While folks are joining

- Get you laptops ready and login to [www.crio.do]
- Open [QCalc ME] and start your workspace.
- Open Terminal and type

```
cd ~/workspace
```

- Clone the repo in ~/workspace directory
  - `git clone git@gitlab.crio.do:bdt-sprint-codes/java-ii/java-ii-session-activities.git`
- Open session-7 folder.
- Wait for Java Language Server Setup to complete.
- [Setup Video for Reference]

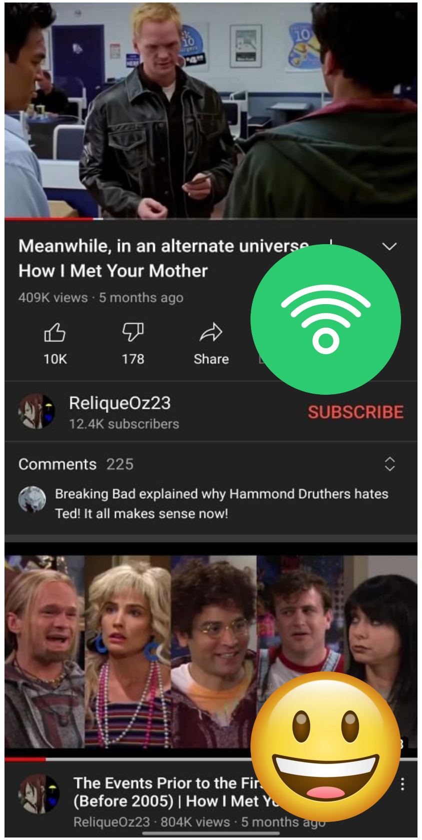
Crio Sprint: JAVA-112

Session 8 - Exception Handling

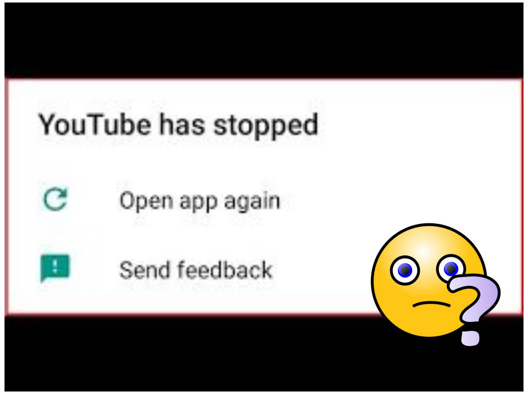
Today’s Session Agenda

- Exception Handling
- QCalc: Module 3 Introduction
- QCalc: Module 4 Introduction

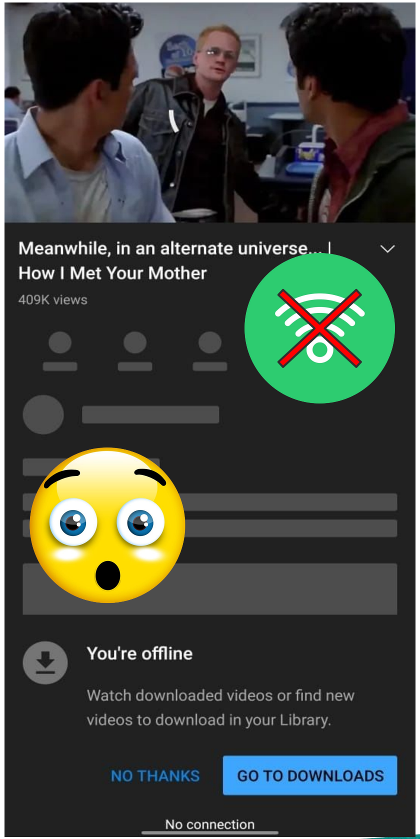
Video Playback


Meanwhile, in an alternate universe... | How I Met Your Mother
409K views · 5 months ago
10K
178
Share
ReliqueOz23
12.4K subscribers
SUBSCRIBE
Comments 225
Breaking Bad explained why Hammond Druthers hates Ted! It all makes sense now!
The Events Prior to the Fi
(Before 2005) | How I Met Yo
ReliqueOz23 · 804K views · 5 months ago

Network loss is an unexpected event.



What just happened here? Unexpected huh!
Is it the right thing to do?
Can we handle it better?

