Exception Handling

Unit 3

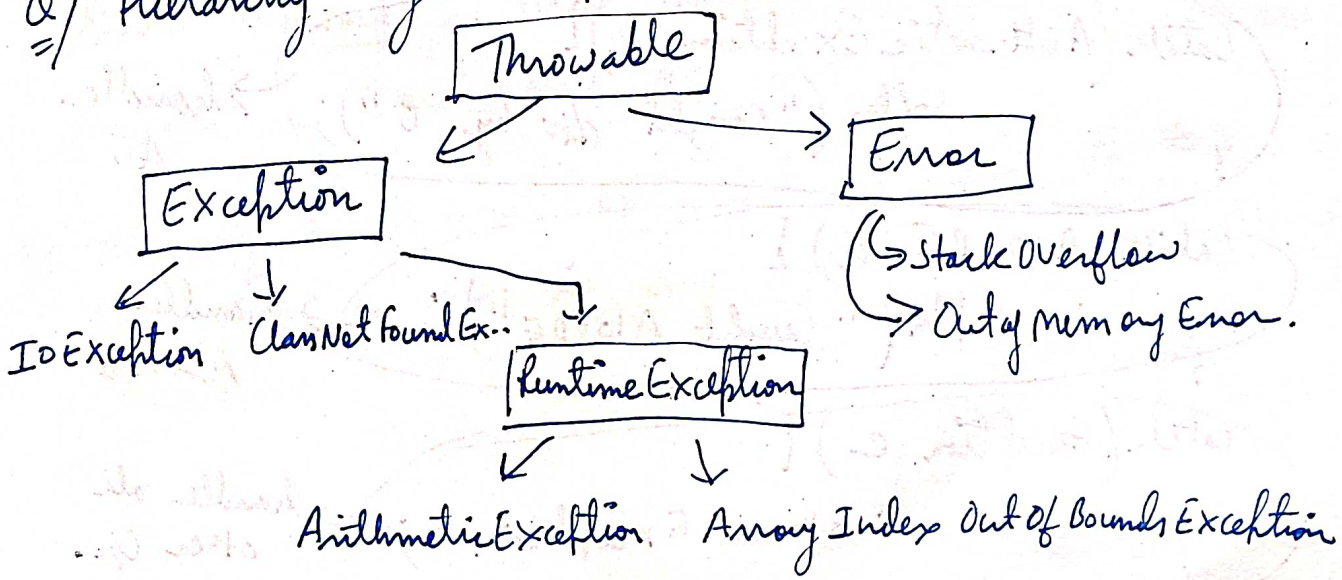
try, catch, throw, finally, throws, checked, unchecked Exceptions

Q) What r Exceptions?

A) Exceptions are runtime errors.
- They r undesirable as they result in abnormal program termination
- Exceptions are unpredictable.

Eg:- Div by 0
Sq. root of -ve No.
File/link not found.
m/m not available

Q) Hierarchy of Exception classes?



Q) Explain how Ex. r handled using try & catch block? WAP to demonstrate.

A) try :- part of code where Ex. can occur is placed inside a try block.

Catch :- The Ex is handled inside the catch block. There may be multiple catch blocks to handle different types of Ex.

```
C.D{  psvm(){
    int ar[] = {10,20}; //create an arr.
    int a = sc.nextInt(); // take user input
    try{                                          → try block
        Sopln(5/a); // div by 0 ex. can occur.
        ar[1]=200;
        ar[2]=300; // Array IndexOutofBounds Excep can occur.
    }
    Catch(ArithmeticException e){
            Sopln("Caught div by 0");             → handle AE
    }
    Catch(AIOOBE e){
            Sopln("Caught AIOOBE");               → handle AIOOBE
    }
    Catch(Exception e){
            "other Exceptions";                   → handle all other ex.
    }
```

Note 1) The last catch block is optional. It handles all other types of Exceptions
- All Ex. can be caught by the Exception class.

2) ** Catch block with "Exception class" must be placed at last. Order is imp.
- If catch block with "Exception class" is placed at first, then it will catch all Ex., other catch blocks will never run.
- So compile time error will occur, since unreachable code is an error in java.

