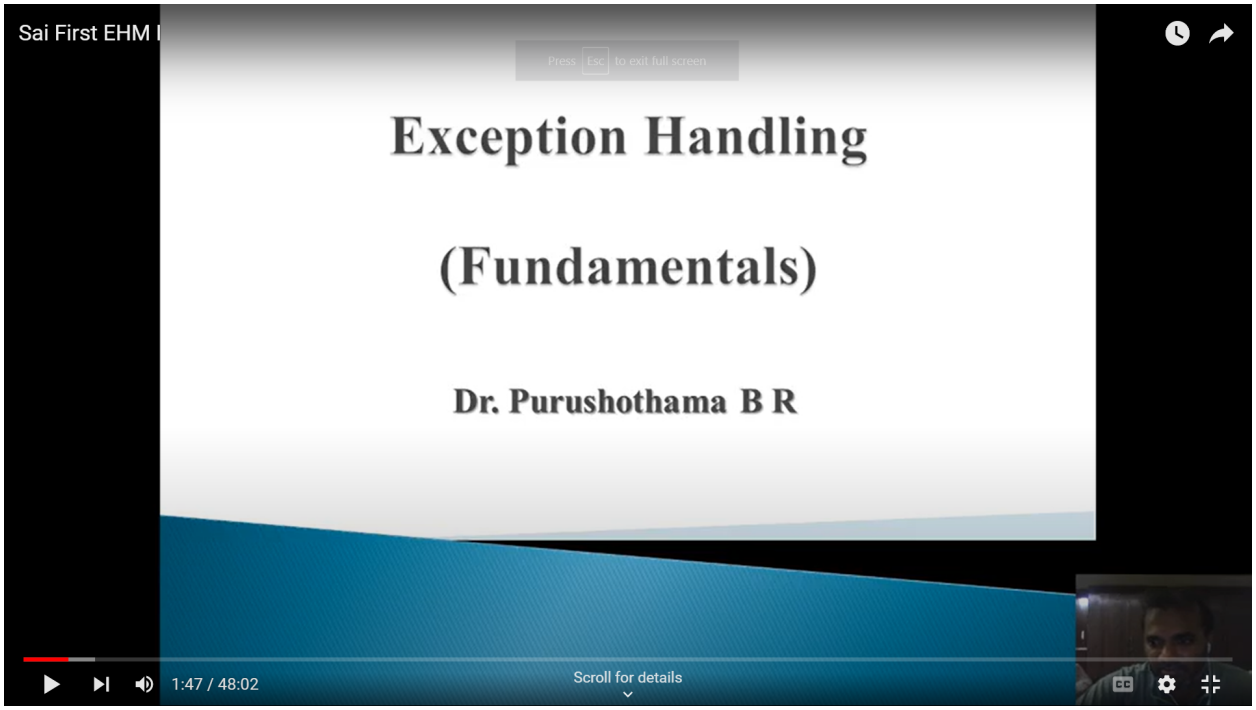
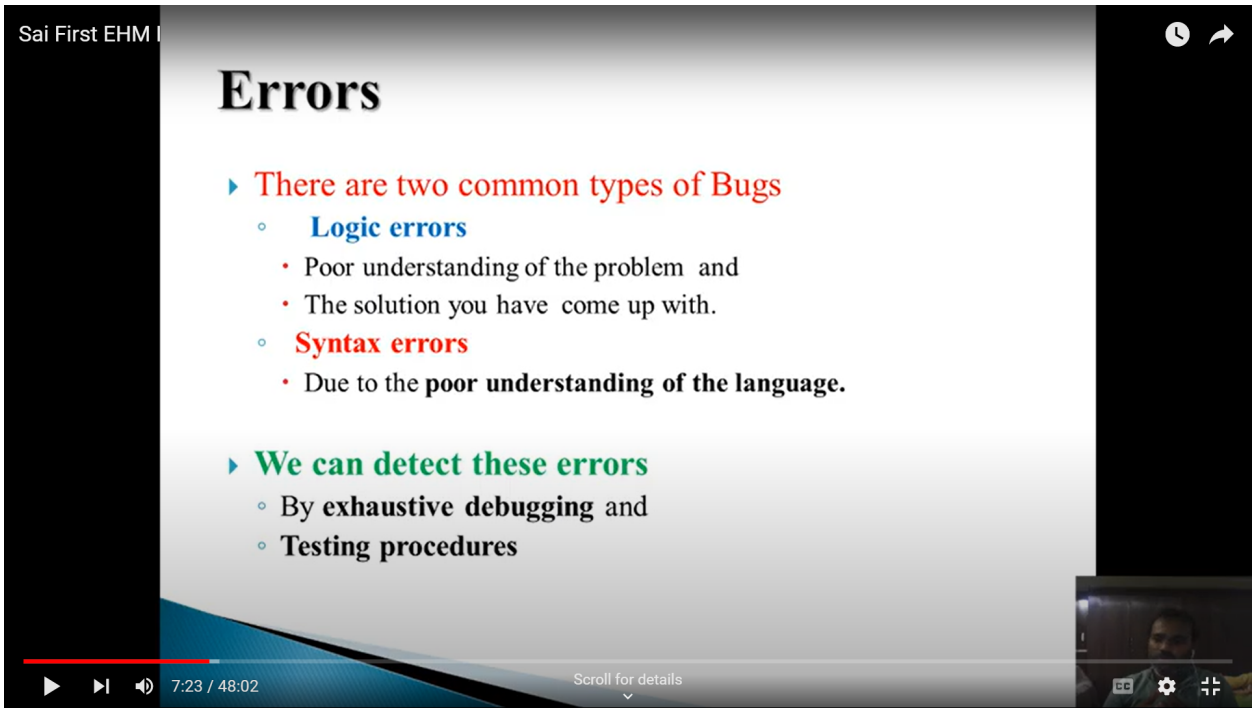
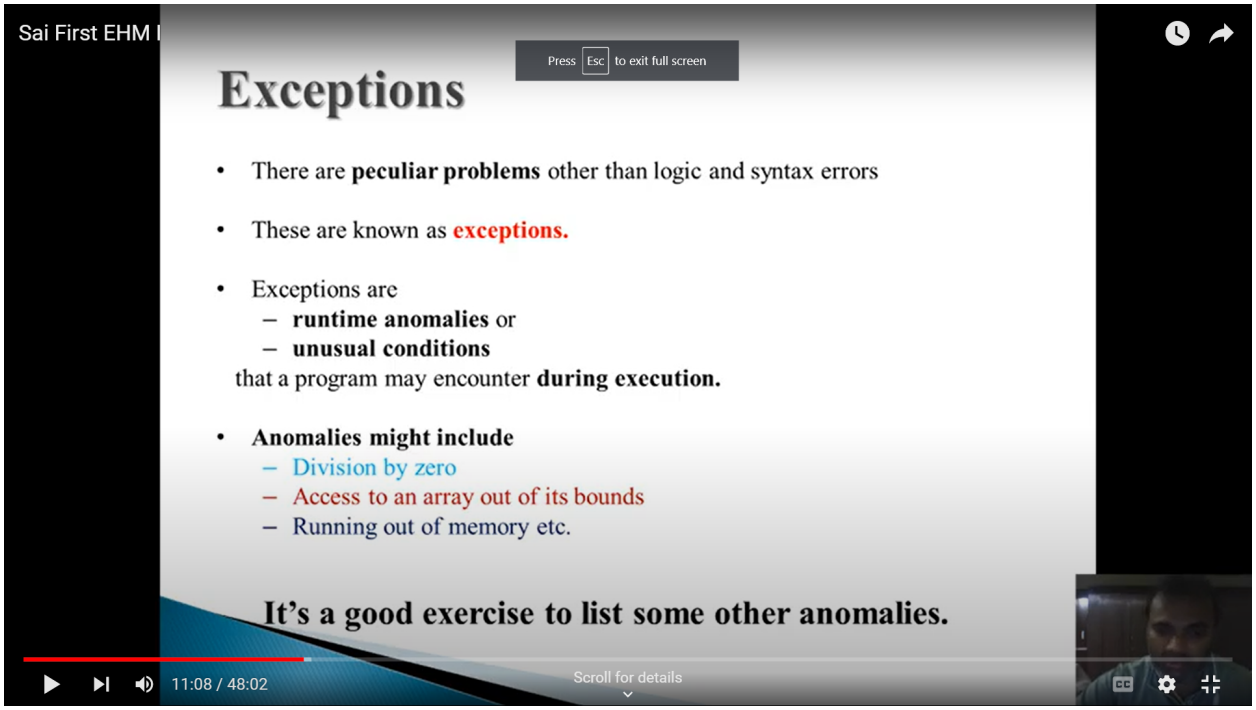
