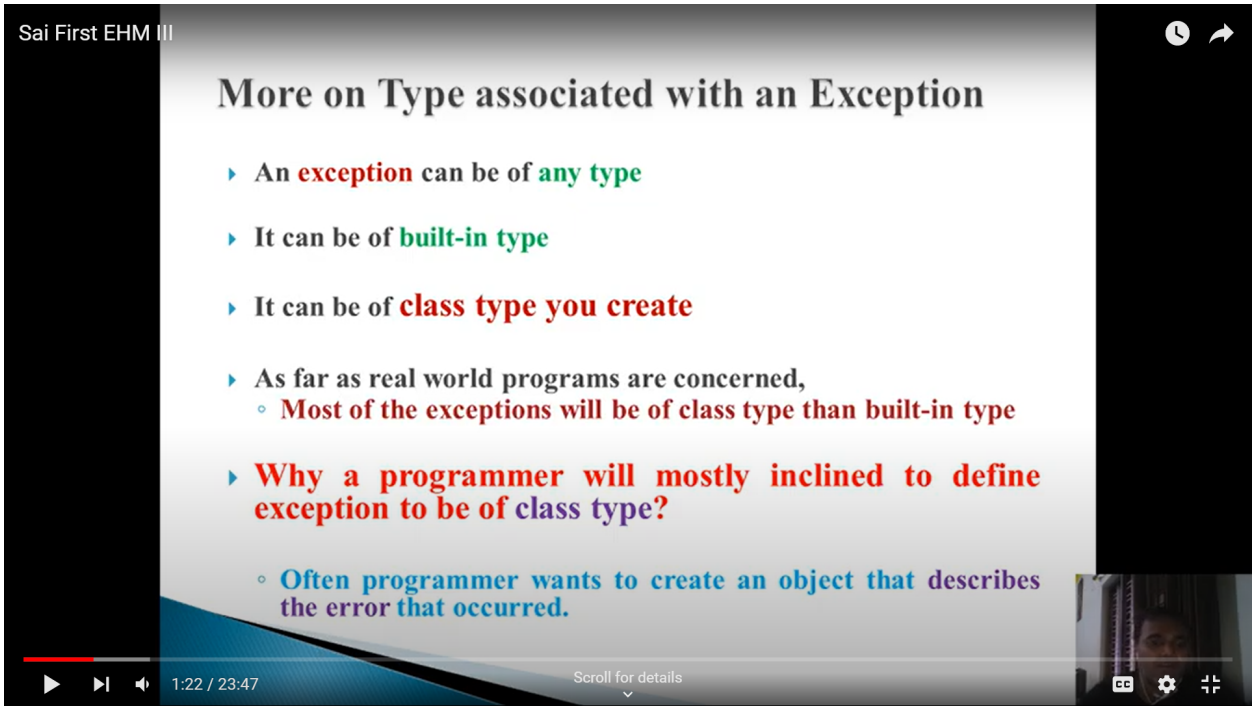
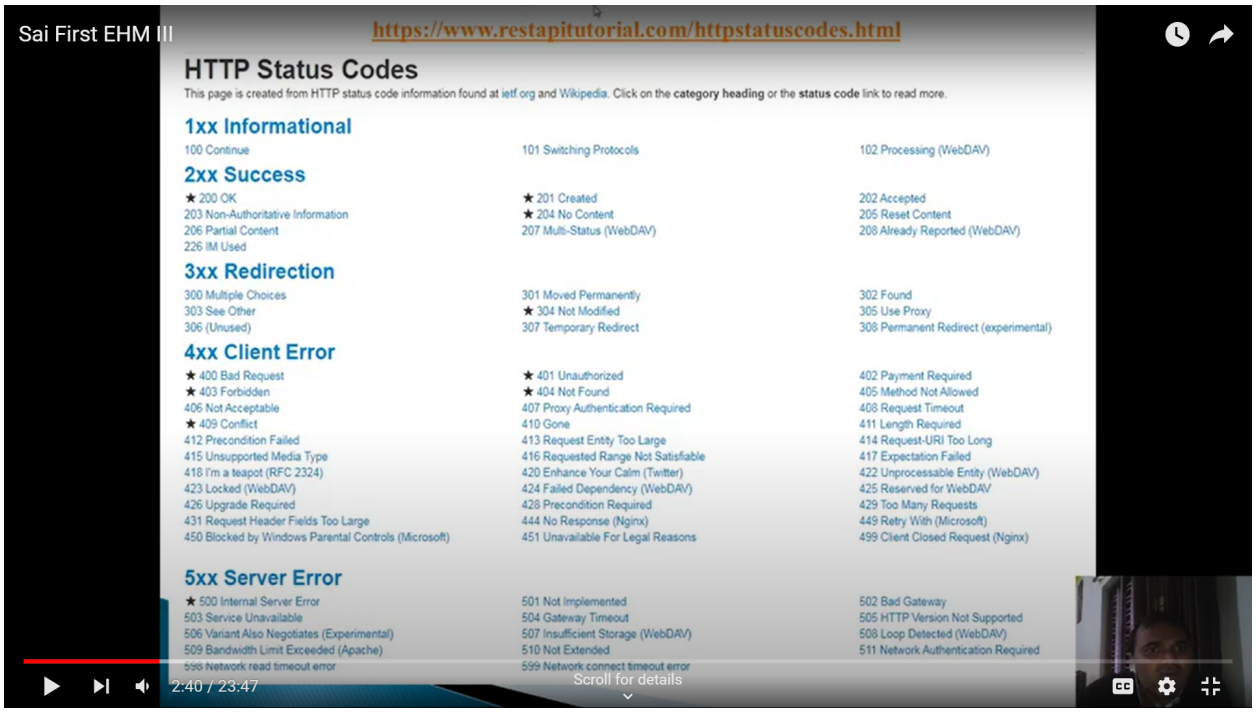
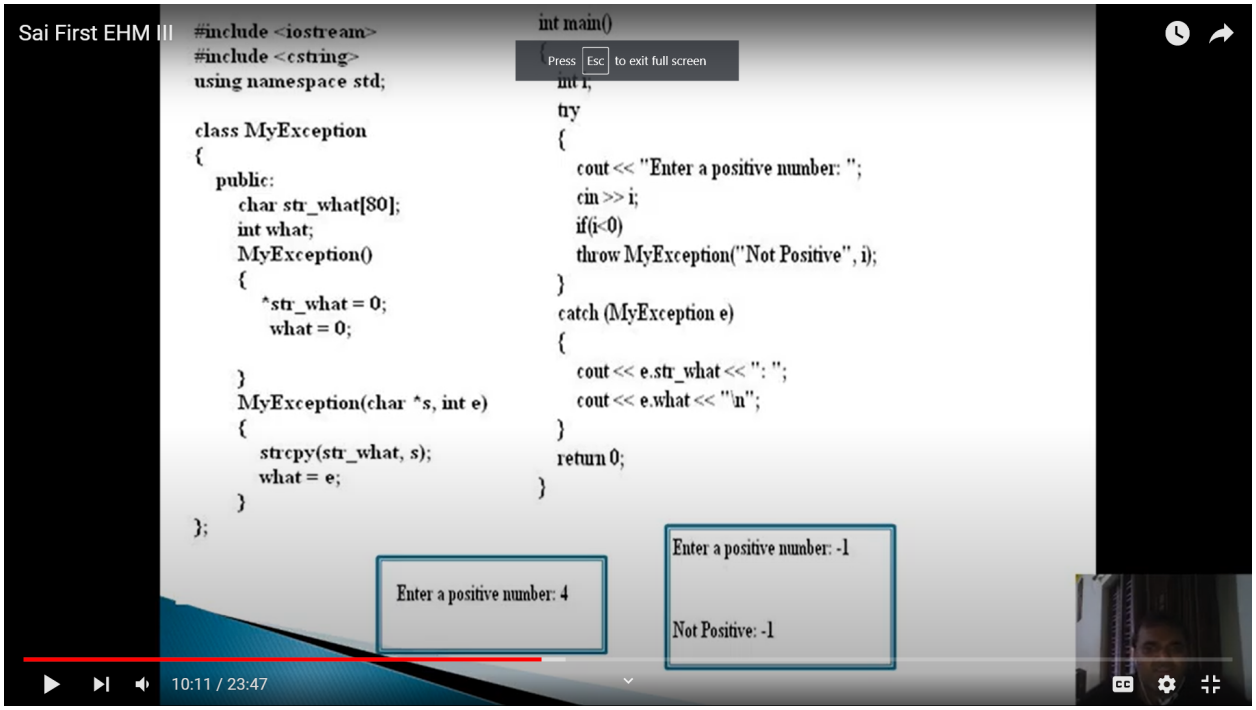
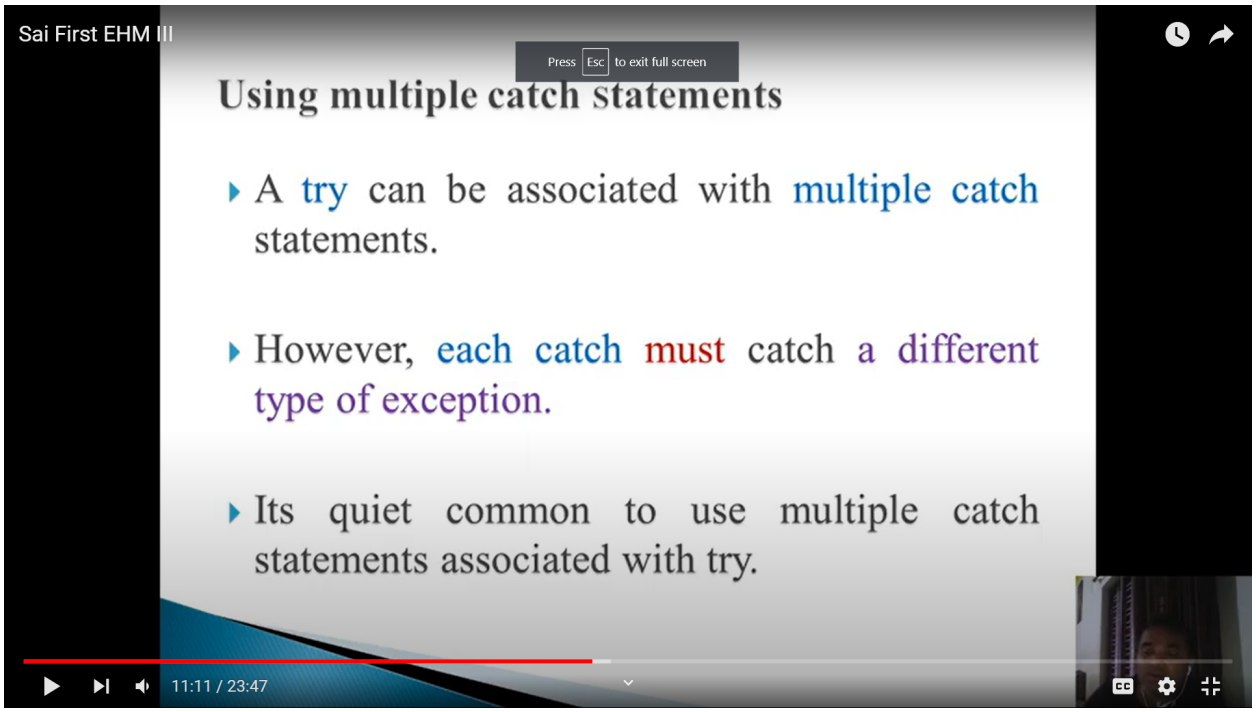