---

throwing Ex. using throw keyword

- / by 0 (div by 0) & AIOOBE (an index out of Bounds Ex.)
  ↳ auto^{y} thrown by java compiler.

- throw keyword is used when u want to manually throw an Ex. based on some external condition.

eg div by 1 is not ex., but u want to throw an Ex, when a no. is div by 1, then use throw keyword.

syntax :-

```
ij( ){
    throw new <type of Ex.>
}
```

eg WAP to throw AE (Arithmetic Exception) when a No. is div by 1

```
CD { PSVM(){
    int a = sc.nextInt();
    try {
        Sopln(5/a); //if a is 0, then AE is auto thrown.
    }
    if(a==1){
        throw new ArithmeticException("msg");
    //if a is 1, then AE is manually thrown.
    // msg can be anything;
    }
    } //try block
    Catch(ArithmeticException e){
        "Caught";
    }
}
}
```

Exceptions & functions

- There are two ways to use ex. handling w/ fun.

1) complete try catch block can be inside the fun.

eg
```
fun ( ) {
    try {
        =
    }
    catch ( ) {
        =
    }
}
main ( ) {
    fun ( );
}
```
// try catch inside fun.

// fun is called inside main.

---

2) the fun call can be put inside try catch block.

```
fun ( ) {
    =
}
```
→ No try, catch here.

```
main ( ) {
    try {
        fun ( );
    }
    catch ( ) {
        =
    }
}
```
→ fun call is put inside the try catch block.

finally block

- any code that needs to be mandatorily executed after try & catch shud be placed inside a "finally" block.
- usually the pgm continues after try catch block However there are some cases in wh. the pgm may not continue.

eg: uncaught exception.

- If pgm contains any uncaught ex., then abnormal pgm termination will occur & pgm will not continue after try & catch.
- the finally will run just B4 abnormal pgm termination.

```
CD { PSVM() {
    int ar[] = {10, 20};
    try{
        sopln(5/a); // AE can occur.
        ar[1]=100;
        ar[2]=200; // AIOOBE can occur.
// AIOOBE is not handled, so abnormal pgm termination will occur. But finally block will run b4 termination.
}
```

```
catch (AE e){
        =
        =
 }
 finally {
    // runs even after abnormal pgm termination //
 }
 }
}
```

---

eg2. Suppose the fun returns inside try block.
So pgm will not get a chance to continue after try & catch.

• In this case finally block is run by the pgm fun returns.

```
class demo{
    public static void fun(int x){
        try{
            System.out.println(5/x);
            return; // fun returns here. But finally
// block is run by returning
        }
        catch (AE e){
            "caught";
        }
        finally {
            "this will run";
        }
        "after try catch - this part will not run";
    }
```

```
public static void main(String args[]){
        fun(10);
        fun(0);   } // call fun inside main
    }
}
```

Nested try catch & rethrowing an Ex.

Q) What is rethrowing ex.? Why is it done?

A) ex. can be nested.
try catch can be inside other try catch block.

rethrowing

- rethrowing is done w/ nested try catch.
- inner catch will handle an ex. & rethrow it.
- the rethrown ex. will be caught by outer catch block.

Purpose:-

- Ex. r rethrown to handle same ex. in 2 diff manner.
- 1st inner catch block will handle thez ex., then outer catch block will handle in diff. manner.

- Checked and Unchecked Exceptions
- throws keyword.

There are 2 types of ex. in java

1) Checked Ex.
2) Unchecked Ex.

Unchecked Ex:

- If any fun contains unchecked Ex. – then it is not necessary to handle the Ex.
- So java doesn't checks whether an unchecked Ex. is handled or not.

eg ⇒ Arithmetic Exception
Array Index Out of Bound Exception
} . There r unchecked. . If a fun contains there, then it is not necessary to handle them.

eg ⇒
```
class demo {
    PS void fun(int x) {
        Sopln(5/x); // this fun contains AE here.
        ar[1] = 100;
        ar[2] = 200; // AIOOBE occurs here.
    }
    PSVM() {
        fun(10);
        fun(0);
    }
}
```
. There Ex. r not handled.
. This pgm compiles.