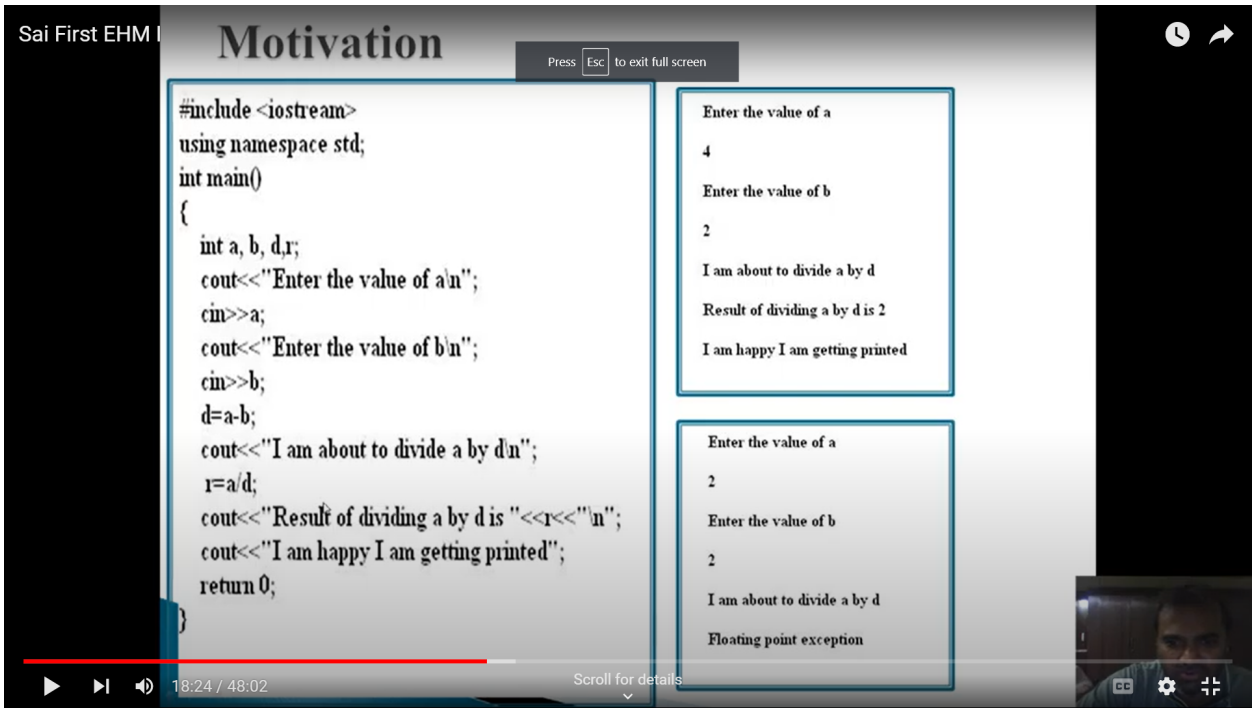
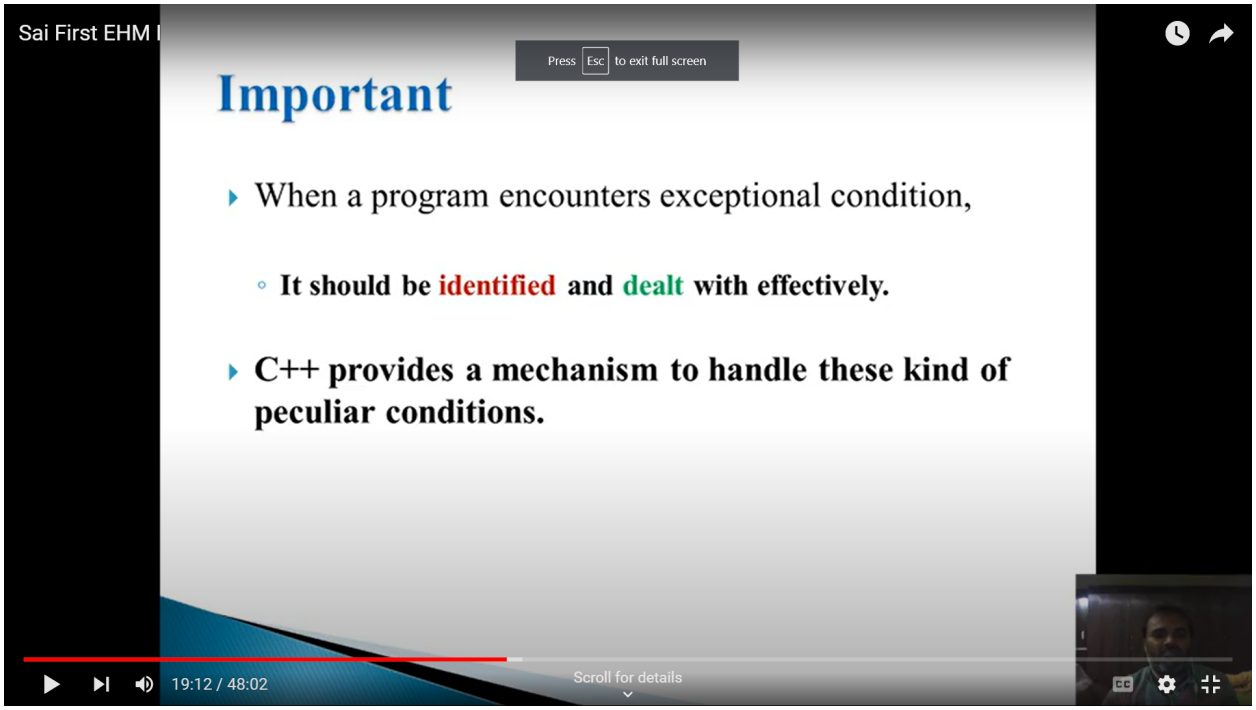
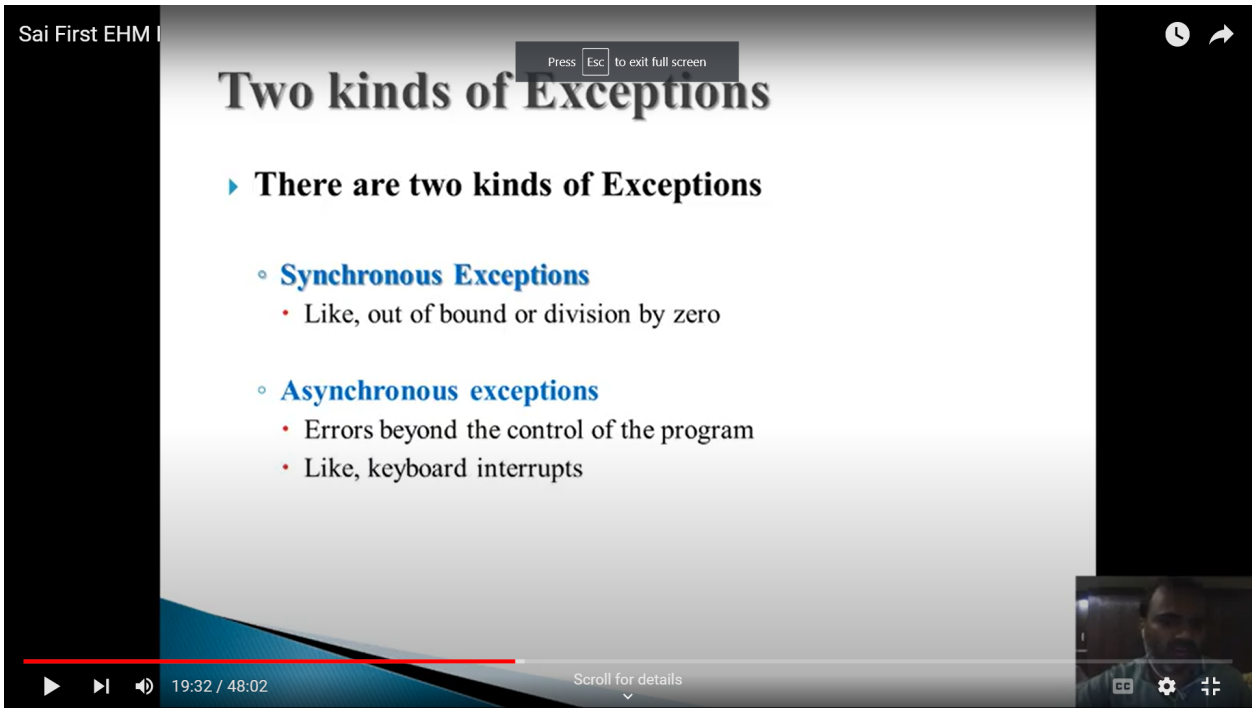