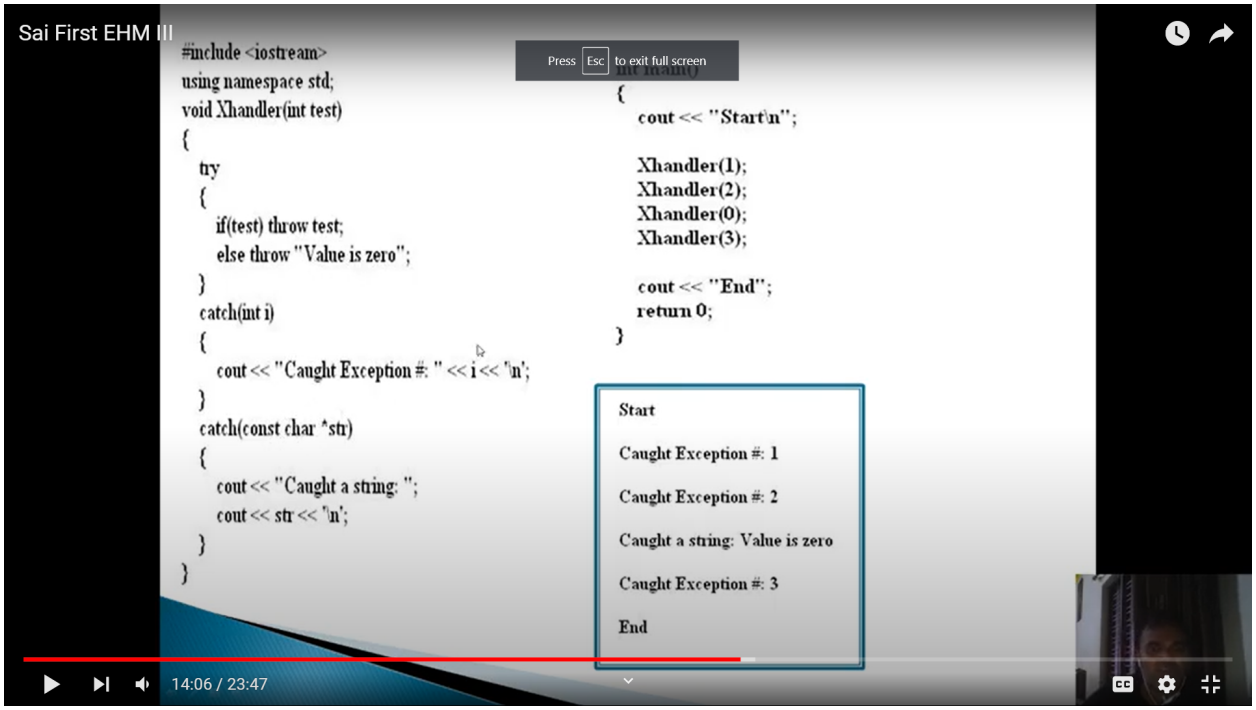
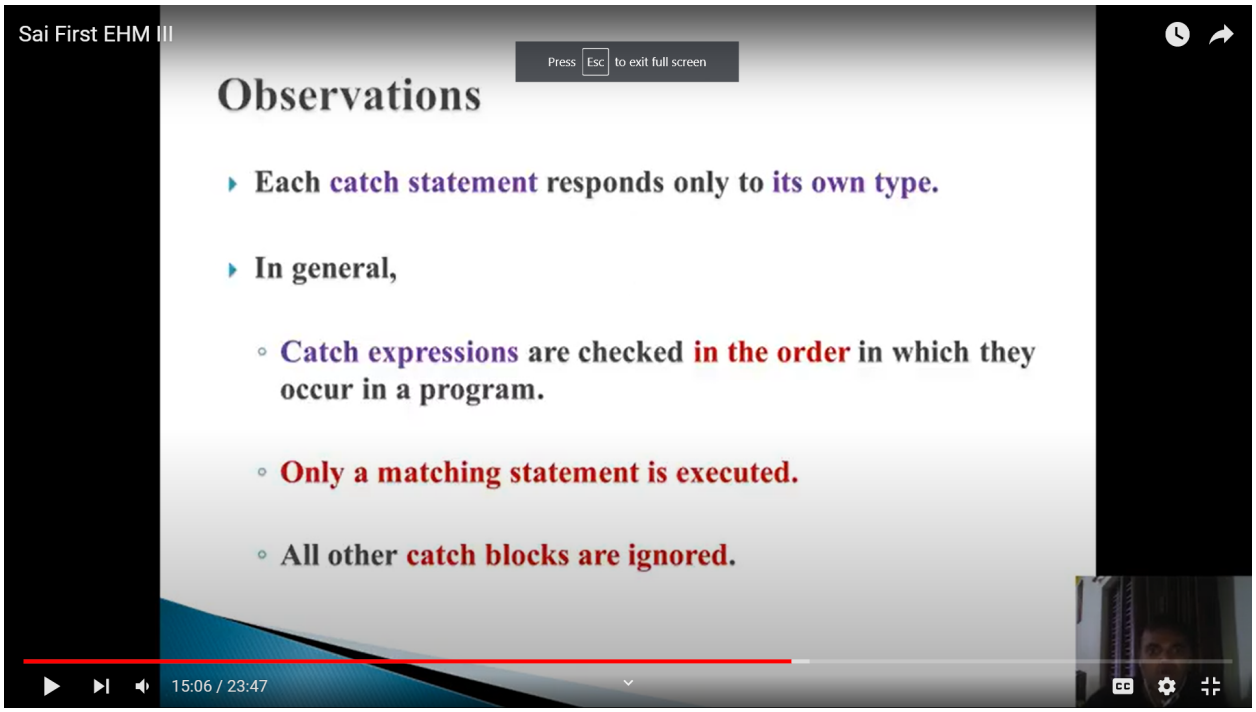

Sai First EHM III
More on Type associated with an Exception
An exception can be of any type
It can be of built-in type
It can be of class type you create
As far as real world programs are concerned,
Most of the exceptions will be of class type than built-in type
Why a programmer will mostly inclined to define exception to be of class type?
