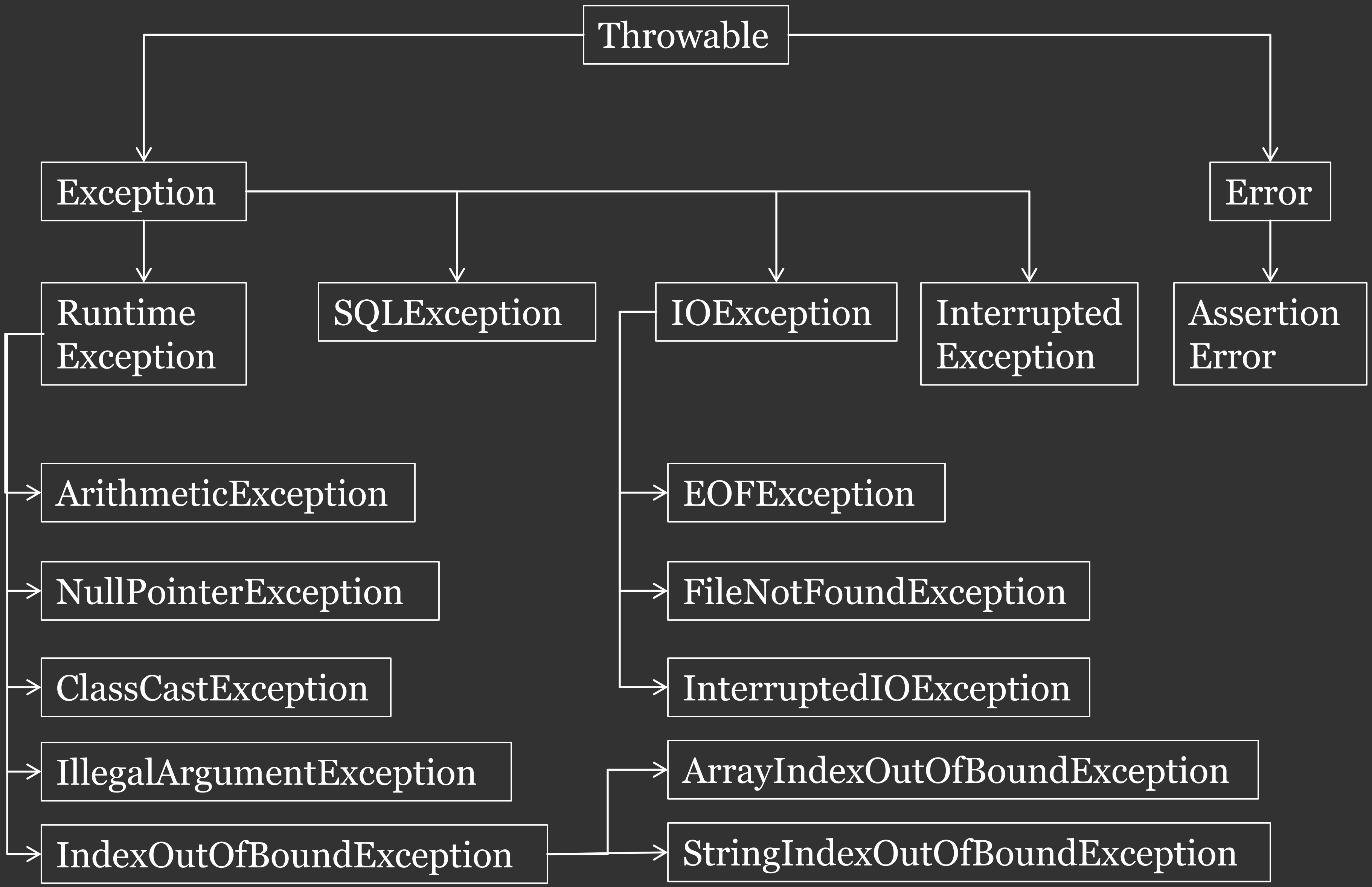
Exception Handling

Ram Sharma

Introduction

- In Java, Exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.
- Exception handling is to define alternative way to continue rest of the program normally whenever exception arises.
- If Exception is not handled properly, JVM terminates the program abruptly.
- If an exception arises, the method in which it is raised is responsible to create Exception object by including the following:
  - Name of Exception
  - Description of Exception
  - Location of Exception(stack trace)

Exception Hierarchy



Points to Ponder

- Exceptions, in many cases, caused by our program and hence they are recoverable.
- Error are mostly caused due to lack of system resources and they are non-recoverable.
- The Exceptions which are checked by compiler for smooth execution of the program at runtime are called '***Checked Exception***'.
- Whether an Exception is Checked or Un-checked, it is raised only at runtime.

Exception Handling

- We can maintain Risky Code within the try block and corresponding Handling code inside catch block.

```
try{
        //Risky Code
}catch(SomeException e){
        //Handling Code
}finally{
        //Resource closing
}
```

- Irrespective of whether an Exception arises or not, finally block is executed.
- Within try block, if anywhere an Exception is raised, the rest of try block won't be executed even though we have handled that Exception.

Methods for Exception Info

- Throwable class defines the following methods to get Exception Information.
- public void printStackTrace()

  This method prints Exception information in the following format:

      Name of Exception:Description

      Activation Records

- public String toString()

  It return name and description of the Exception.

throw and throws

throw

- We can create Exception object manually and handover that object to JVM explicitly by using throw keyword.

```
throw new ArithmeticException()
```

- After throw keyword, we must have an object of Exception or its child class.
- It is used when we have customized Exception.

throws

- In our program, if there is a chance of raising CheckedException, we can to handle it or we can delegate this responsibility to the caller method.

```
public void myMethod() throws InterruptedException{
        Thread.sleep(1000);
}
```

- In case of Unchecked Exceptions, it is not required to use throws keyword.