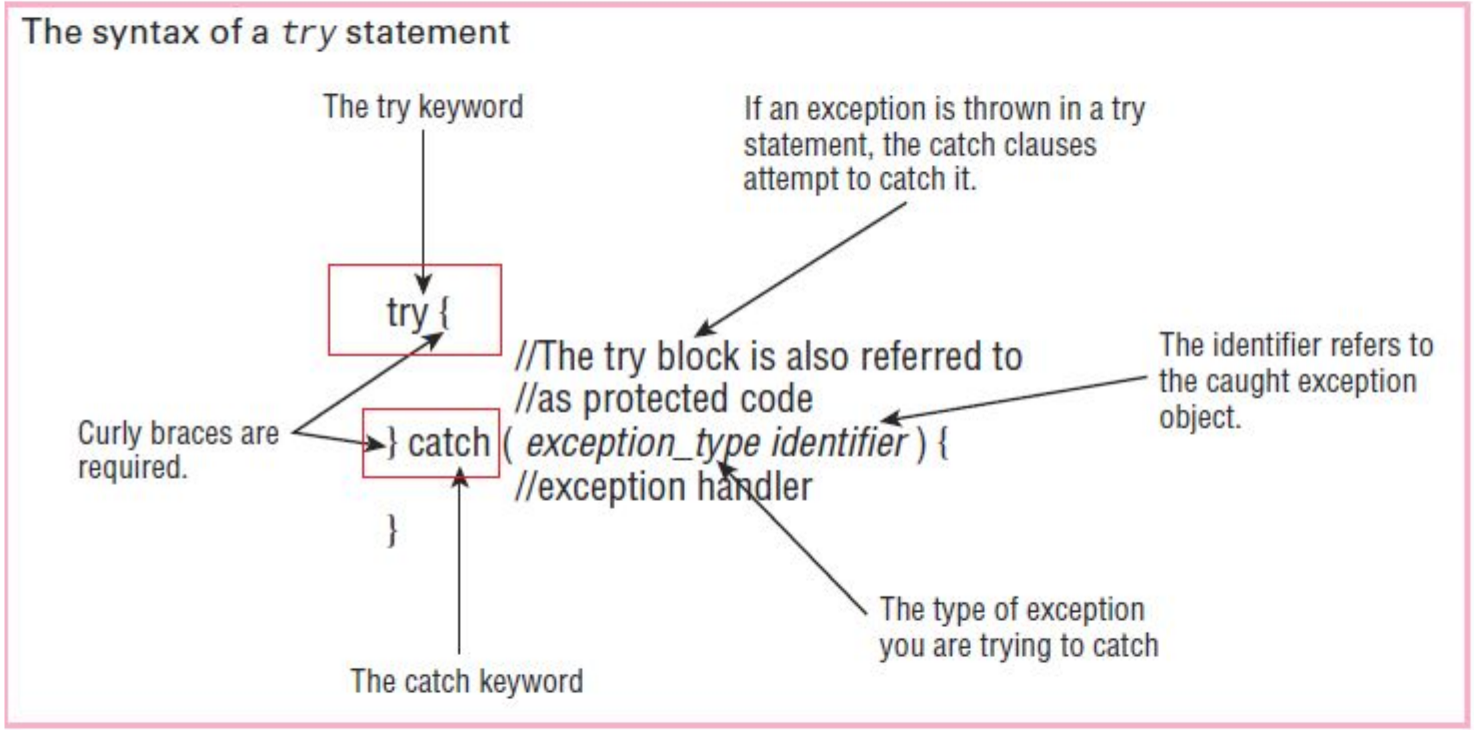

Meanwhile, in an alternate universe... | How I Met Your Mother
409K views
You're offline
Watch downloaded videos or find new videos to download in your Library.
NO THANKS
GO TO DOWNLOADS
No connection

What is an Exception and Exception Handling?

- During the **execution** of a program, things can, and **do go wrong.**
- The user may **enter inappropriate data**,and **program during runtime may throw an error** and terminate**.**
- **Exception handling** is the process designed to **handle such exceptions,** so that the program can **continue gracefully**.
- Some of the Exceptions you might face in real world
  - Invalid User Input
    - Assigning String to an Integer Variable
  - Device / Hardware Failure
    - Phone Camera stopped working unexpectedly
  - Network Loss
  - Out of Disk Memory
    - Phone Storage is full.
  - File Operations
    - Opening unavailable file (File Access Issue)

How can you stop the process from crashing?

