SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Information Technology

19CS204 OBJECT ORIENTED PROGRAMMING

I YEAR /II SEMESTER

Topic – Exception Handling

Exception Handling

- An exception is an abnormal condition that arises in a code sequence at run time.
- In other words, an exception is a runtime error.
- In computer languages that do not support exception handling, errors must be checked and handled manually—typically through the use of error codes, and so on.
- This approach is as cumbersome as it is troublesome.
- Java's exception handling avoids these problems and, in the process, brings run-time error management into the object-oriented world.

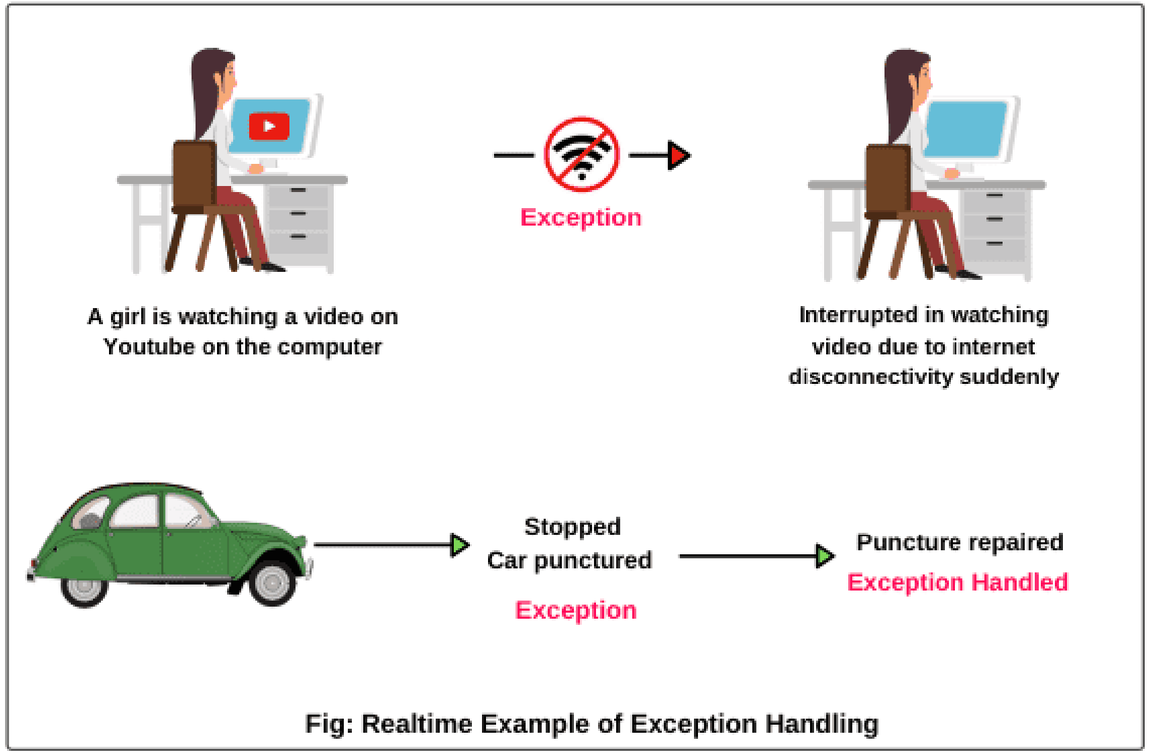


Fig: Realtime Example of Exception Handling

Exception Handling

- A Java exception is an object that describes an exceptional (that is, error) condition that has occurred in a piece of code.
- When an exceptional condition arises, an object representing that exception is created and thrown in the method that caused the error.
- That method may choose to handle the exception itself, or pass it on. Either way, at some point, the exception is caught and processed.
- Exceptions can be generated by the Java run-time system, or they can be manually generated by your code.
- Exceptions thrown by Java relate to fundamental errors that violate the rules of the Java language or the constraints of the Java execution environment.
- Manually generated exceptions are typically used to report some error condition to the caller of a method.

Exception Handling

Java exception handling is managed via five keywords:

- try, catch, throw, throws, and finally.

