method call , if memeory is not available in stack.

**4-OutOfMemoryError->**

While creating the object , if there is no memory in heap

Int arr[] = new int[453453];

**5-VirtualMachineError->**

If there is a problem in starting jvm

**Q-7: What is the purpose of throw**

**A: 1-** >throw is a keyword

**2->** it is used to throw the exception explicitly

**3->** you can throw any checked or unchecked exception

**4->** basically it is used to generate custom exception

**5->** Sometimes we can create Exception object explicitly and we can

handover that exception object to the JVM explicitly by throw

keyword.The purpose of throw keyword is to handover our created

exception object explicitly to the JVM.

**Ex:**

class Bar

{

int age;

public void saveUserAge(int age) {

if(age<18){

throw new ArithmeticException();

}

else{

System.out.println("Correct are is entered");

}

}

public static void main(String[] args) {

Bar b1 = new Bar();

try

{

b1.saveUserAge(15);

}

catch(ArithmeticException e)

{

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

}

}

}

**Ex:2**

class Demo{

int sum(int num1, int num2)

{

if (num2 == 0)

throw new ArithmeticException("second parameter is not valid");

else

System.out.println("Both parameters are correct!!");

return num1/num2;

}

public static void main(String args[])

{

Demo d1 = new Demo();

int res=d1.sum(12,0);

System.out.println(res);

}

}

**Q-8: How to create custom exception. Write the code also.**

**A:Creation of Custom Exception**

1- if you are creating your own exception known as custom or user define exception.

2- if you want to write custom exception it must be inherit

the Exception class

**Ex:**

public class MyException extends Exception {

static int accno[]={100,101,102,103,104,105};

static String name[]={"raja","rama","subh","anay","lakshmi"};

static long bal[]={1000,12000,1335,999,3500};

public static void main(String[] args) {

try

{

System.out.println("ACCNO"+"\t"+"CUSTOMER"+"\t"+"BALANCE");

for(int i=0;i<5;i++)

{

System.out.println(accno[i]+"\t"+name[i]+"\t"+bal[i]);

if(bal[i]<1000)

{

MyException me = new MyException();

throw me;

}

}

}

catch(MyException me)

{

System.out.println("balance amount is less");

me.printStackTrace();

}

}

}

**Q-9: What is the purpose of throws**

**A: 1-** >throw is a keyword

**2->it**  is used to declare an exception. It gives an information to the programmer that there may occur an exception so it is better for the programmer to provide the exception handling code so that normal flow can be maintained.

**3->** you can define any checked or unchecked exception

**4->** you can define any Built-in or custom exception at method level

**5->** when the exception is unchecked exception then throws keyword is

Optional but for the checked exception throws keyword is mandatory

**6->** The main purpose of throws keyword is to give the responsibilities of

exception handling to the caller. It requires in the case of checked

exception.

**Note:** in case of of checked you have to compulsory mention the throws but in case of unchecked no need of throws

import java.io.\*;

class Example

{

void mymethod(int num) {

if(num==1)

throw new ArithmeticException("Exception Message1");

else if(num==2)

throw new NullPointerException("Exception Message2");

else

{

System.*out*.println(num);

}

}

}

class Demo{

public static void main(String args[])

{

try{

Example obj=new Example();

obj.mymethod(1);

}catch(Exception ex){

System.*out*.println(ex);

}

}

}

**Ex-2: in the previous example we are not using beacue of unchecked exception**

**But in this we have to mention the because of checked exception**

import java.io.\*;

class Example

{

void mymethod(int num)throws IOException, ClassNotFoundException

{

if(num==1)

throw new IOException("Exception Message1");

else if(num==2)

throw new ClassNotFoundException("Exception Message2");

else

{

System.out.println(num);

}

}

}

class Demo{

public static void main(String args[])

{

try{

Example obj=new Example();

obj.mymethod(1);

}catch(Exception ex){

System.out.println(ex);

}

}

}

**Ex-2:**

import java.util.\*;

class InvalidAgeException extends Exception

{

InvalidAgeException(String s)

{

super(s);

}

}

class Excep

{

static void validate(int age)throws InvalidAgeException

{

if(age<18)

throw new InvalidAgeException("not valid");

else

System.out.println("welcome to vote");

