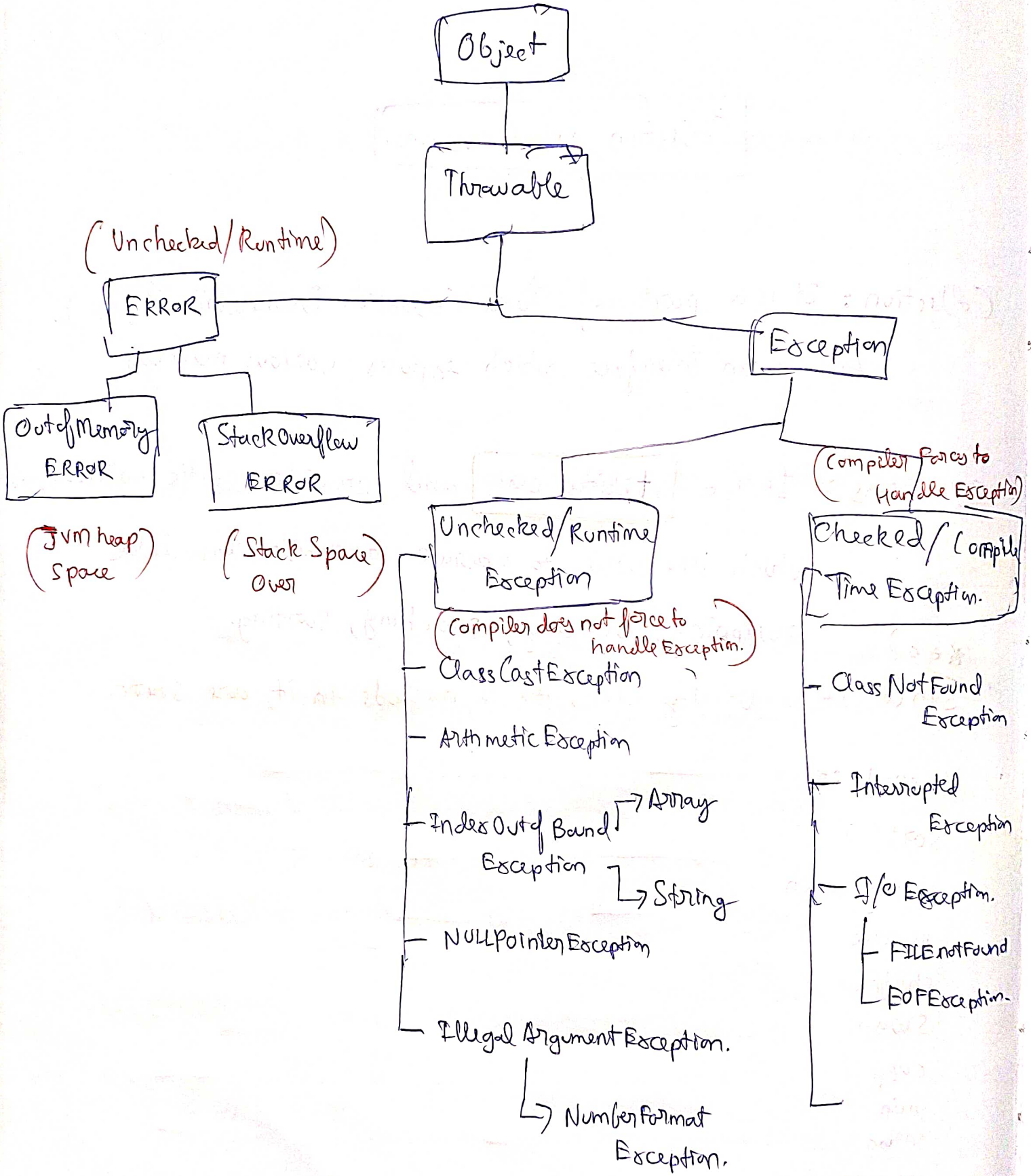
19. Exception Handling In Java with Examples.

Hierarchy of Exception Objects



~~Generally when~~

Whenever there is an Exception. Java throws us an Exception Object, which needs to be handled.

Unchecked/Runtime Exception.

```
public class Main {

    public static void Main(){
        method1();
    }

    public static void method1(){
        throw new ArithmeticException();
    }
}
```

→ Compiler not forcing us to handle it.

Checked/CompileTime Exception.

```
psvm(){
    method1();
}

method1(){
    throw new ClassNotFoundException();
}
```

→ Compiler will force you to handle this exception.

Exception chaining

```
method1(){
    caller1();
}

caller1(){
    caller2();
}

caller2(){

}
```

Goes to main() method Eventually and throws Exception

Stack Trace

↑ If No handled.

Check in caller method if exception handled or Not.

↑ If No

→ If this throws Exception.
It will check if Exception is handled here or Not.

[Handling Exception using throws keyword]

All the parents of the method all the ancestors need to use the throws keyword to handle Exception.

```
caller1(){ caller2();
}
```

Use catch block or throw to caller.

throws keyword says this method might throw Exception, So it tells caller to handle it

```
caller2() throws ClassNotFoundException {

}.
```

(88) [Handling Exception Using try/catch].

This is actual Exception Handling - Wherever this is written, the exception is handled there itself and does not propagate to caller.

```
method(){

    try{
        throws new ClassNotFoundException();
    }
    catch(Exception e){
        e.stackTrace();
    }
}
```

Example → Delegating to Caller()

```
method(){

    try{ method1(); }
    catch(Exception e){
        e.printStackTrace();
    }
}
```

method1() throws ClassNotFoundException

(Caller handles Exception)

```
method1() throws ClassNotFoundException{
    throws new ClassNotFoundException();
}
```

[Multiple Exception Catching]

```
psvm() {
    try {
        method();
    }
    catch (ClassNotFoundException | InterruptedException e) {

    }
    catch (FileNotFoundException e)
}
```

→ Error since try block does not throw this ERROR.

```
void method() throws ClassNotFoundException, InterruptedException{

}
```

Finally block

* Finally block will always get executed. Even if Return Statement is there in try.
* We can only add 1 finally block in a scope
* In JVM related issues like (Out of Memory) [Heap Issue], process is forcefully killed. Process Forcefully Killed. Finally block is not executed then.