- Program statements that you want to monitor for exceptions are contained within a try block. If an exception occurs within the try block, it is thrown.

- Your code can catch this exception (using catch) and handle it in some rational manner.

- System-generated exceptions are automatically thrown by the Java runtime system.

- To manually throw an exception, use the keyword throw.

- Any exception that is thrown out of a method must be specified as such by a throws clause.

- Any code that absolutely must be executed after a try block completes is put in a finally block.

1. try
2. catch
3. throw
4. throws
5. finally

Exception Handling

This is the general form of an exception-handling block:

```
try {
// block of code to monitor for errors
}

catch (ExceptionType1 exOb) {
// exception handler for ExceptionType1
}

catch (ExceptionType2 exOb) {
// exception handler for ExceptionType2
}

// ...
finally {
// block of code to be executed after try block ends
}
```

Exception Handling - Types

Exception Types

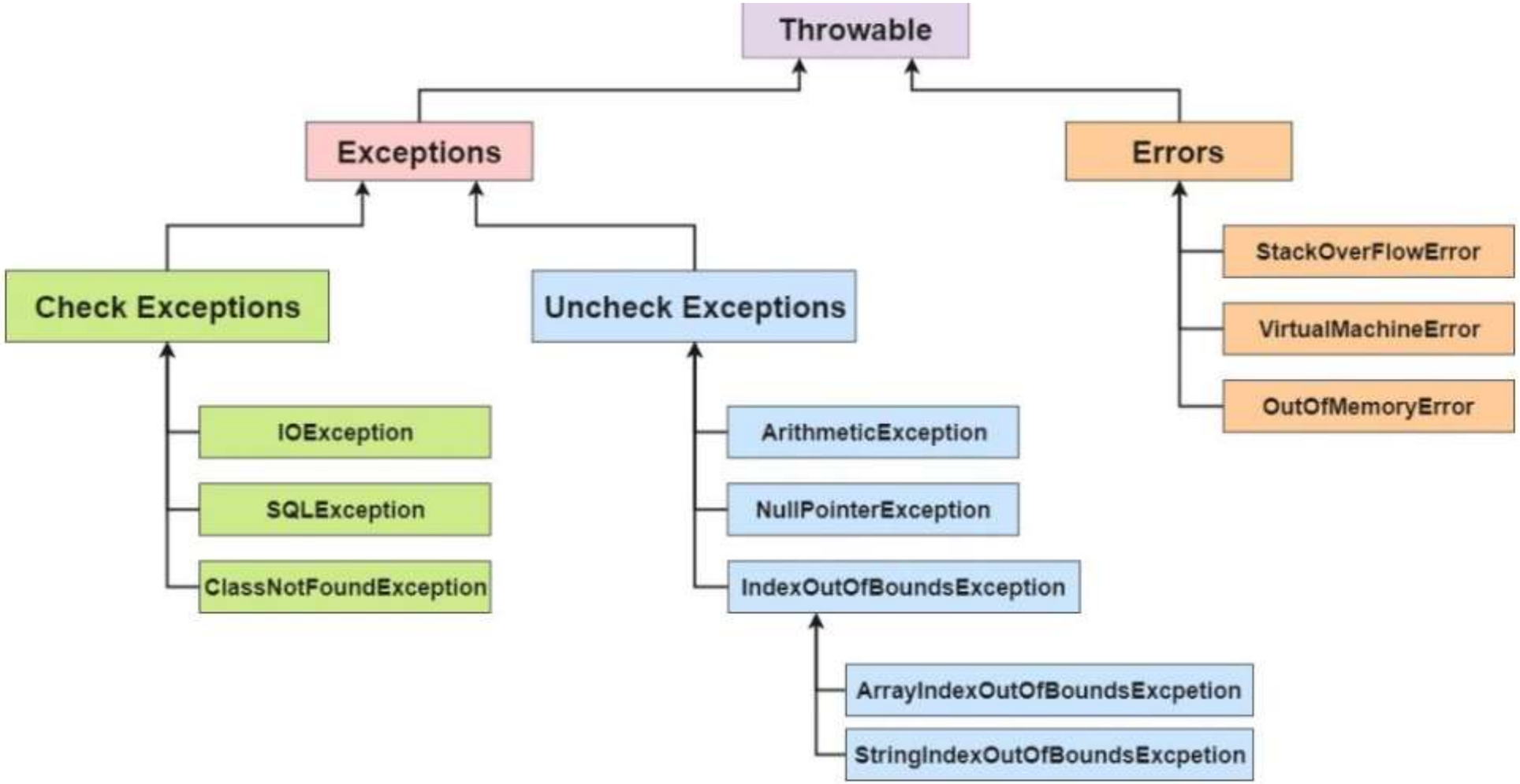
- All exception types are subclasses of the built-in class Throwable. Java.lang.Throwable;