Often programmer wants to create an object that describes the error that occurred.
1:22 / 23:47
Scroll for details

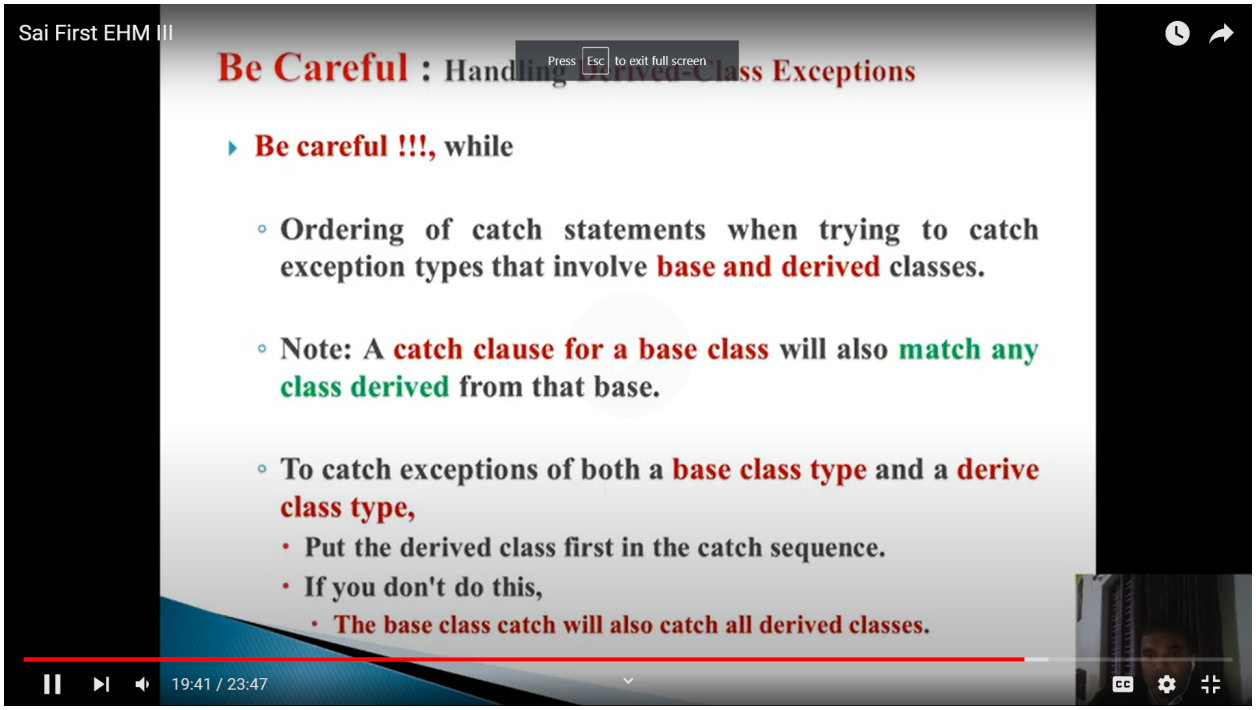

Sai First EHM III
https://www.restapitutorial.com/httpstatuscodes.html
HTTP Status Codes
This page is created from HTTP status code information found at ietf.org and Wikipedia. Click on the category heading or the status code link to read more.