}

public static void main(String[] args) {

try

{

Scanner s1 = new Scanner(System.in);

System.out.println("enter age");

int age = s1.nextInt();

validate(age);

}

catch(Exception e)

{

System.out.println("exception occured :"+e);

}

}

}

**Q-10: What is the diff b/w throw and throws**

**A:** **1-**throw keyword is used to throw an exception explicitly.

Throws clause in used to declare propagate an exception

**2-**The keyword throw is used inside method body to invoke an exception

throws clause is used in method declaration

**3-**By using Throw keyword in java you cannot throw more than one exception

but using throws you can declare multiple exceptions

**4-**Checked exception cannot be propagated using throw only.

Checked exception can be propagated with throws.

**Q-11: What is OutOfMemoryError.**

**public** **class** Demo {

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

Long maxMemory = Runtime.*getRuntime*().maxMemory();

System.*out*.println(maxMemory);

**int**[] matrix = **new** **int**[(**int**) (maxMemory+1)];

**for**(**int** i = 0; i < matrix.length; ++i)

matrix[i] = i+1;

}

}

Max size is 1039990784 but when execute this program it will show 259522560 and shows java.lang.OutOfMemoryError: Java heap space

**A:** So we can resolve errorBy customize your heap size.

**Q-12: How to resolve OutOfMemoryError.**

**A:** we can customize heap size in Eclipse

**1. Solution – VM arguments**

On Eclipse menu, clicks Run -> Run Configurations.., select the Java application you want to run, clicks on theArguments tab, update the VM arguments with the following options

-Xms<size> - Set initial Java heap size

-Xmx<size> - Set maximum Java heap size

**For example, -Xms512M -Xmx1024M**

**Q-13:What is Stack Overflow Error**

**public** **class** Demo {

**public** **static** **void** recursivePrint(**int** num)

{

System.*out*.println("Number: " + num);

**if**(num == 0)

**return**;

**else**

*recursivePrint*(++num);

}

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

{

Demo.*recursivePrint*(1);

}

}

**A:** in this program it will showjava.lang.StackOverflowError

**Solution of the problem->**

Open the **Run Configuration** for your application (Run/Run Configurations..., then look for the applications entry in 'Java application').

The **arguments** tab has a text box **Vm arguments**, enter -Xss1m (or a bigger parameter for the maximum stack size). The default value is 512 kByte

**Q-14: Difference between ClassNotFoundException Vs NoClassDefFoundError**

**A:** **ClassNotFoundException** is a run time exception which is thrown when an application tries to load a class at run time using **Class.forName()**  or **loadClass()**  or **findSystemClass()** methods and the class with specified name are not found in the classpath. **For example:** you may have come across this exception when you try to connect to MySQL or Oracle databases and you have not updated the classpath with required JAR files. In most of time, this exception occurs when you try to run an application without updating

**NoClassDefFoundError** is an error which occurs when a particular class is present at compile time but it was missing at run time.

**Q-15: Can we write finally without try-catch.**

**A:** No

**Q-16: What happens when exception is thrown by main method?**

**A:** When exception is thrown by main() method, Java Runtime terminates the program and print the exception message and stack trace in system console.

**Q-17: if I do not want to execute finally block but still their finally block than what to do.**

**A:** use System.exit(0) in try block.

**Q-18: Is it possible to take try, catch inside try block?**

**A:** Yes, It is possible to take try, catch inside try block. That is nesting of try catch is possible.

**Q-19:Is it possible to take try, catch inside catch block?**

**A:**Yes, It is possible to take try, catch inside catch block.

**Q-20: Is it possible to write any statement between try-catch and finally?**

**A:**No, it is not possible to write any statement between try catch and finally. If we will try to write any statement between them then we will get compile time error.

**Q-21: Is syntax try-finally-catch is valid ?**

**A :** No, this syntax is not valid. It should be like try-catch-finally then only code will compile.

**Q-22: Is it possible to throw an Error?**

**A:** Yes, It is possible to throw any Throwable type including Error.

**Q-23: Is it possible to throw any java object?**

**A:** No, we can use throw keyword only for throwable objects otherwise we will get compile time error saying incompatible type.

**Q-24: Is it possible to use throws keyword for any java class?**

**A :** No, we can use throws keyword only for Throwable classes. Otherwise we will get compile time error saying Incompatible types.