Dec-8th-2019 N Hemanth

1. Why do we need to handle the exception ? and how do we do exception with example ?

→ we need to handle the exception so we can maintain the normal flow, without terminating it . in java we have exception handling to maintain the runtime errors to maintain the normal flow of the application can be maintained.

Example : FileNotFoundException.

Public class FileOpen{

Public static void main(String[] main){

try{

File file = new File(“D://file.txt);

}catch(FileNotFoundException e){

System.out.println(e);

}

}

}

Example : we thrown the file not found exception so that when program search for the files it throws an exception but flow of program will be normal.

2.what is a exception ? explain about kinds of exceptions ?

→ exception is an event that disrupts the normal flow of program, it is an object thrown at the time of runtime.

Kinds of exceptions :- total three types of exceptions

A.checked

B.unchecked

C.error

a.) checked exception :- checked exceptions are checked at a compile time.the classes which can inherit directly throwable class except runtime exception. Ex . IO Exception,SQLException etc.

b.)unchecked exceptions :- the class which inherit runtimeexception are known as unchecked exception ex . Arithmeticexception.NullPointerException etc

c.) Error : Error is cannot solve, beyond the line of it.

3.Differentiate between error and exception ?

→ an exception is a event which can interrupts the normal flow of a program while error occurs with in a method of class which directly occur at a runtime. Bot unchecked exceptions and errors are at a runtime..

4.Explain about Exception Hierarchy ?

→ java.lang.throwable is the root class of java exception hierarchy which inherited by two subclasses exception and error.

Exception class : in Exception class there are 4 types IOException,SQLException,ClassNotFoundException and RuntimeException.

From RuntimeException there are again four exception. a.Arithmetic exception b.NullPointerException c.NumberFormatException d.indexoutofbound exception from Indexoutofboundexception two exception Arrayindexoutofboundexception and second one is Stringoutofbound exception.

Error class : StackOverFlowerror,Virtualmachineerror and outofmemoryerror.

5.Differentiate between throw and throws with example ?

Throw :

→ throw is used to explicitly used to throw exception.

→ checked exception cannot be propagated using throw only

→ throw is followed instance only

→ throw is used within method only

→ you cannot throw multiple exceptions.

throws :

→ throws keyword is used to declare exception

→ checked exception can be propagated with throws

→ throws is followed by class.

→ throws is used with method signature.

→ you can declare multiple exceptions e.g. public void method() throws exception SQLException.

6.what are the different types of Exceptions in java ? Explain in detail with examples ?

→ IOException :- Input and Output Exception where the exception thrown when the input or output fails example with file systems

→ SQLException : it's database exception when java working on sql queries it throws an sql exception for databases and queries .

→ classNotFoundException :- when you try to load a classes which are not found from classpath where usually it try to load classes at a runtime using class.forname and load.classes().Noclassdeffounderror which occur when the class is compiled at compiler but not found in runtime.

Runtime Exception : runtime exceptions or unchecked exception which occur at runtime

→ Arithmetic exception : exceptions and errors which are related to the mathematical functions are thrown by arithmetic exception . it belongs to unchecked exception or runtime exception.

Public class MathematicalException{

Public static void main(String[] main){

Int num = 6/0;

}

}.

→ NullPointerException : when a pointing to a variable or accessing when it does not have anything or having null it will throw null pointer exception.

Public class NullException{

Public static void main(String[] main){

method(null)

}

Public void method(String s){

System.out.println(s.toLowercase());

}

}.

→ NumberFormatexception : for mismatch input or any other non synchronized data it throws numberformatexception.

Public class NumberFormat{

Public static void main(String[] main){

Int k = “hemanth’;

}

}.

→ IndexOutOfBoundException : ArrayIndexoutofboundexception and stringoutofboundexception are both come under indexoutofboundexception where when we trying to access the value which out of array length it throws indexoutofbound exception its same as for string to

Example :

Public class OutOfBoundIndex{

Public static void main(String[] main){

Int[] arr = new Int[5];

Int k = arr[6];

}

}.

7.a)

import java.util.Scanner;

class Division {

public static void main(String[] args) {

int a, b, result;

Scanner input = new Scanner(System.in);

System.out.println("Input two integers");

a = input.nextInt();

b = input.nextInt();

result = a / b;

System.out.println("Result = " + result);

}

}

Result for the above program is divisible of two numbers input is int so no decimal formal will

Be come into it .