1xx Informational
100 Continue
101 Switching Protocols
102 Processing (WebDAV)
2xx Success
★ 200 OK
203 Non-Authoritative Information
206 Partial Content
226 IM Used
★ 201 Created
★ 204 No Content
207 Multi-Status (WebDAV)
202 Accepted
205 Reset Content
208 Already Reported (WebDAV)
3xx Redirection
300 Multiple Choices
303 See Other
306 (Unused)
301 Moved Permanently
★ 304 Not Modified
307 Temporary Redirect
302 Found
305 Use Proxy
308 Permanent Redirect (experimental)
4xx Client Error
★ 400 Bad Request
★ 403 Forbidden
406 Not Acceptable
★ 409 Conflict
412 Precondition Failed
415 Unsupported Media Type
418 I'm a teapot (RFC 2324)
423 Locked (WebDAV)
426 Upgrade Required
431 Request Header Fields Too Large
450 Blocked by Windows Parental Controls (Microsoft)
★ 401 Unauthorized
★ 404 Not Found
407 Proxy Authentication Required
410 Gone
413 Request Entity Too Large
416 Requested Range Not Satisfiable
420 Enhance Your Calm (Twitter)
424 Failed Dependency (WebDAV)
428 Precondition Required
444 No Response (Nginx)
451 Unavailable For Legal Reasons
402 Payment Required
405 Method Not Allowed
408 Request Timeout
411 Length Required
414 Request-URI Too Long
417 Expectation Failed
422 Unprocessable Entity (WebDAV)
425 Reserved for WebDAV
429 Too Many Requests
449 Retry With (Microsoft)
499 Client Closed Request (Nginx)
5xx Server Error