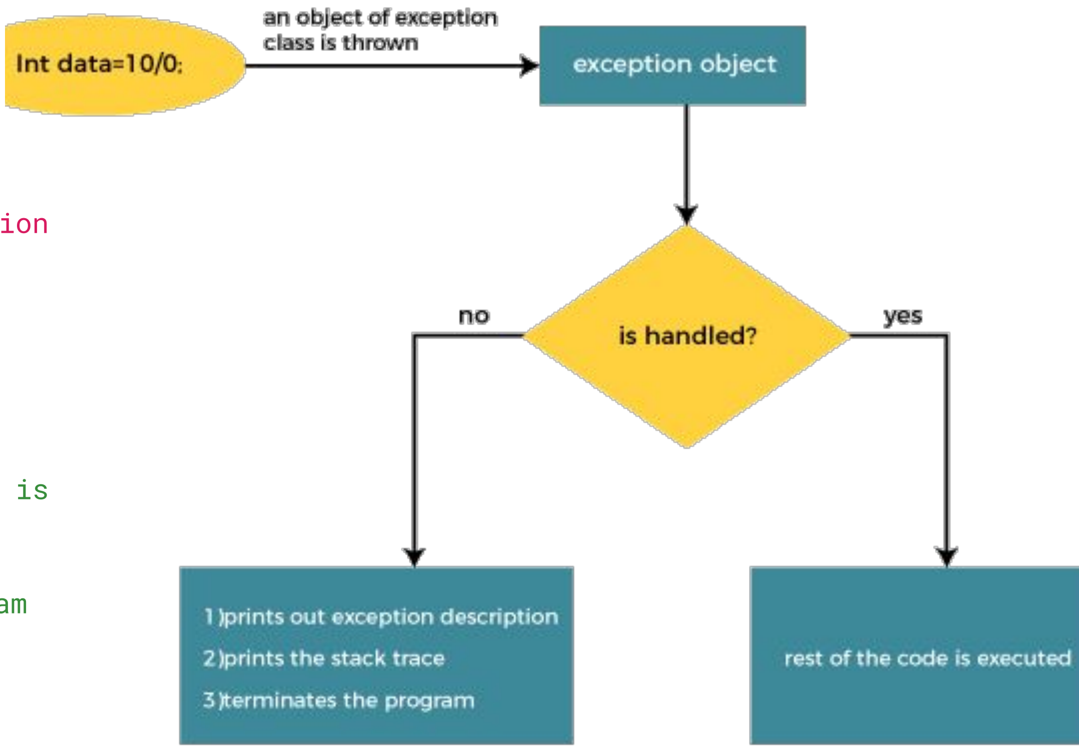
Using the **try catch** syntax


The syntax of a *try* statement
The try keyword
If an exception is thrown in a try statement, the catch clauses attempt to catch it.
try {
//The try block is also referred to
//as protected code
} catch (*exception_type identifier*) {
//exception handler
}
Curly braces are required.
The identifier refers to the caught exception object.
The type of exception you are trying to catch
The catch keyword

Try Catch Finally Block

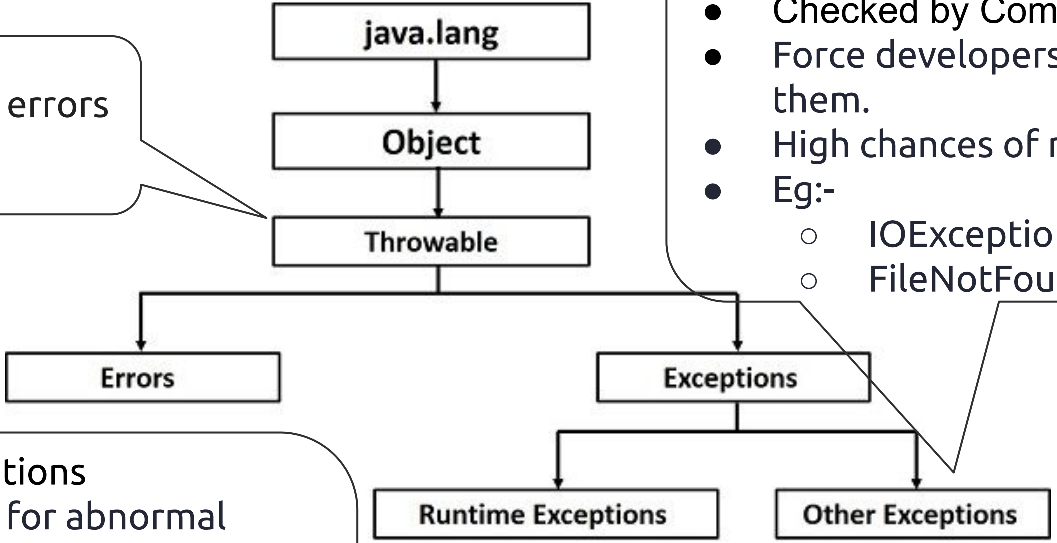
```
public class Main {

    public static void main(String[] args) {
        try{
            int data=10/0; //may throw exception
        }
        //handling the exception
        catch(ArithmeticException e){
            System.out.println(e);
            e.printStackTrace();
        }finally {
            System.out.println("finally block is
always executed");
        }
        System.out.println("Rest of the program
can continue after graceful handling");
    }
}
```


Int data=10/0;
an object of exception class is thrown
exception object
is handled?
no
yes
1)prints out exception description
2)prints the stack trace
3)terminates the program
rest of the code is executed

Can we handle every type of Exceptions using try-catch?
What are this different types of Exceptions?

Types of Exceptions



Superclass of all errors and exceptions

- Checked Exceptions
- Checked by Compiler at compile time
- Force developers to handle or rethrow them.
- High chances of recovery
- Eg:-
    - IOException
    - FileNotFoundException