Sai First EHM
Press Esc to exit full screen
Exception Handling
(Fundamentals)
Dr. Purushothama B R
1:47 / 48:02
Scroll for details

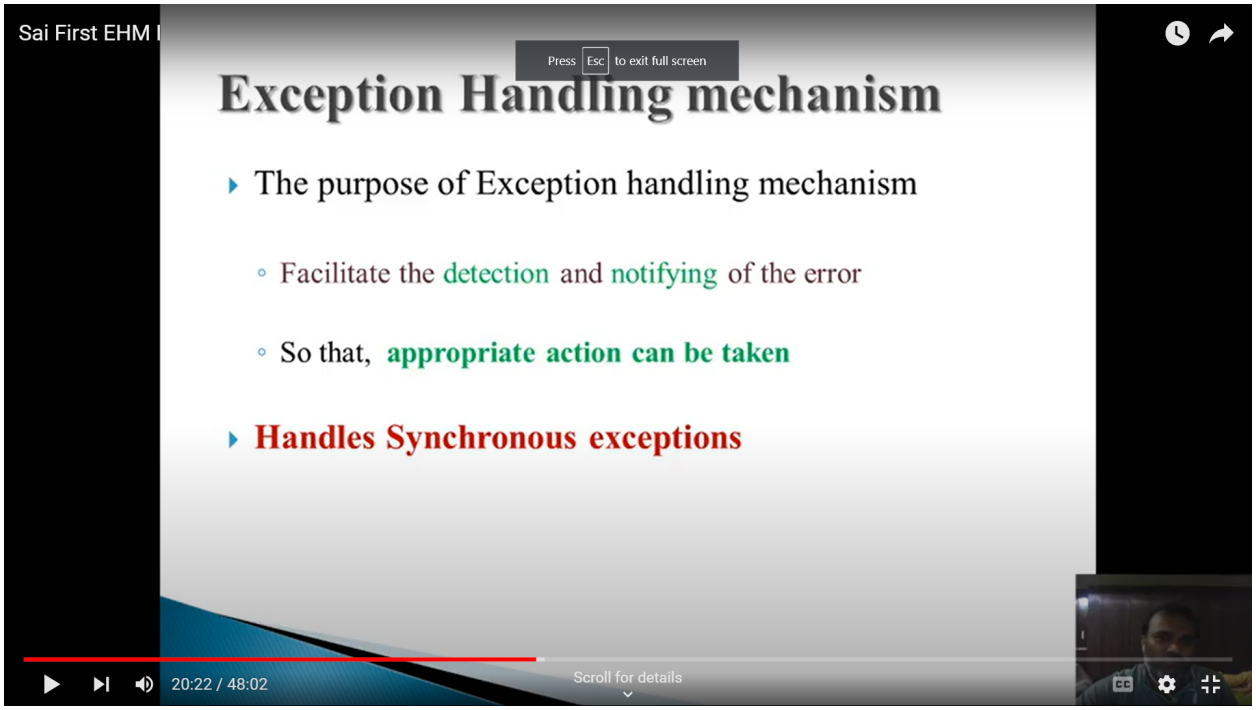

Sai First EHM
Errors
There are two common types of Bugs
Logic errors
Poor understanding of the problem and
The solution you have come up with.
Syntax errors
Due to the poor understanding of the language.
We can detect these errors
By exhaustive debugging and
Testing procedures
7:23 / 48:02
Scroll for details