★ 500 Internal Server Error
503 Service Unavailable
506 Variant Also Negotiates (Experimental)
509 Bandwidth Limit Exceeded (Apache)
598 Network read timeout error
501 Not Implemented
504 Gateway Timeout
507 Insufficient Storage (WebDAV)
510 Not Extended
599 Network connect timeout error
502 Bad Gateway
505 HTTP Version Not Supported
508 Loop Detected (WebDAV)
511 Network Authentication Required
2:40 / 23:47
Scroll for details

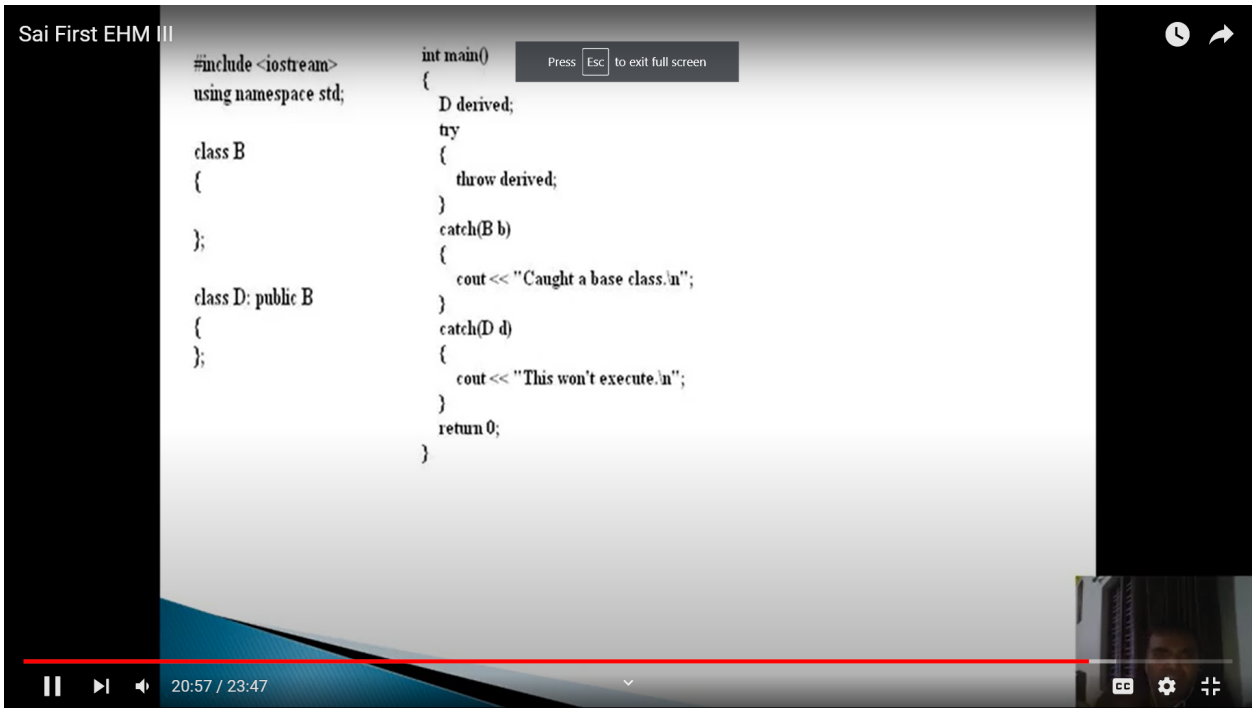

Sai First EHM III
Press Esc to exit full screen
#include <iostream>
#include <cstring>
using namespace std;
class MyException
{
public:
char str_what[80];
int what;
MyException()
{
*str_what = 0;
what = 0;
}
MyException(char *s, int e)
{
strcpy(str_what, s);
what = e;
}
};
int main()
{
int i;
try
{
cout << "Enter a positive number: ";
cin >> i;
if(i<0)
throw MyException("Not Positive", i);
}
catch (MyException e)
{
cout << e.str_what << ": ";
cout << e.what << "\n";
}
return 0;
}
Enter a positive number: 4
Enter a positive number: -1
Not Positive: -1
10:11 / 23:47

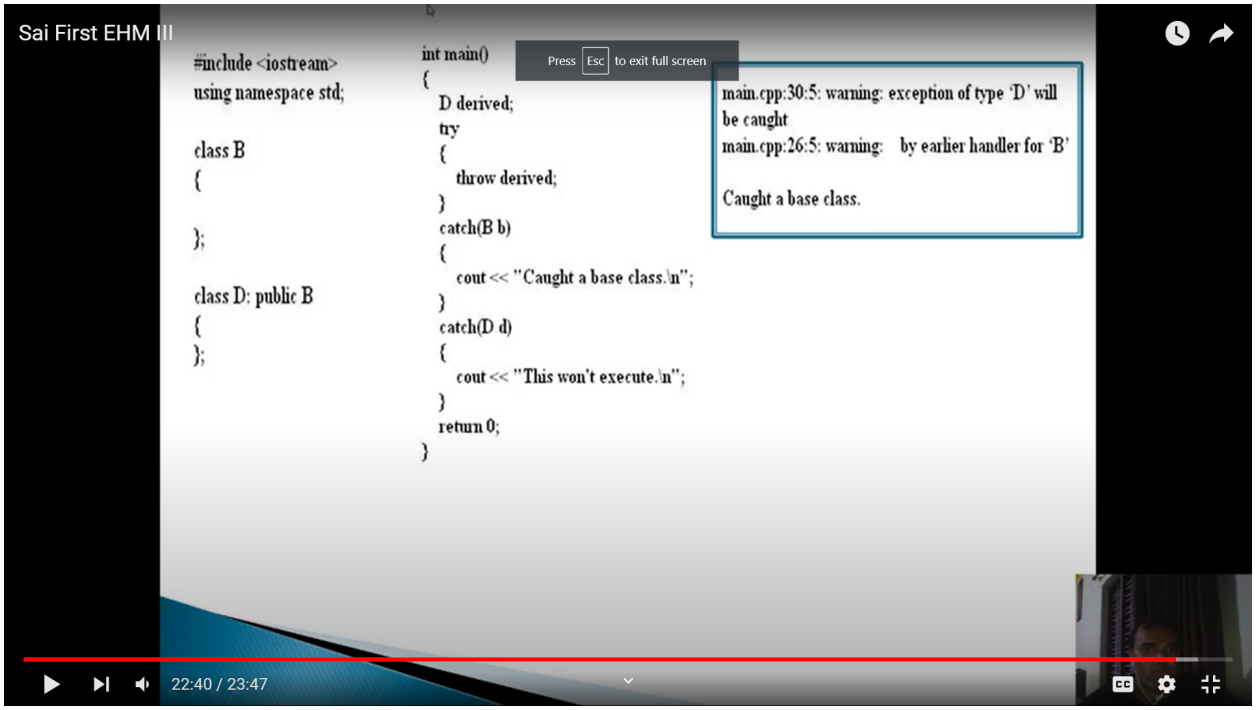

Sai First EHM III
Press Esc to exit full screen
Using multiple catch statements
A try can be associated with multiple catch statements.
However, each catch must catch a different type of exception.
Its quiet common to use multiple catch statements associated with try.
