Exceptions

Errors:-

- Compile Time
- Runtime
- Logical Error.

→ Exceptions are Runtime Errors.

Eg:- Divide by zero. error.
Arithmetic Exception.

→ try{} block is executed, if exception occurs, executions flows to the catch block.

And after catch block.
rest of code is executed.

CODE:-

```
public class Demo{
    public static void main(String a[]) {

        int i = 0;
        int j = 0;

        try
        {
            j = 18/i;
        }
        catch(Exception e)
        {
            System.out.println(x: "Something Went Wrong
        }


        System.out.println(j);

        System.out.println(x: "Bye");
```

OUTPUT:-

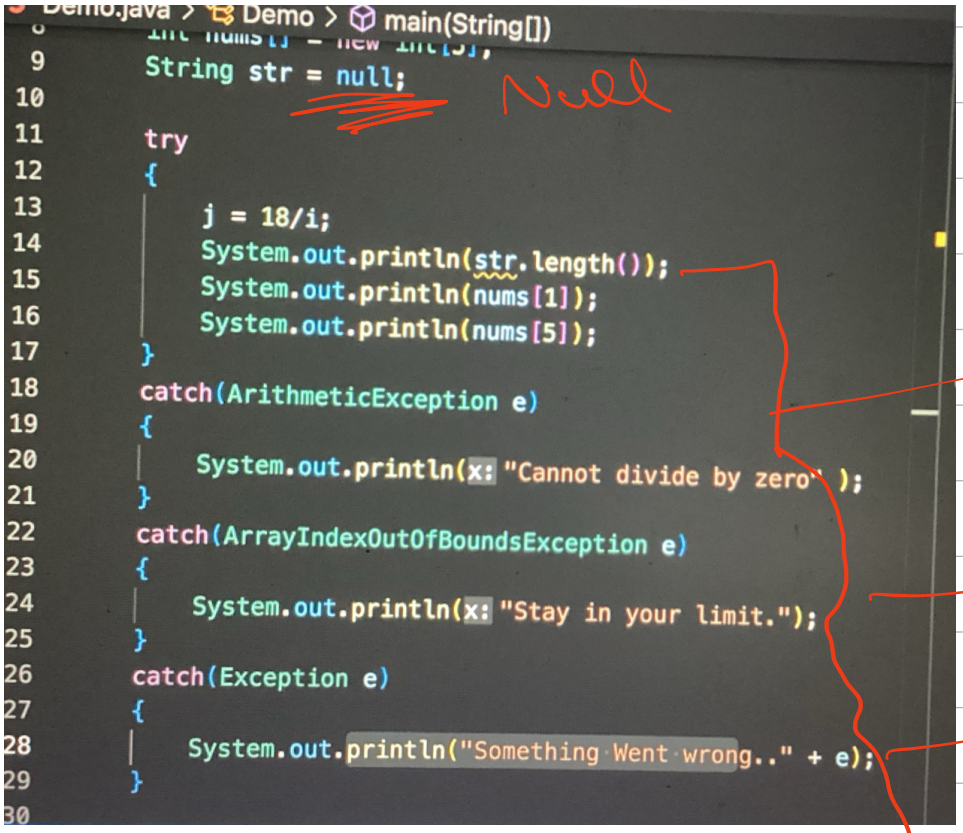
```
Something Went Wrong..
0
Bye
```

How to handle different exceptions:-

- Arithmetic Exception → To handle divide by zero

- ArrayIndexOutOfBoundsException → To array index out of bound.

- Exception → To handle any exception.



→ Handles / by 0

→ array [5]

→ Null Pointer exception as str = null;

- String str = null;

Note:-

Parent Exception classes should be written below child.