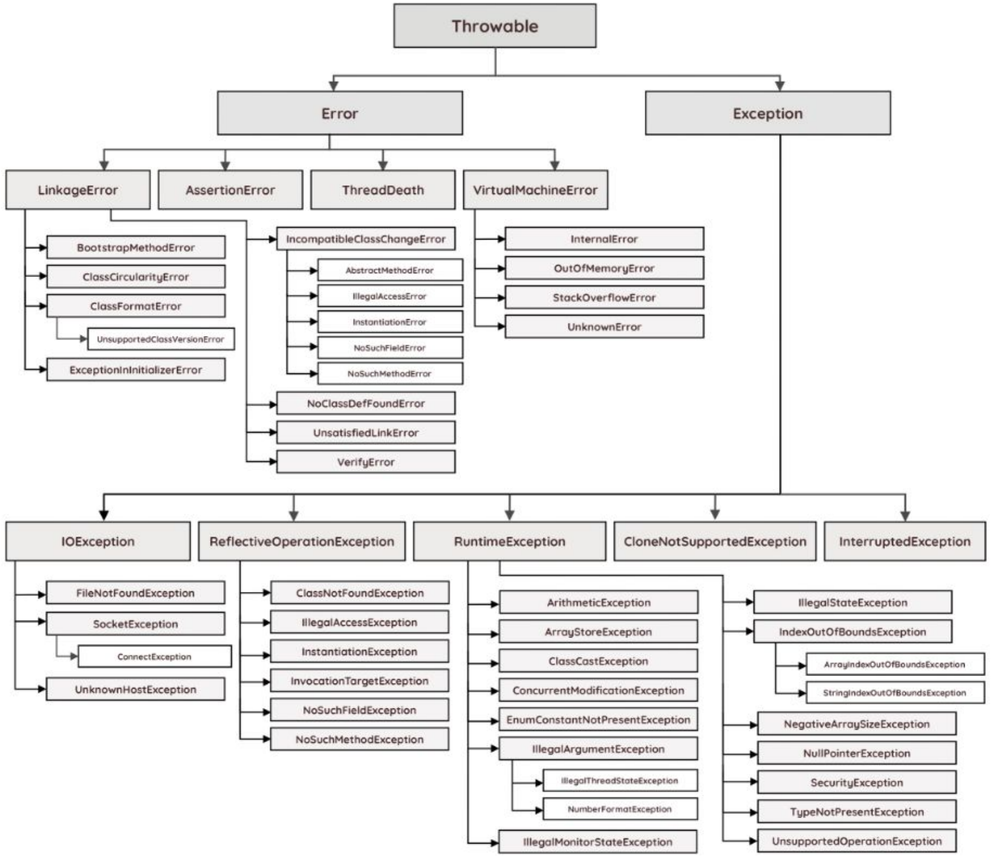
- Unchecked Exceptions
- Error case stands for abnormal situations.
- No chance for recovery. Program will most likely shutdown.
- Eg:-
- OutOfMemory
- Stackoverflow
- NoClassDefFoundError

- Unchecked Exceptions
- Not checked by compiler
- Will occur at runtime in buggy code
- Small chance of Recovery
- Eg:-
    - ArithmeticException
    - NullPointerException

Common Exception Methods (For Reference)

- **public String getMessage()**
  - Returns a detailed message about the exception that has occurred.
- **public Throwable getCause()**
  - Returns the cause of the exception as represented by a Throwable object.
- **public String toString()**
  - Returns the name of the class concatenated with the result of getMessage()
- **public void printStackTrace()**
  - Prints the result of toString() along with the stack trace to System.err, the error output stream

Java Exception Class Hierarchy (For Reference)



Would we ever need to throw an exception explicitly?

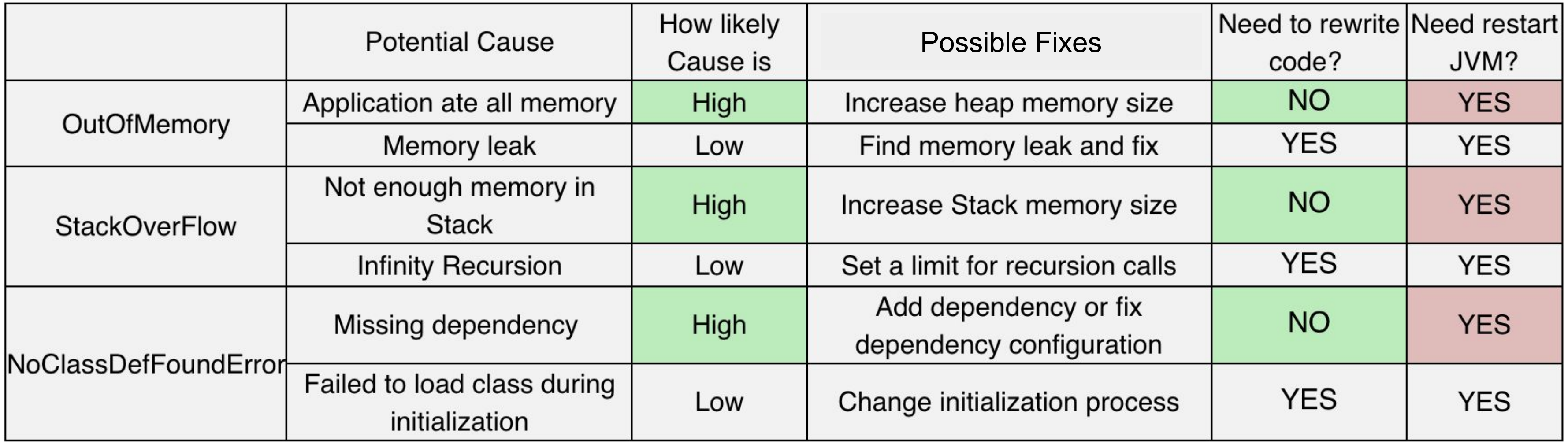
- Sometimes we might want to define our own set of conditions or rules and avoid abnormal behaviour.
  - For.e.g, Person having age less than 18 are not eligible for voting.
  - This abnormal behaviour can be handled using exceptions gracefully.
  - We can throw ArithmeticException with a message “Not eligible for Voting with Age < 18” specifying the correct error.
- We can define our own custom exceptions. We’ll visit this in further slides.

```
public class Main {
  public static void validate(int age) {
    if(age<18) {
      throw new ArithmeticException("Person is
not eligible to vote");
    }
    else {
      System.out.println("Person is eligible to
vote!!");
    }
  }
  public static void main(String args[]) {
      validate(13);
      System.out.println("rest of the code...");
   }
}
```