11:11 / 23:47

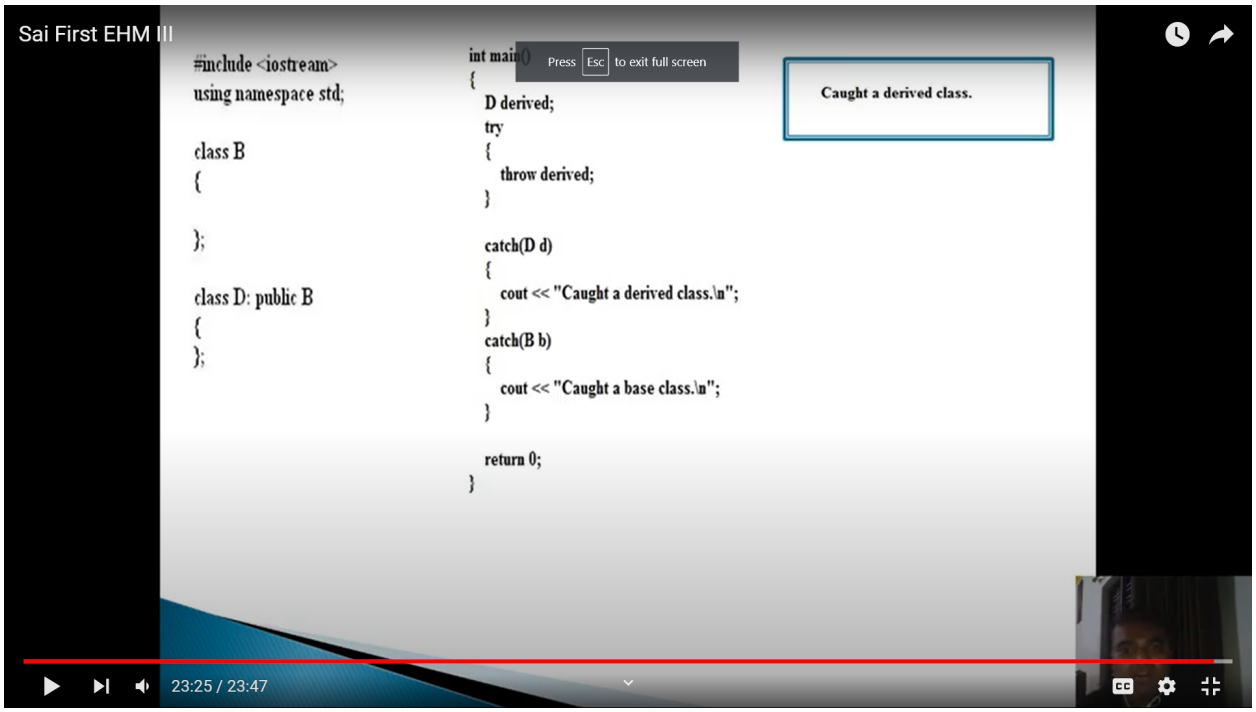

Sai First EHM III
Press Esc to exit full screen
#include <iostream>
using namespace std;
void Xhandler(int test)
{
try
{
if(test) throw test;
else throw "Value is zero";
}
catch(int i)
{
cout << "Caught Exception #: " << i << '\n';
}
catch(const char *str)
{
cout << "Caught a string: ";
cout << str << '\n';
}
}
int main()
{
cout << "Start\n";
Xhandler(1);
Xhandler(2);
Xhandler(0);
Xhandler(3);
cout << "End";
return 0;
}
Start
Caught Exception #: 1
Caught Exception #: 2
Caught a string: Value is zero
Caught Exception #: 3
End
14:06 / 23:47


Sai First EHM III
Press Esc to exit full screen
Observations
Each catch statement responds only to its own type.
In general,
Catch expressions are checked in the order in which they occur in a program.
Only a matching statement is executed.
All other catch blocks are ignored.
15:06 / 23:47


Sai First EHM III
Press Esc to exit full screen
Be Careful : Handling Derived-Class Exceptions
Be careful !!!, while
Ordering of catch statements when trying to catch exception types that involve base and derived classes.
Note: A catch clause for a base class will also match any class derived from that base.
To catch exceptions of both a base class type and a derive class type,
Put the derived class first in the catch sequence.
If you don't do this,
The base class catch will also catch all derived classes.
19:41 / 23:47


Sai First EHM III
Press Esc to exit full screen
#include <iostream>
using namespace std;
class B
{
};
class D: public B
{
};
int main()
{
D derived;
try
{
throw derived;
}
catch(B b)
{
cout << "Caught a base class.\n";
}
catch(D d)
{
cout << "This won't execute.\n";
}
return 0;
}
20:57 / 23:47


Sai First EHM III
Press Esc to exit full screen
#include <iostream>
using namespace std;
class B
{
};
class D: public B
{
};
int main()
{
D derived;
try
{
throw derived;
}
catch(B b)
{
cout << "Caught a base class.\n";
}
catch(D d)
{
cout << "This won't execute.\n";
}
return 0;
}
main.cpp:30:5: warning: exception of type 'D' will be caught
main.cpp:26:5: warning: by earlier handler for 'B'
Caught a base class.
22:40 / 23:47


Sai First EHM III
Press Esc to exit full screen
#include <iostream>
using namespace std;
class B
{
};
class D: public B
{
};
int main()
{
D derived;
try
{
throw derived;
}
catch(D d)
{
cout << "Caught a derived class.\n";
}
catch(B b)
{
cout << "Caught a base class.\n";
}
return 0;
}
Caught a derived class.
23:25 / 23:47