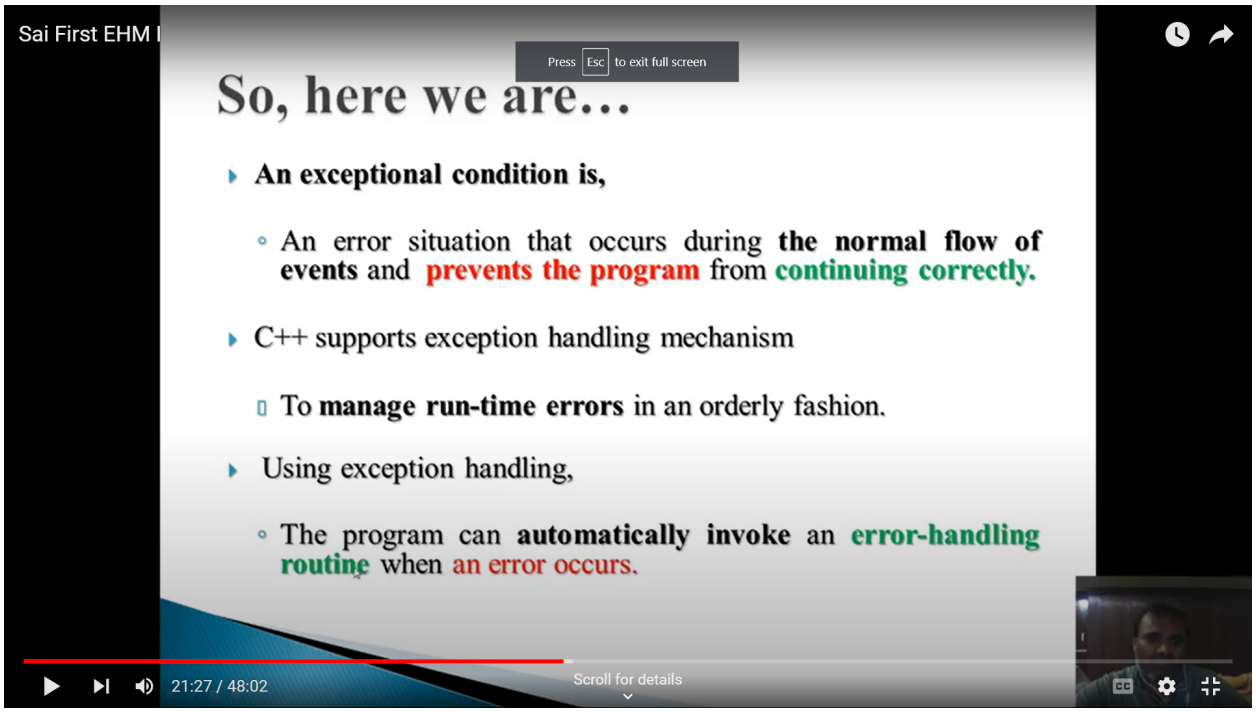

Sai First EHM I
Press Esc to exit full screen
Exceptions
There are peculiar problems other than logic and syntax errors
These are known as exceptions.
Exceptions are
– runtime anomalies or
– unusual conditions
that a program may encounter during execution.
Anomalies might include
– Division by zero
– Access to an array out of its bounds
– Running out of memory etc.
It's a good exercise to list some other anomalies.
11:08 / 48:02
Scroll for details

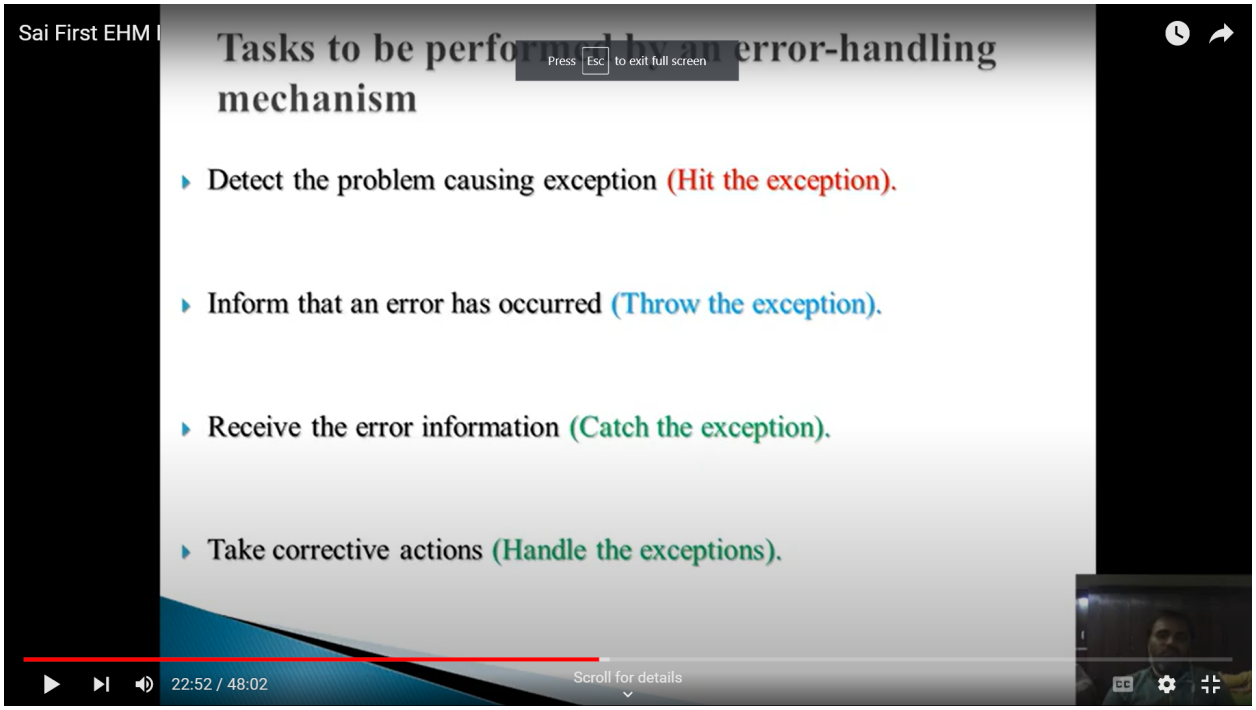

Sai First EHM I
Motivation
Press Esc to exit full screen
#include <iostream>
using namespace std;
int main()
{
int a, b, d,r;
cout<<"Enter the value of a\n";
cin>>a;
cout<<"Enter the value of b\n";
cin>>b;
d=a-b;
cout<<"I am about to divide a by d\n";
r=a/d;
cout<<"Result of dividing a by d is "<<r<<"\n";
cout<<"I am happy I am getting printed";
return 0;
}
Enter the value of a
4
Enter the value of b
2
I am about to divide a by d
Result of dividing a by d is 2
I am happy I am getting printed
Enter the value of a
2
Enter the value of b
2
I am about to divide a by d
Floating point exception
18:24 / 48:02
Scroll for details