Activity #1 - Identify & Fix

- Change directory to activity1 on terminal
- Compile all the classes using javac command
  - javac -d . Main.java A.java B.java
- Run the program using java command
  - java com.crio.session7.activity1.Main
- What is the output?
- Delete A.class in generated com folder
- Run the program using java command and what is the output?
- Which category does this Exception belong to?
  - Errors
- What is the cause of this Exception?
- How can it be fixed?
  - All compiled classes must be present before run.

Investigating Error Exceptions (For Reference)



	Potential Cause	How likely Cause is	Possible Fixes	Need to rewrite code?	Need restart JVM?
OutOfMemory	Application ate all memory	High	Increase heap memory size	NO	YES
	Memory leak	Low	Find memory leak and fix	YES	YES
StackOverFlow	Not enough memory in Stack	High	Increase Stack memory size	NO	YES
	Infinity Recursion	Low	Set a limit for recursion calls	YES	YES
NoClassDefFoundError	Missing dependency	High	Add dependency or fix dependency configuration	NO	YES
	Failed to load class during initialization	Low	Change initialization process	YES	YES

Credits:- [Java Exceptions - DZone Java]

Activity #2 - Identify & Fix

- Run the program. What is the output?
- Which category does this Exception belong to?
  - Checked Exceptions
- What is the cause of this Exception?
- How can it be fixed?

Investigating Checked Exceptions (For Reference)

	Potential Cause	How likely Cause is	Possible Fixes	Need to rewrite code?	Need to restart?
FileNotFoundException	The file doesn’t exist	High	Create file	No	No
	Application call for the wrong path	Low	Fix wrong path generation	Yes	Yes
IOException	Access to resources is invalid	High	Make resources available again	No	No
ClassNotFoundException	The class wasn't added in dependency	High	Add missing dependency	No	Yes
	Implementation calls the wrong class	Medium	Change class call	Yes	Yes
SqlException	Schema doesn't match to query	High	Apply missing script to database	No	No
	Mistake in query	Low	Change the query	Yes	Yes
	Connection refused	High	Turn database on, change the port	No	No
InterruptedException	Dependent thread notified about interruption (lock released, another thread completed the operation)	High	There is no need to fix it; it's a way to notify about events in the dependent thread	No	No
	Another thread became broken and notified related using interrupt	Medium	Fix problem appeared in another thread (can be anything)	Yes	Yes
SocketException	Port is taken	High	Open/Release the port	No	No
	Server dropped connection	High	Check network connection or make	No	No

Activity #3 - Identify & Fix

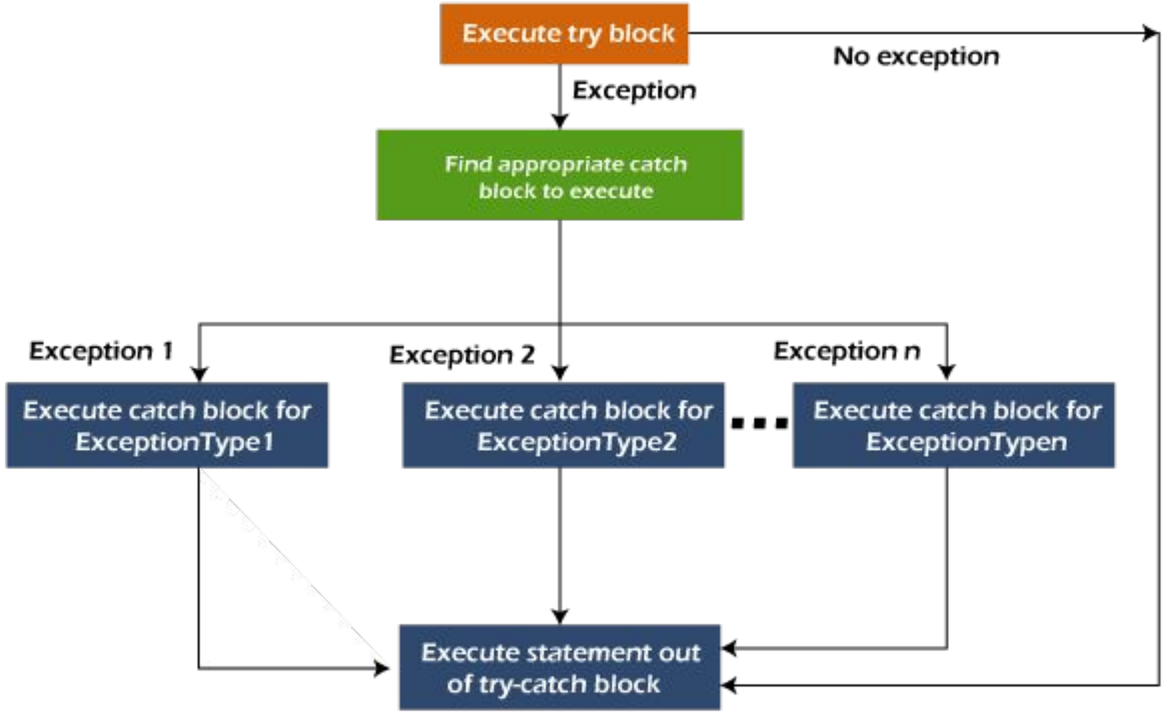
- Run the program. What is the output?
- Which category does this Exception belong to?
  - Runtime Exceptions
- What is the cause of this Exception?
- How can it be fixed?
  - Calling equals on literal rather than object (“Crio.Do”.equals(name))
- How NullPointerException can be avoided?
  - Java Tips and Best practices to avoid NullPointerException in Java Applications (javarevisited.blogspot.com) (Read after Session)

