Exception :

A typical or exceptional condition that signals a piece of code could not execute normally.

→ Exceptions are objects like everything in j
so you can instantiate: Exception e = New Exception

Constructors :-

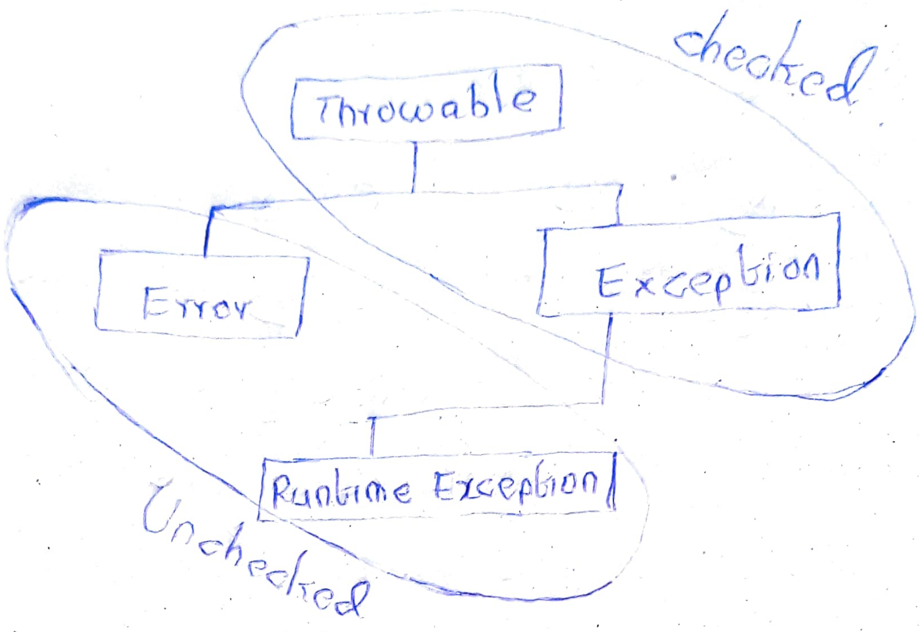
Exception ()

Exception (message)

Exception (cause)
↑ which takes another exception

Exception (message, cause)

The Exception Hierarchy:-



→ When we are handling checked exception you have to mention only possible exceptions in try block. If we try to catch other checked exceptions it will throw $\overset{CE}{\text{error}}$.

Catch rules:

1) catch exception only once
2) subclasses must be caught before their super classes
3) Dont catch checked exceptions that couldn't be thrown. from code present in try

multi-catch :- java 7

Before - 7 We have to write seperate catch block for every exception.

```
try {
    } catch(E... 1) { }
      catch(T... 2) { }
```

After 7 :

We can mention multiple Exceptions in single catch block.

```
Catch (Exceptionone | Exceptiontwo | Exceptionthree e)
{
    e.getMessage(); //
}
```

Rules :

Exceptions are seperated by a pipe and there's only one reference

Use Exceptions not related through inheritance → if they are related mention super class exception

The reference is final.

Finally :

```
try{
    system.exit(0);
} catch (Ex-- e)
{ }
finally {
}
```

→ This will terminate program abnormally without executing finally block

```
finally{
        file.close();
    }
```

but this resource closing also rises exception if file ~~there~~ opening is not done in try block.

so for f.close() also we have to provide condition like

```
finally {
        if( file != null)
           {
             file.close();
           }
        else{ }
}
```

→ So to overcome this problem java provides Try-with-resource block which closes resources automatically. But on one condition → those resources should implome java.lang. AutoCloseable
java.io. closeable.

So there we dont need finally block to close resources.

→ Initialized from left to right but closed in reverse order

→ supressed exceptions.

Advantages of checked Exceptions :-

→ Document a method
→ Force to deal with exception
→ Help write more robust programs.

Advandages of unchecked Exceptions
catch only what you want
Less cluttered code
provide more flexibility :

Creating ~~the~~ custom exceptions :-

Use checked exceptions for recoverable conditions and runtime exeptions for programming errors.