7.b)

class Division {

public static void main(String[] args) {

int a, b, result;

Scanner input = new Scanner(System.in);

System.out.println("Input two integers");

a = input.nextInt();

b = input.nextInt();

try {

result = a / b;

System.out.println("Result = " + result);

}

catch (ArithmeticException e) {

System.out.println("Exception caught: Division by zero.");

}

}

}

Output :

It will throw arithmetic exception if incase its occur it sees the low of program will occur normal/

7.c)

class Exceptions {

public static void main(String[] args) {

String languages[] = { "C", "C++", "Java", "Perl", "Python" };

try {

for (int c = 1; c <= 5; c++) {

System.out.println(languages[c]);

}

}

catch (Exception e) {

System.out.println(e);

}

}

}

The above program will catch IndexOutofBound exception

8.What is Serialization and Deserialization in java ? explain with examples ?

→ Serialization :- Serialization in java is a mechanism of writing a object in to a bytestream

example :

Import java.io.\*;

public class Serialization\_Example implements Serializable{

Int id;

String name;

public Serialization\_Example(int id,String name){

This.id = id;

This.name = name;

}

public static void main(String[] main){

Serialization\_Example se = new Serialization\_Example(89,”hemanth”);

FileOutputStream fout = new FileOutputStream(/home/hemanth/Documents/hemanth\_file/text.txt”);

ObjectOutputStream oos = new ObjectOutputStream(fout);

oos.writeObject(se);

oos.flush();

oos.close();

}

}

Deserialization : it’s a reconstructing the object from serialized data, it is a reverse option for serialization.

Example :

public class des{

public static void main(String[] main){

ObjectInputStream ois = new ObjectInputStream(new FileInputStream(“/home/text.txt”));

Serialization\_Example se = new (Serialization\_Example) ois.readObject();

System.out.println(se.id+”,”+se.name);

}

}

9.What is Byte Stream ?

→ stream is sequence of data, in java byte stream a stream is composed with bytes of data is called byte stream.

10.what is inputstream and outputstream ?

→ Inputstream : java application uses inputstream to read data from source like file,array, data structure etc. inputstream is a abstract class it's a superclass for all classes representing the inputstream of bytes

→ outputstream : java application uses output stream to send data to destination from source like file,array,data structure,socket etc.outputstream is an abstract class, its a superclass for all classes representing the outputstream of bytes.

11.mention about the methods used in fileinputstream and fileoutputstream ?

→ java Fileinputstream class obtain bytes from file , it is used for reading byte oriented data such as images,audio,video etc. we can also read character stream data for character stream data if we prefer filereader class.

Methods used in FileInputstream :

Int available() : it is used to return no of estimated number of bytes that can read from input stream.

Int read() : it is used to read byte of data from input stream

Int read(byte[] b) : it is used to read b.length of data from input stream.

Int read(byte[] b. Int off, int len) : it is used to read length of data from input stream

Long skip(long x) : it is used to skip and discard over x bytes of data from input stream.

Filechannel getChannel() : it is used to return unique file channel object from associated with input stream

File Descriptor getFD() : it is used to return file descriptor object object.

Void close() : it used to close the stream

protected void finalize() : it is used ti ensure that when close method is called there is no more referce to the file inputsrream.

FileOutputstream : using file output stream we can write primitive data , byte orinted data and character oriented data for charcater oriented we can use filewriter.

Methods in fileoutputstream :

protected void finalize() : it is used to clean up the connection with fileoutputstream

Int write(int x) : it used to write the data to file

Int write(byte[] b) : it is used to write b.length of data from input stream.

Int write(byte[] b. Int off, int len) : it is used to write length of data from input stream

Filechannel getChannel() : it is used to return unique file channel object from associated with input stream

File Descriptor getFD() : it is used to return file descriptor object object.

Void close() : it used to close the stream

12.12. Write a program for following

a. create a directory

b. In that directory add a file called text.txt

c. add content "This is a demo text file" in text.txt

d. Read the text.txt file

/\*\*

\*

\*/

package files;

import java.io.BufferedOutputStream;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

/\*\*

\* @author hemanth

\*

\*/

public class Create\_Directory\_File {

/\*\*

\* @param args

\*/

public static void main(String[] args)throws IOException {

// TODO Auto-generated method stub

File file = new File("/home/hemanth/Documents/hemanth\_file");

if (!file.exists()) {

file.mkdir();

System.out.println("Dircetory created sucessfully");

}else {

System.out.println("directory exist");

}

Path path = Paths.get("/home/hemanth/Documents/hemanth\_file/text.txt");

if (!Files.exists(path)) {

Files.createFile(path);

if (Files.exists(path)) {

String line = "This is a demo file";