Investigating Runtime Exceptions (For Reference)

	Potential Cause	How likely Cause is	Possible Fixes	Need to rewrite code?	Need to restart?
NullPointerException	The expected non-nullable object was null	High	Add validation layer before calling	YES	YES
	Some resource wasn't available and returned null data instead	Medium	Add validation layer before calling	YES	YES
ConcurrentModificationException	The collection has been changed during iteration	High	Make collection iteration and modification separately	YES	YES
	The collection has been changed from another thread during iteration	High	Add synchronization for collection	YES	YES
IlliegalArgumentException	Passed parameter is invalid	High	Add validation before passing param	YES	YES
NumberFormatException	The passed parameter has the wrong format or wrong symbol	High	Add format or remove invisible symbols before passing data	YES	YES
ArrayIndexOutOfBoundsException	Instruction tried to access to cell by non-existent index	High	Change accessing logic to the proper one	YES	YES
NoSuchElementException	Access to element been when pointer already changed the position	High	Change accessing logic to the proper one	YES	YES
	The collection has been modified during the iteration	High	Add synchronization for collection	YES	YES

5 minute break

Catch Multiple Exceptions



Execute try block
No exception
Exception
Find appropriate catch block to execute
Exception 1
Exception 2
Exception n
Execute catch block for ExceptionType1
Execute catch block for ExceptionType2
...
Execute catch block for ExceptionTypen
Execute statement out of try-catch block

```
readFile {
    try {
        open the file;
        determine its size;
        allocate that much memory;
        read the file into memory;
        close the file;
    } catch (fileOpenFailed) {
       doSomething;
    } catch (sizeDeterminationFailed) {
        doSomething;
    } catch (memoryAllocationFailed) {
        doSomething;
    } catch (readFailed) {
        doSomething;
    } catch (fileCloseFailed) {
        doSomething;
    }
}
```