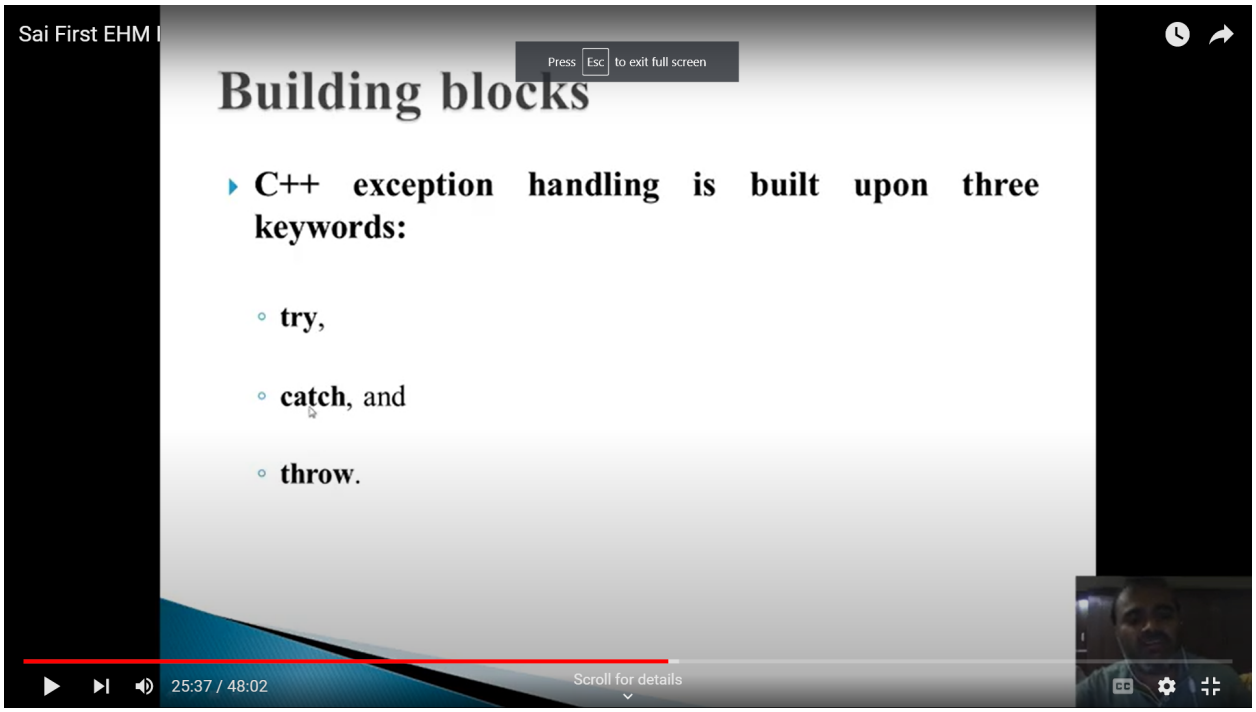

Sai First EHM I
Press Esc to exit full screen
Important
When a program encounters exceptional condition,
It should be identified and dealt with effectively.
C++ provides a mechanism to handle these kind of peculiar conditions.
19:12 / 48:02
Scroll for details

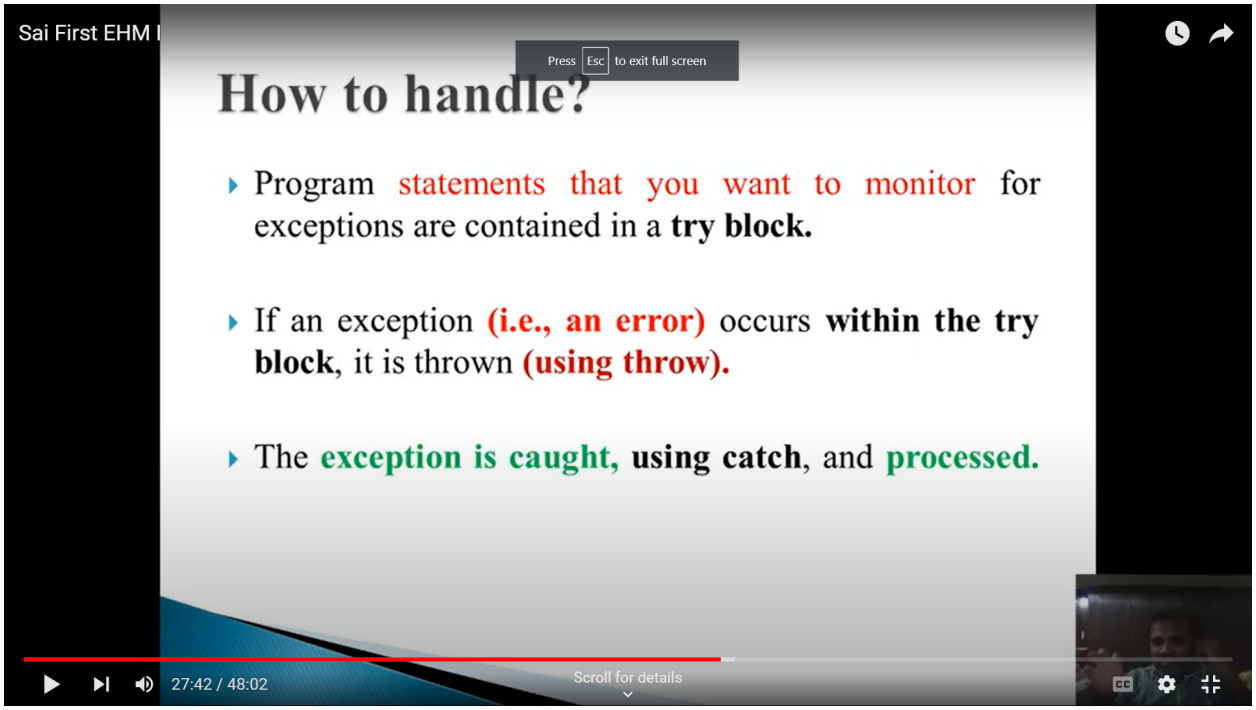

Sai First EHM I
Press Esc to exit full screen
Two kinds of Exceptions
There are two kinds of Exceptions
Synchronous Exceptions
Like, out of bound or division by zero
Asynchronous exceptions
Errors beyond the control of the program
Like, keyboard interrupts
19:32 / 48:02
Scroll for details

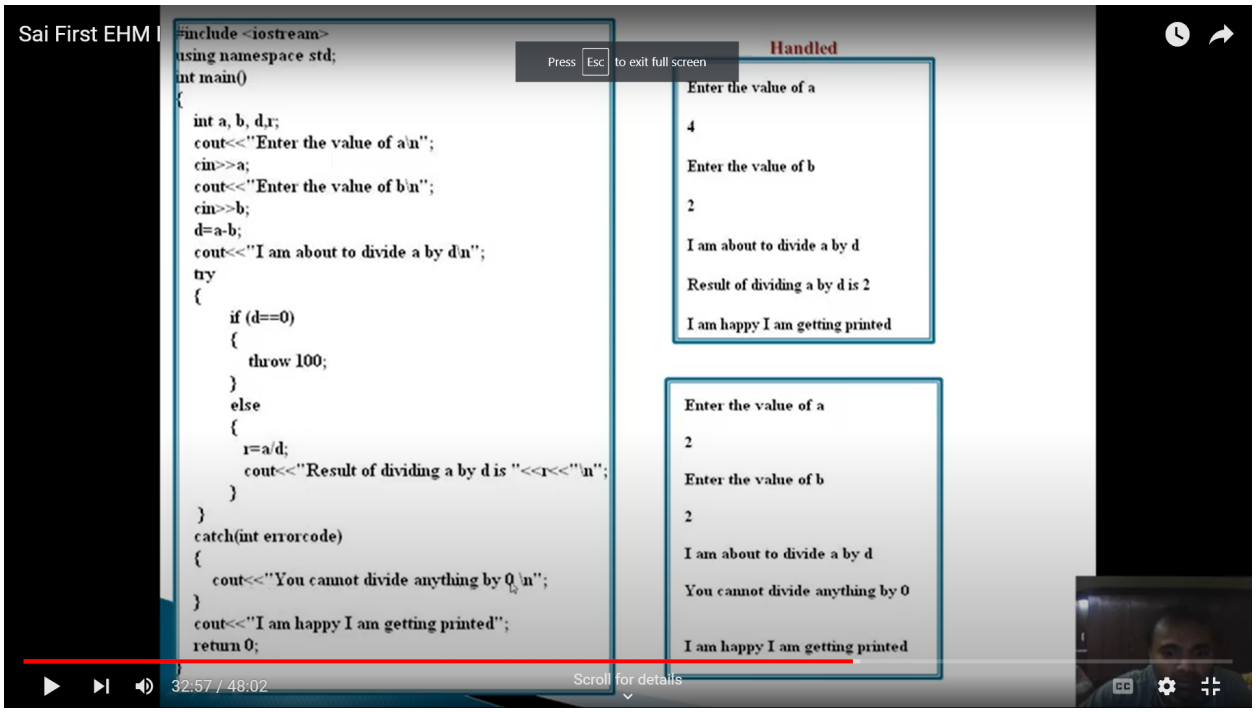

Sai First EHM I
Press Esc to exit full screen
Exception Handling mechanism
The purpose of Exception handling mechanism
Facilitate the detection and notifying of the error
So that, appropriate action can be taken
Handles Synchronous exceptions
20:22 / 48:02
Scroll for details