FileOutputStream out = new FileOutputStream("/home/hemanth/Documents/hemanth\_file/text.txt");

BufferedOutputStream bout = new BufferedOutputStream(out);

byte[] arr = line.getBytes();

bout.write(arr);

bout.flush();

bout.close();

out.close();

System.out.println("data written to file");

}

} else {

System.out.println("file exist");

}

}

}

13. Program to check if a file or directory physically exist or not ?

/\*\*

\*

\*/

package files;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

/\*\*

\* @author hemanth

\*

\*/

public class Check\_File\_Directory\_ExistorNot {

/\*\*

\* @param args

\* @throws IOException

\*/

public static void main(String[] args) throws IOException {

// TODO Auto-generated method stub

Path path = Paths.get("/home/hemanth/Documents/hemanth\_file/text.txt");

if (!Files.exists(path)) {

Files.createDirectories(path);

Files.createFile(path);

}else {

System.out.println("FIles and Directory exist");

}

}

}

14 Consider a file contains a paragraph of data. Write a program to read the file line by line

/\*\*

\*

\*/

package files;

import java.io.IOException;

import java.io.InputStream;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

/\*\*

\* @author hemanth

\*

\*/

public class Read\_Paragraph\_LinebyLine {

/\*\*

\* @param args

\*/

public static void main(String[] args)throws IOException {

// TODO Auto-generated method stub

Path path = Paths.get("/home/hemanth/Documents/hemanth\_file/paragraph.txt");

if (Files.exists(path)) {

InputStream fr = Files.newInputStream(path);

int strline;

while ((strline=fr.read())!=-1) {

System.out.print((char)strline);

}

System.out.println();

} else {

System.out.println("File not exist");

}

}

}

15.Mentioned about the methods in java.io.File class in java ?

→ File class is a abstract representation of path and directory pathname, pathname can be either abstract or relative.

Modifier\_type Method Description

|  |  |  |
| --- | --- | --- |
| static File | createTempFile(String prefix, String suffix) | It creates an empty file in the default temporary-file directory, using the given prefix and suffix to generate its name. |
| boolean | createNewFile() | It atomically creates a new, empty file named by this abstract pathname if and only if a file with this name does not yet exist. |
| boolean | canWrite() | It tests whether the application can modify the file denoted by this abstract pathname.String[] |
| boolean | canExecute() | It tests whether the application can execute the file denoted by this abstract pathname. |
| boolean | canRead() | It tests whether the application can read the file denoted by this abstract pathname. |
| boolean | isAbsolute() | It tests whether this abstract pathname is absolute. |
| boolean | isDirectory() | It tests whether the file denoted by this abstract pathname is a directory. |
| boolean | isFile() | It tests whether the file denoted by this abstract pathname is a normal file. |
| String | getName() | It returns the name of the file or directory denoted by this abstract pathname. |
| String | getParent() | It returns the pathname string of this abstract pathname's parent, or null if this pathname does not name a parent directory. |
| Path | toPath() | It returns a java.nio.file.Path object constructed from the this abstract path. |
| URI | toURI() | It constructs a file: URI that represents this abstract pathname. |
| File[] | listFiles() | It returns an [array](https://www.javatpoint.com/array-in-java) of abstract pathnames denoting the files in the directory denoted by this abstract pathname |
| long | getFreeSpace() | It returns the number of unallocated bytes in the partition named by this abstract path name. |
| String[] | list(FilenameFilter filter) | It returns an array of strings naming the files and directories in the directory denoted by this abstract pathname that satisfy the specified filter. |
| boolean | mkdir() | It creates the directory named by this abstract pathname |

16.what is garbage collection ?

→ garbage collection means reclaiming the runtime unused memory automatically, it is used to destroy the unused objects.it is automatically done by garbage collector as part of jvm . it makes java more memory efficient by removing unreferenced objects from heap memory.

Object can be unreferenced in three ways :-

→ by nulling the reference .

→ by assigning reference to another.

→ by anonymous object.

finalize() method : finalize () method revoked every time before object is garbage collected.the method can be used to perform cleanup processing.the method is defined in object class also.

gc() method : the gc() method is used to invoke the garbage collector to perform the clean up processing . gc() method found in system and runtime classes.

18. Write a program to perform garbage collection ?

/\*\*

\*

\*/

package test;

/\*\*

\* @author hemanth

\*

\*/

public class Grabage\_Collection {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

Grabage\_Collection gc = new Grabage\_Collection();

Grabage\_Collection cg = new Grabage\_Collection();

gc=cg;

System.out.println(gc+"..."+cg);

System.gc();

}

}

19.Explain about ways for requesting JVM to run Garbage Collector with example ?