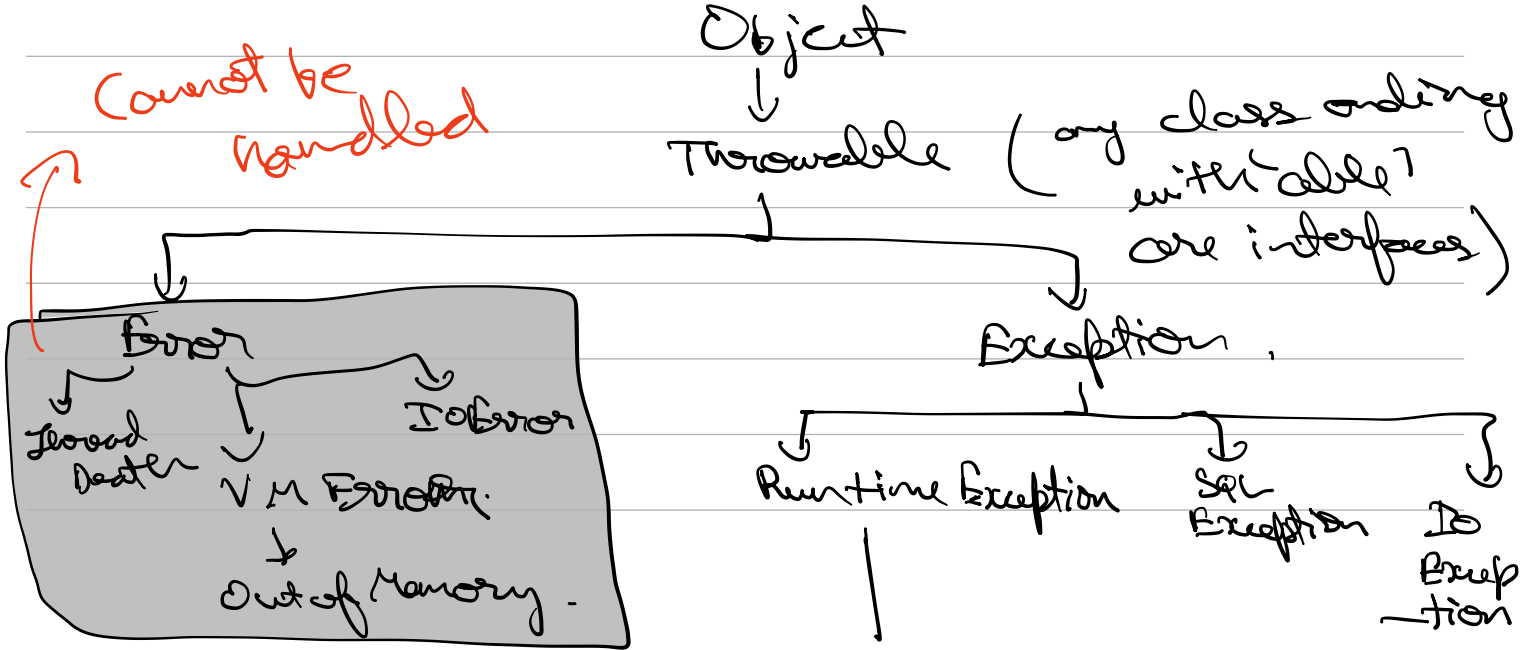
For example:-

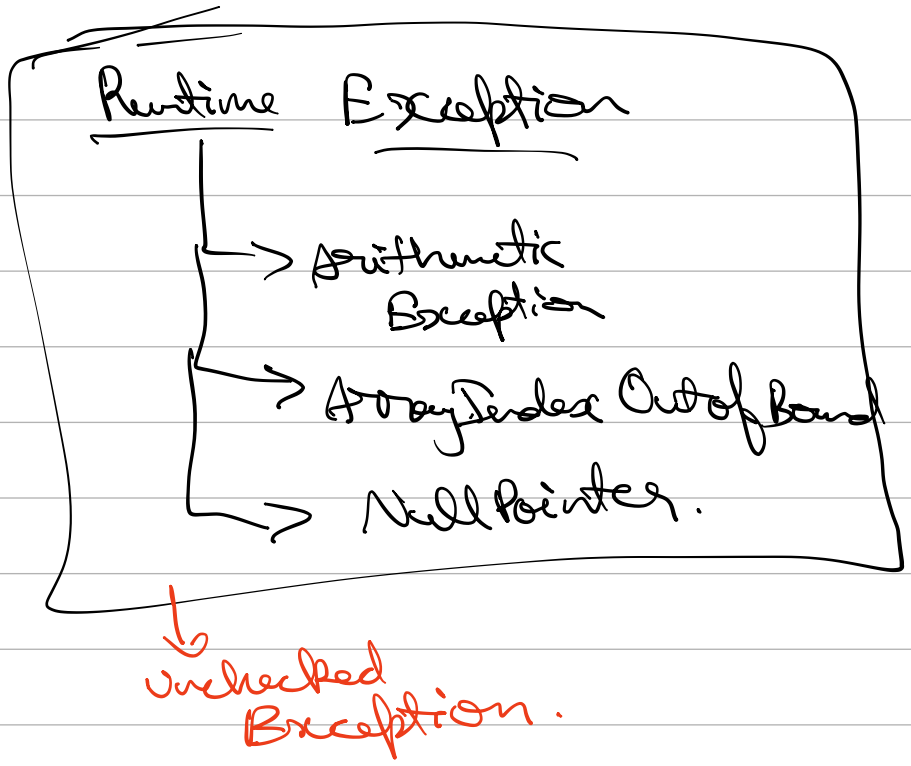
```
catch (ArithmeticException e)
{
    .  .  .  .  .
}

catch (Exception e)
{
    .  .  .  .  .
}
```

child

Parent

Exception Hierarchy





Unchecked Exception.

SQL Exception and IO Exception ↓ are Checked Exception

Checked Exception:- Are exception which we are forced to handle

Unchecked Exception:- May or may not handle.