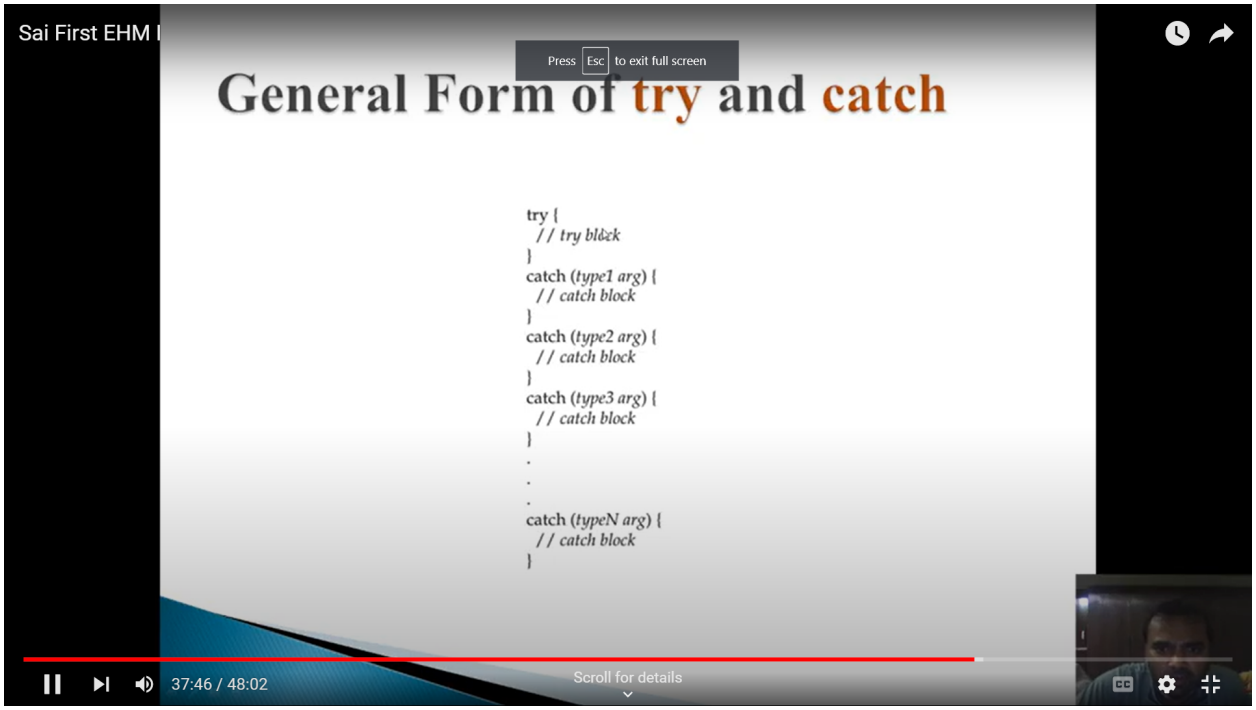

Sai First EHM I
Press Esc to exit full screen
So, here we are…
An exceptional condition is,
An error situation that occurs during the normal flow of events and prevents the program from continuing correctly.
C++ supports exception handling mechanism
To manage run-time errors in an orderly fashion.
Using exception handling,
The program can automatically invoke an error-handling routine when an error occurs.
21:27 / 48:02
Scroll for details

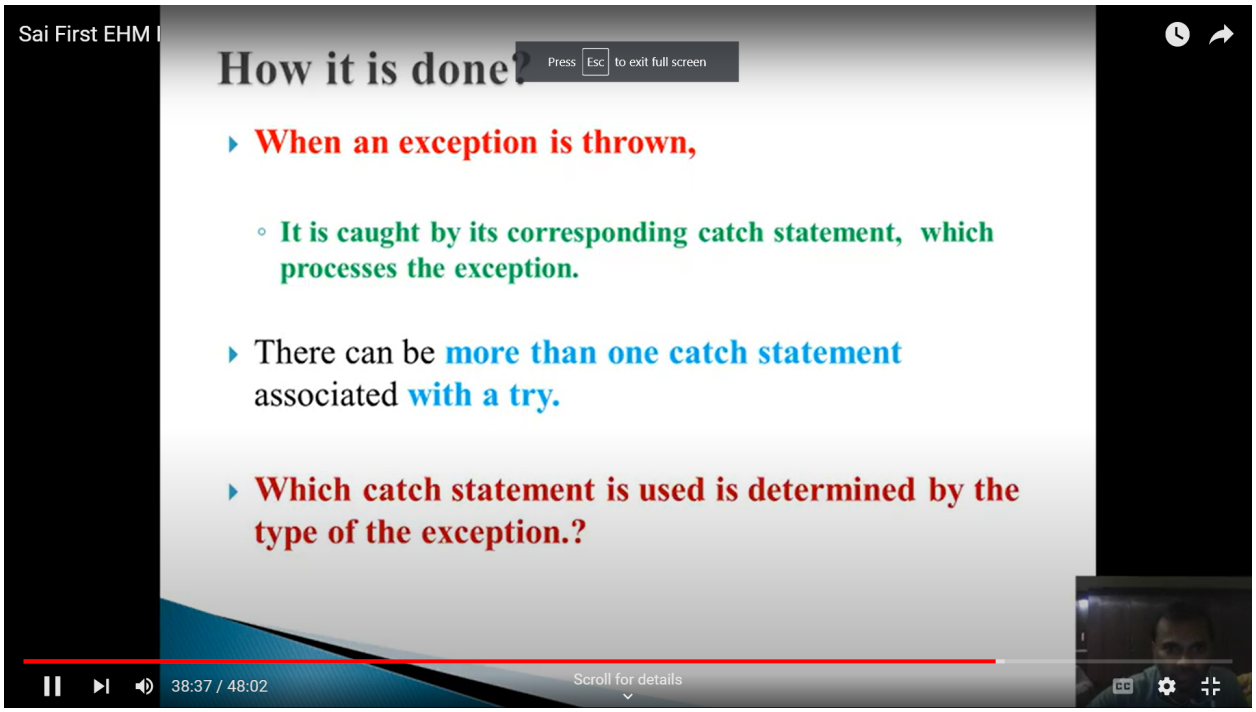

Sai First EHM I
Press Esc to exit full screen
Tasks to be performed by an error-handling mechanism
Detect the problem causing exception (Hit the exception).
Inform that an error has occurred (Throw the exception).
Receive the error information (Catch the exception).
Take corrective actions (Handle the exceptions).
22:52 / 48:02
Scroll for details