- Throwable has two subclasses Exception, Error

Exception
- Exceptional conditions that user programs should catch.
- There is an important subclass of Exception, called RuntimeException.
- Exceptions of this type are automatically defined for the programs that you write and include things such as division by zero and invalid array indexing.

Error
- Exceptions that are not expected to be caught under normal circumstances by your program
- Exceptions of type Error are used by the Java run-time system to indicate errors having to do with the run-time environment, itself.
- Stack overflow is an example of such an error.

Exception Handling - Types



Exception Handling - Types

Unchecked Exceptions:

- They are not checked at compile-time but at run-time.
- For example: ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, exceptions under Error class, etc.

Checked Exceptions:

- They are checked at compile-time.
- For example, IOException, InterruptedException, etc.

Exception Handling

Uncaught Exception

```
public class Exc0 {
public static void main(String args[]) {
int d = 0;
int a = 42 / d;
}
}
```

- In this example, we haven't supplied any exception handlers of our own, so the exception is caught by the default handler provided by the Java run-time system.

- Any exception that is not caught by your program will ultimately be processed by the default handler

Here is the exception generated when this example is executed:

java.lang.ArithmeticException: / by zero
at Exc0.main(Exc0.java:4)

Exception Handling – Try, Catch

What if I want to handle exception by myself manually?

Using Try and Catch

Manually handle exception. It has two benefits

- First, it allows you to fix the error.
- Second, it prevents the program from automatically terminating.
- To guard against and handle a run-time error, simply enclose the code that you want to monitor inside a try block.
- Immediately following the try block, include a catch clause that specifies the exception type that you wish to catch.

Exception Handling – Try Catch

What if I want to handle exception by myself manually?

Using Try and Catch

```
public class Exc2 {
public static void main(String args[]) {
int d, a;
try { // monitor a block of code.
d = 0;
a = 42 / d;
System.out.println("This will not be printed.");
}
catch (ArithmeticException e) { // catch divide-by-zero error
System.out.println("Division by zero.");
}
System.out.println(“Hello I caught exception”);
}}
```

Notice that the call to println( ) inside the try block is never executed.

Once an exception is thrown, program control transfers out of the try block into the catch block.

Exception Handling – Multiple Catch

Multiple catch Clauses

- In some cases, more than one exception could be raised by a single piece of code.
- To handle this type of situation, you can specify two or more catch clauses, each catching a different type of exception.
- When an exception is thrown, each catch statement is inspected in order, and the first one whose type matches that of the exception is executed.
- After one catch statement executes, the others are bypassed, and execution continues after the try / catch block

Exception Handling – Multiple Catch

Multiple catch Clauses

```
public class MultipleCatches {
public static void main(String args[]) {
try {
String a=null;
System.out.println("a = " + a.length());
int b = 42 / 0;
int c[] = { 1,2,3 };
c[5] = 99;
}
catch(ArithmeticException e) {
System.out.println("Divide by 0: " + e);
}
catch (NullPointerException e){
    System.out.println("Null Pointer Exception " +e);
}
 catch(ArrayIndexOutOfBoundsException e) {
System.out.println("Array index oob: " + e);
}
catch(Exception e){
    System.out.println("General Exception " + e);
}
System.out.println("After try/catch blocks.");
}
}
```

THANK YOU