Throw keyword :-

-> It is used to throw an exception.

Syntax:- throw new ArithmeticException();

Code :-

```
2   :lass Demo{
3   .ic static void main(String a[]) {
4
5    int i = 20;
6    int j = 0;
7
8    try {
9        j = 18/i;
10       if(j==0)
11           throw new ArithmeticException();
12   }
13   catch(ArithmeticException e) {
14       j = 18/1;
15       System.out.println(x: "thats the default output ");
16   }
17   catch(Exception e) {
18       System.out.println("Something Went wrong.." + e);
19   }
20
21   System.out.println(j);
```

→ Here we get $j = 18/20 = 0$.

So if(j==0) throwing exception.

↳ means calling this block.

☆). We can pass message when we throw an exception.

Code :-

```
= 20;
= 0;


= 18/i;
(j==0)
  throw new ArithmeticException(s:"I dont want to print ze

ArithmeticException e) {
= 18/1;
stem.out.println("thats the default output " + e);

Exception e) {
stem.out.println("Something Went wrong.." + e);


.out.println(j);
```

→ This message is printed here

Custom Exceptions

```java
public class CustomException {
    public static void main(String[] args) {


        int i = 10;
        int j = 0;
        int num = 0;
        try {
            num = i / j;

        } catch (ArithmeticException e) {
            System.out.println(e);
        }
        System.out.println("program Continued");
        try{
            num=j/i;
            if(num==0)
                throw new VarunException("This is Custom Exception !");
        }
        catch (VarunException e){
            System.out.println(e);
        }
    }
}
```

→ Custom exception called when num == 0.

catch block of custom exception.

This class should be defined to handle exception.

```java
2 usages
class VarunException extends Exception {
    1 usage
    public VarunException(String obj){
        super(obj);
    }
}
```

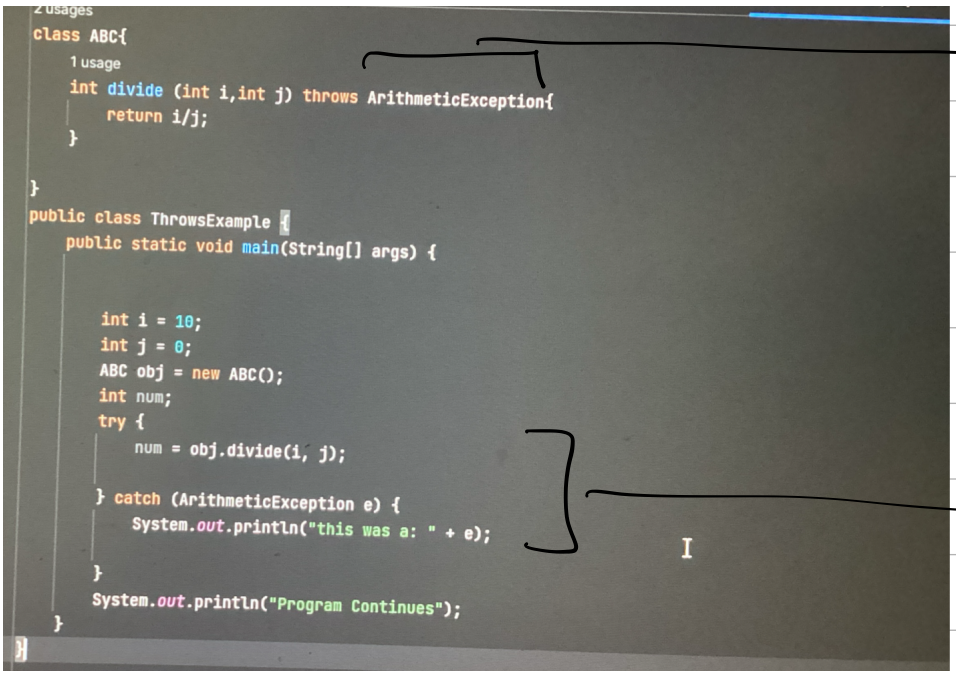
→ This is default parent class for Custom Exceptions.

Super class constructor is called to take string as input and print that message.

Throws keyword :-

-> This keyword is used to throws exception from a method.

This exception is chandled by the method calling it.



```
2 usages
class ABC{
    1 usage
    int divide (int i,int j) throws ArithmeticException{
        return i/j;
    }

}
public class ThrowsExample {
    public static void main(String[] args) {


        int i = 10;
        int j = 0;
        ABC obj = new ABC();
        int num;
        try {
            num = obj.divide(i, j);

        } catch (ArithmeticException e) {
            System.out.println("this was a: " + e);

        }
        System.out.println("Program Continues");
    }
}
```

-> divide() method throws exception to main() method. if j = 0.

-> This exception is handled here.

Output :-

```
this was a: java.lang.ArithmeticException: / by zero
Program Continues

Process finished with exit code 0
```