Activity #4 - Rethrow

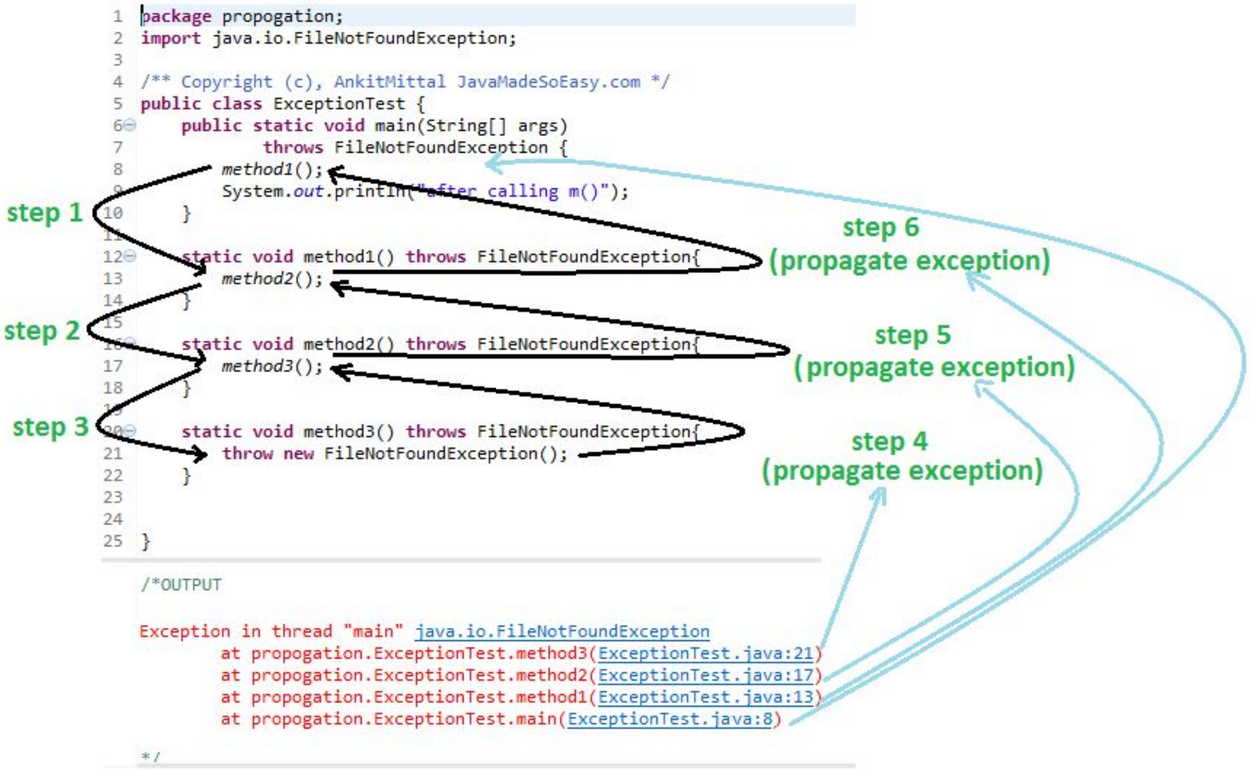
- What if we don’t want to handle exception in a called method?
  - Declare exception type in the method which can possibly occur using **throws.**
  - Simplest way to “handle” an exception is to rethrow it

```java
public int getServerConfiguration(String config)
  throws FileNotFoundException {

    Scanner contents = new Scanner(new File(config));
    return Integer.parseInt(contents.nextLine());
}
```

  - Someone calling this method must handle it using try-catch or rethrow it again.
  - Run the program in Workspace. What do you see?
    - Java Exception Propagation. Let’s find out how it works!

Java Exception Propagation



```
package propogation;
import java.io.FileNotFoundException;

/** Copyright (c), AnkitMittal JavaMadeSoEasy.com */
public class ExceptionTest {
    public static void main(String[] args)
            throws FileNotFoundException {
        method1();
        System.out.println("after calling m()");
    }

    static void method1() throws FileNotFoundException{
        method2();
    }

    static void method2() throws FileNotFoundException{
        method3();
    }

    static void method3() throws FileNotFoundException{
        throw new FileNotFoundException();
    }


}
```

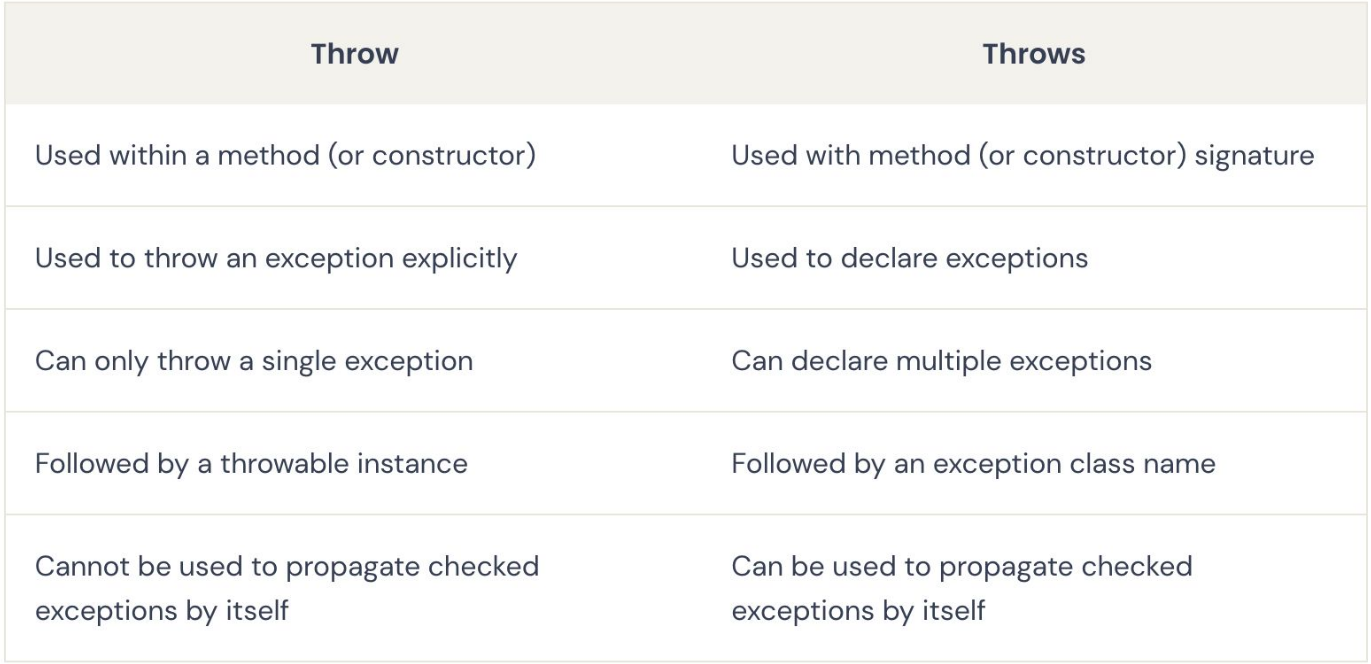
```
/*OUTPUT

Exception in thread "main" java.io.FileNotFoundException
        at propogation.ExceptionTest.method3(ExceptionTest.java:21)
        at propogation.ExceptionTest.method2(ExceptionTest.java:17)
        at propogation.ExceptionTest.method1(ExceptionTest.java:13)
        at propogation.ExceptionTest.main(ExceptionTest.java:8)

*/
```