→ whenever a jvm runs a garbage collector , then only object will be destroyed.

By System class :

System class contains a static method garbagecollector ,called System.gc().

By Runtime class :

By using runtime object a java application can communicate with jvm. Runtime class is a singleton class we can’t create a runtime object by using constructor.

We can create a runtime object by using factory method getRunTime()

i f we can get runtime object then we can apply freememory() return free memory to heap,totalmemory() return total memory of heap(heap size),gc() for requesting jvm to invoke garbage collector.

System.gc(),Runtime.getRunTime.gc() are two proper way of methods to invoke or request garbage collector.

20. What is enumeration? Explain with an example ?

→ enum is a datatype which can store fixed set of constants,enum improves types safety

Can be easily used in switch and traversed , it can have fields,constructors and methods, enum can implement interface but cannot extend any classes because inbuilt extended Enum class.

Example.

/\*\*

\*

\*/

package test;

/\*\*

\* @author hemanth

\*

\*/

public class Enum\_num {

enum colors{

SAFFRON{

public String getinside() {

return "DivineColor";

}

},

RED{

public String getString() {

return "Dangerous";

}

};

}

enum season {

WINTER(10),SUMMER(15),SPRING(20),RAINY(25);

private int value;

private season(int value) {

// TODO Auto-generated constructor stub

this.value = value;

}

public void nothing() {

value = 10;

System.out.println("public method called"+value);

}

}

}

21.21. Consider a variable Directions of enum type, which is a collection of four constants EAST, WEST, NORTH and SOUTH.

Create a class EnumDemo and make use of enum variable. (example if you get the enum value is EAST. you should display "you are at EAST direction")

/\*\*

\*

\*/

package test;

/\*\*

\* @author hemanth

\*

\*/

public enum Directions {

NORTH,SOUTH,EAST,WEST;

}

/\*\*

\*

\*/

package test;

/\*\*

\* @author hemanth

\*

\*/

public class Enum\_Demo {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

Directions direction = Directions.EAST;

if (direction == Directions.EAST) {

System.out.println("your at east direction");

}else if(direction == Directions.WEST) {

System.out.println("your at west direction");

}else if(direction == Directions.SOUTH) {

System.out.println("your at south direction");

}else if(direction == Directions.NORTH) {

System.out.println("your at north direction");

}else {

System.out.println("unknown direction:");

}

}

}

22.Explain about Autoboxing and Unboxing with an example ?

Autoboxing : automatic conversion of primitive data types into its equivalent wrapper class is called autoboxing.

Example:

/\*\*

\*

\*/

package test;

/\*\*

\* @author hemanth

\*

\*/

public class Autoboxing {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

int hemanth = 89;

Integer g = new Integer(hemanth);

System.out.println("the above instruction shows the autoboxing");

}

}

Unboxing : automatic conversion of wrapper into primitive type is called unboxing.

Example :

/\*\*

\*

\*/

package test;

/\*\*

\* @author hemanth

\*

\*/

public class Unboxing {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

Character ch = new Character('h');

char ohoo = ch;

}

}

23. What are annotations in java ? Explain with an example ?

→ java annotation is a tag that represents the metadata to indicate some additional information which can be used additionally for jvm and compiler. It make an alternative for XML and java marker interfaces.

There are two types of annotations

1.Built-in

2.custom made

Built-in annotation :

1. @Override
2. @Suprresedwarnings
3. @Deprecated
4. @Target
5. @Retention
6. @Inherited
7. @Documented.

Example :

Default method program :

@Override

public String toString() {

// TODO Auto-generated method stub

return name + "," + rollnumber + "," + marks;

}

Custommade program :

/\*\*

\*

\*/

package test;

import java.lang.annotation.\*;

import java.lang.reflect.\*;

/\*\*

\* @author hemanth

\*

\*/

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.METHOD)

@interface hems{

int value();

}

class Test{

@hems(value = 89)

public void exec() {

System.out.println("hello");

}

}

public class Built\_Inbuilt\_Annotations {

/\*\*

\* @param args

\* @throws SecurityException

\* @throws NoSuchMethodException

\*/

public static void main(String[] args) throws NoSuchMethodException, SecurityException {

// TODO Auto-generated method stub

Test tst = new Test();

Method m = tst.getClass().getMethod("exec");

hems hs = m.getAnnotation(hems.class);

System.out.println(hs.value());

}

}