Checked Ex :-

- If any fun contains checked Ex., then it must be handled. Eg. of Checked Ex. are FileNotFoundEx., ClassNotFoundEx.
- There r 2 ways to handle it.

eg: ①
```
CD {
    PS void fun() {
        try {
        }
        FileInputStream fin = new FileInputStream("test.txt");
        //FileNotFoundEx. occurs here
        //In this eg. - this Ex. is handled inside the fun
        using try catch block.
        }
        Catch (FNFE e) {
            "caught FNFE";
        }
} //fun ends.
PS V M ( ) {  //main fun.
    fun(); // fun called here.
}
```

Ex. is handled inside fun.

This is 1 way to handle Checked Ex.

- The second way to handle checked Ex. is by using throws keyword.
- If the Checked Ex. is not handled inside the fun, then throws keyword must be used after the fun name & specify the Ex.

```
class demo{
    public static void fun()
        throws FileNotFoundException {
//use throw keyword   // Specify Checked Ex. here.
FileInputStream fin = new FileInputStream("test.txt");
// this throws FNFE.
```

→ the FNFE is not handled inside fun.

```
}
psvm(){
    try{
        fun();  → fun called
    }
    catch(FileNotFoundEx e){
        "caught";
    }
}
```

→ ex. is handled here, not inside fun.

Note :-

- If a fun contains UnChecked Ex., then it is not necessary to handle it.
- If a fun contains Checked Ex., then it is necessary to handle it.
- It can be handled in 2 ways :-
  1. use try catch block inside the fun.
  2. use throws keyword after fun name & specify the Ex.
     - the try catch block will not placed inside the fun
     - try catch " " be placed inside the main() ~~function~~.

```
fun(){
    try
    {
    }
    catch
    {
    }
}
main(){
    fun()
}
```

try catch inside fun

```
fun() throws ----{
}
main(){
    try{
        fun()
    }
    catch{
    }
}
```

throws keyword here.

try catch here.

Checked Exception

- Mandatory to handle these ex.
- throws keyword can be used
- eg ArithmeticEx.
  ArrayIndexOutOfBoundsEx.
- If CEx. r not handled then compile time error.

eg
```
CD {
    PSV fun() {
        a=sc.nextInt();
        soln(5/a);
    }
    PSVM() {
        fun();
    }
}
```
// The pgm will not compile bcoz Check. Ex. is thrown but it is not handled.

Unchecked Exception

- Not mandatory.
- Not required.
- eg FileNotFoundEx.
  ClassNotFoundEx.
- If UncheckedEx. r not handled, then still the pgm will compile.

eg
```
CD {
    PSV fun() {
        a=sc.nextInt();
        soln(5/a);
        // Unchecked Ex. is thrown - AE.
    }
    PSVM() {
        fun();
    }
}
```
- // The Unchecked Exc. is not handled, still the pgm will compile & run.

```
FileInputStream fin = new FileInputStream("m");
// FNFE may occur
// Checked Exc. is thrown
// It is not handled.
```

throw	throws
• Used to manually throw an exc.	• Used to specify that a fun contains CheckedExcaption

• used inside the fun body
fun(){
throw ---- ;
}
	• Used after fun name.
	fun() throws AE {
	≡
	}

• Only 1 ex. can be specified after fun name.	• throws can specify mul Excep.
• throw keyword is followed by an obj	• throws keyword is followed by classname.

User Defined Ex.

- User defined Ex. can be created by extending the Exceptions. class.
- this ex. can be thrown by using the throw keyword.

Pgm to demonstrate user defined Ex.

```
class myexception extends Exception {        → inbuilt class.
    public myexception (String s){
        super(s);
        //Call constructor of parent Ex. class
    }
}
-----------
CD { psvm() {
    try {
        throw new myexception ("msg");
    }       // user defined ex. is thrown.
    catch (myexception e){
        // user defined ex. is caught.
        Sopln ("caught");
    }
}
}
```