Curious Cats

- What is difference between throw and throws?



Throw	Throws
Used within a method (or constructor)	Used with method (or constructor) signature
Used to throw an exception explicitly	Used to declare exceptions
Can only throw a single exception	Can declare multiple exceptions
Followed by a throwable instance	Followed by an exception class name
Cannot be used to propagate checked exceptions by itself	Can be used to propagate checked exceptions by itself

Why need Custom Exceptions?

- Java exceptions cover almost all general exceptions.
- However, we sometimes need to supplement these standard exceptions with our own.
  - Business logic exceptions – exceptions that are specific to the business logic and workflow. These help the application users or the developers understand what the exact problem is.
  - To catch and provide specific treatment to a subset of existing Java exceptions

Checked Custom Exception

```java
public class CustomerService {

    public Customer findByName(String name) throws NameNotFoundException {

        if ("".equals(name)) {
            throw new NameNotFoundException("Name is empty!");
        }

        return new Customer(name);

    }

    public static void main(String[] args) {

        CustomerService obj = new CustomerService();

        try {

            Customer cus = obj.findByName("");

        } catch (NameNotFoundException e) {
            e.printStackTrace();
        }

    }
}
```

```java
public class NameNotFoundException extends Exception {

    public NameNotFoundException(String message) {
        super(message);
    }

}
```

Unchecked Custom Exception

```java
public class CustomerService {

    public void analyze(List<String> data) {

        if (data.size() > 50) {
            //runtime exception
            throw new ListTooLargeException("List can't exceed 50 items!");
        }

                //...
    }

    public static void main(String[] args) {

        CustomerService obj = new CustomerService();

                //create 100 size
        List<String> data = new ArrayList<>(Collections.nCopies(100, "mkyong"));

        obj.analyze(data);


    }
}
```

```java
public class ListTooLargeException extends RuntimeException{

    public ListTooLargeException(String message) {
        super(message);
    }

}
```

Activity #5 - Bank Account

- Run the program. Try out invalid operations as well. What do you see?
- How can we ensure correctness of the program?
  - Handle using Custom Exceptions.
- What changes can be done?
  - Create two Custom Exceptions
    - InvalidAmount Exception
    - InsufficientBalance Exception
  - Declare Exceptions in methods (Both in Interface and Class implementing it.)
  - Replace Error Printing with throw Exceptions
  - Handle Exceptions at Class calling those methods with try catch.

QCalc - Module 3: Apply OOPS

- In this module:
  - Support arithmetic operations for floating point values. (double data types)
    - Instead of modifying existing methods of calculators do the following:-
      - Use Method Overloading for double data types
  - Write Unit Tests for the new overloaded methods as well
  - Execute the unit tests written for the above method to verify correctness of the implementation.
  - Submit the code for assessment.

QCalc - Module 4: Debug and Handle Exceptions

- In this module:
  - Debug and Handle Exceptions for invalid data.
  - Write unit test to validate the Exception being thrown.

Questions

1. What is exception handling in Java, and why is it important?
2. What is the difference between checked and unchecked exceptions in Java?
3. What is the purpose of the try-catch-finally block in Java exception handling?
4. What is the purpose of the throw keyword in Java exception handling?
5. What are the common practices for exception handling in Java?

Session Revision Quiz

Quiz Link

Solve this quiz to access your understanding of session’s topics clearly

Take home exercises for the session

- You will have to complete the below modules of QCalc Micro-Experience**:**
    - Module 3: Apply OOPS
    - Module 4: Debug and Handle Exception

Further Reading

- Create a Custom Exception in Java | Baeldung
- ClassNotFoundException vs. NoClassDefFoundError - DZone Java
- Null Pointer Exception In Java - GeeksforGeeks
- Why, When and How to Implement Custom Exceptions in Java – Stackify
- Unchecked Exceptions — The Controversy (The Java™ Tutorials > Essential Java Classes > Exceptions) (oracle.com)
- Difference between java.lang.RuntimeException and java.lang.Exception - Stack Overflow
- Chained Exceptions in Java with Example - Scientech Easy

References

- Java Exceptions - Checked vs Unchecked (howtodoinjava.com)
- Java Lab 8, Exceptions (programming.vip)
- Exception Handling in Java | Baeldung
- Java Exceptions - DZone Java

Thank you