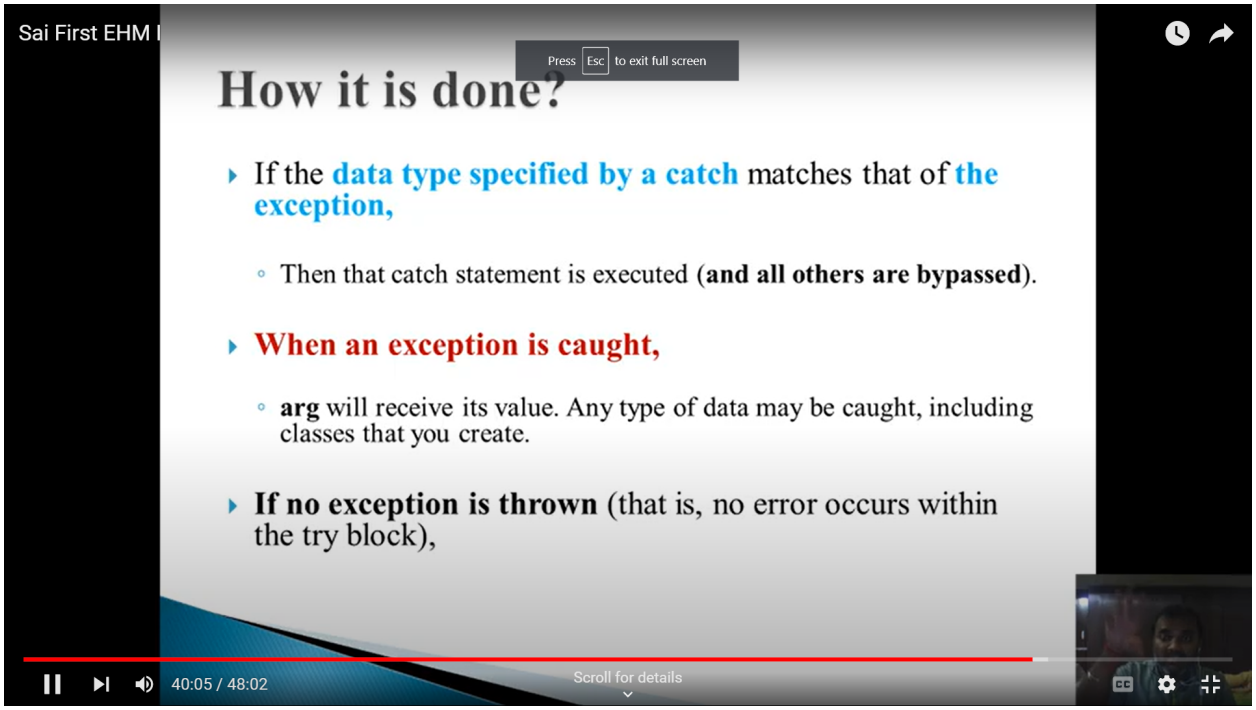

Sai First EHM I
Press Esc to exit full screen
Building blocks
C++ exception handling is built upon three keywords:
try,
catch, and
throw.
25:37 / 48:02
Scroll for details

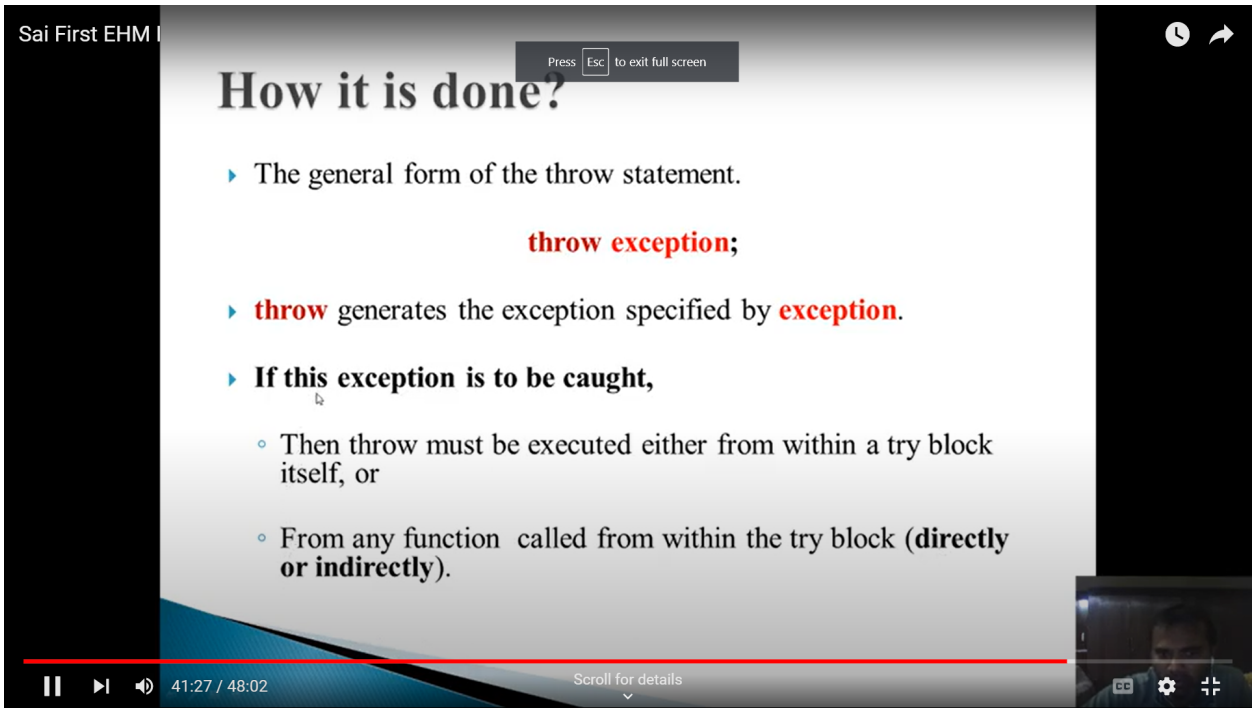

Sai First EHM I
Press Esc to exit full screen
How to handle?
Program statements that you want to monitor for exceptions are contained in a try block.
If an exception (i.e., an error) occurs within the try block, it is thrown (using throw).
The exception is caught, using catch, and processed.
27:42 / 48:02
Scroll for details

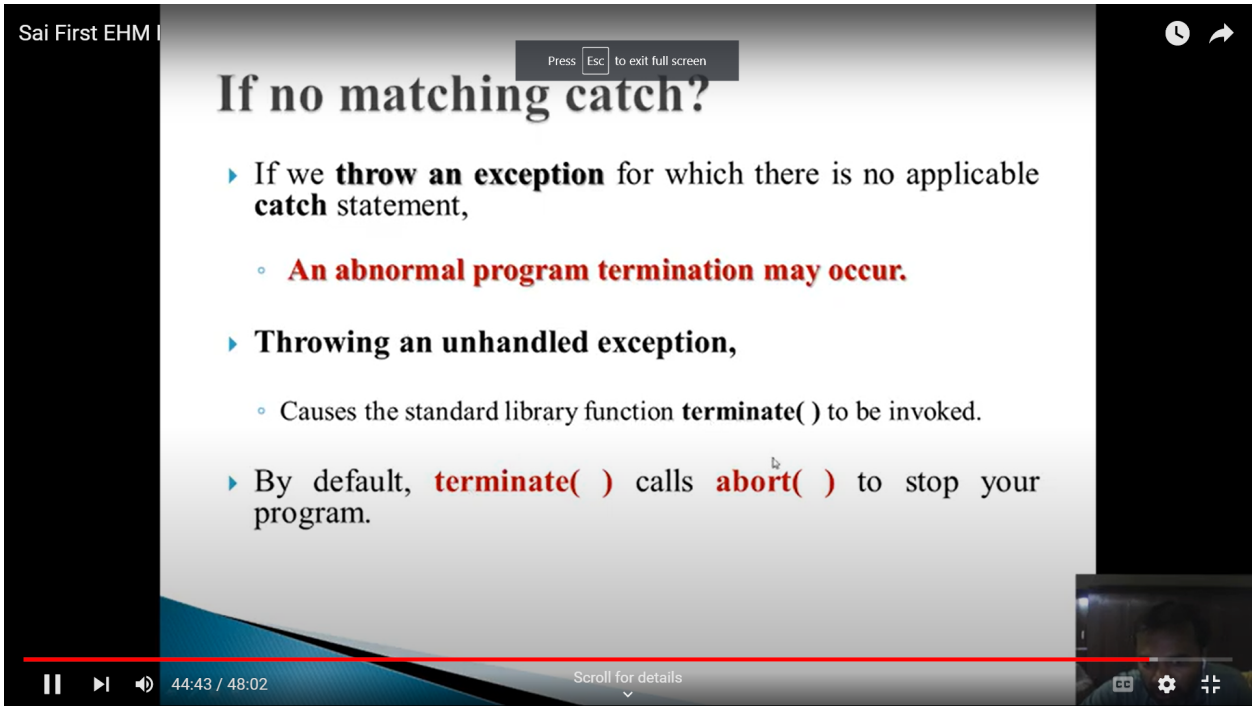

Sai First EHM I
Press Esc to exit full screen
#include <iostream>
using namespace std;
int main()
{
int a, b, d,r;
cout<<"Enter the value of a\n";
cin>>a;
cout<<"Enter the value of b\n";
cin>>b;
d=a-b;
cout<<"I am about to divide a by d\n";
try
{
if (d==0)
{
throw 100;
}
else
{
r=a/d;
cout<<"Result of dividing a by d is "<<r<<"\n";
}
}
catch(int errorcode)
{
cout<<"You cannot divide anything by 0 \n";
}
cout<<"I am happy I am getting printed";
return 0;
}
Handled
Enter the value of a
4
Enter the value of b
2
I am about to divide a by d
Result of dividing a by d is 2
I am happy I am getting printed
Enter the value of a
2
Enter the value of b
2
I am about to divide a by d
You cannot divide anything by 0
I am happy I am getting printed
32:57 / 48:02
Scroll for details

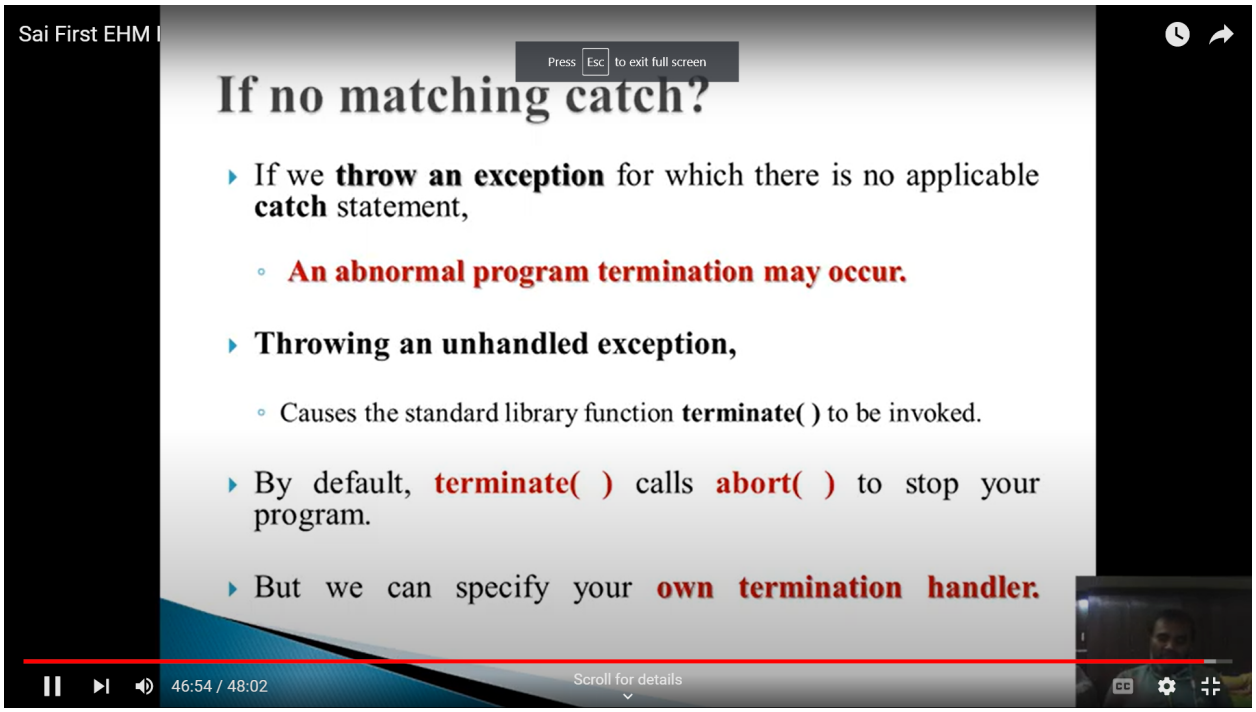

Sai First EHM I
Press Esc to exit full screen
General Form of try and catch
try {
// try block
}
catch (type1 arg) {
// catch block
}
catch (type2 arg) {
// catch block
}
catch (type3 arg) {
// catch block
}
.
.
.
catch (typeN arg) {
// catch block
}
37:46 / 48:02
Scroll for details


Sai First EHM I
Press Esc to exit full screen
How it is done?
When an exception is thrown,
It is caught by its corresponding catch statement, which processes the exception.
There can be more than one catch statement associated with a try.
Which catch statement is used is determined by the type of the exception.?
38:37 / 48:02
Scroll for details


Sai First EHM I
Press Esc to exit full screen
How it is done?
If the data type specified by a catch matches that of the exception,
Then that catch statement is executed (and all others are bypassed).
When an exception is caught,
arg will receive its value. Any type of data may be caught, including classes that you create.
If no exception is thrown (that is, no error occurs within the try block),
40:05 / 48:02
Scroll for details


Sai First EHM I
Press Esc to exit full screen
How it is done?
The general form of the throw statement.
throw exception;
throw generates the exception specified by exception.
If this exception is to be caught,
Then throw must be executed either from within a try block itself, or
From any function called from within the try block (directly or indirectly).
41:27 / 48:02
Scroll for details


Sai First EHM I
Press Esc to exit full screen
If no matching catch?
If we throw an exception for which there is no applicable catch statement,
An abnormal program termination may occur.
Throwing an unhandled exception,
Causes the standard library function terminate( ) to be invoked.
By default, terminate( ) calls abort( ) to stop your program.
44:43 / 48:02
Scroll for details


Sai First EHM I
Press Esc to exit full screen
If no matching catch?
If we throw an exception for which there is no applicable catch statement,
An abnormal program termination may occur.
Throwing an unhandled exception,
Causes the standard library function terminate( ) to be invoked.
By default, terminate( ) calls abort( ) to stop your program.
But we can specify your own termination handler.
46:54 / 48:02
Scroll for details