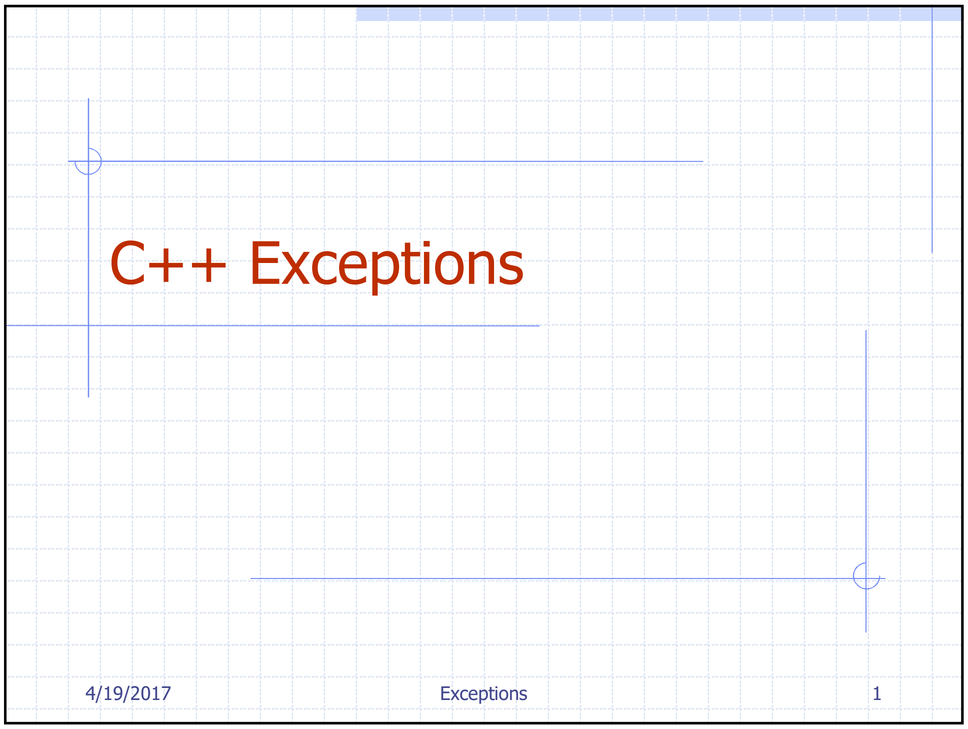
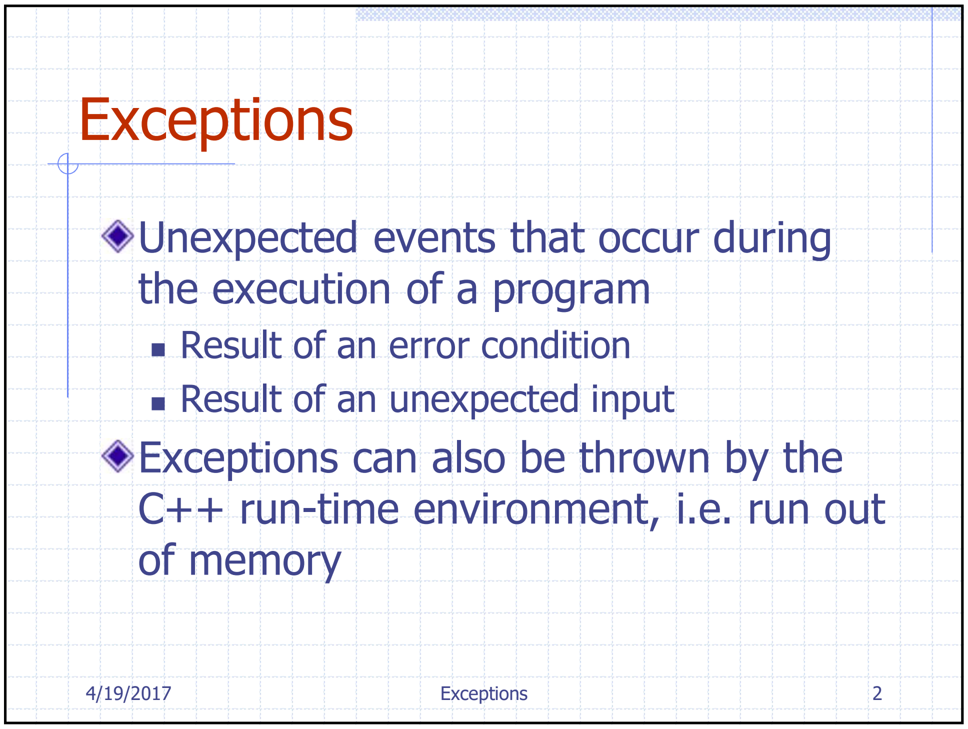
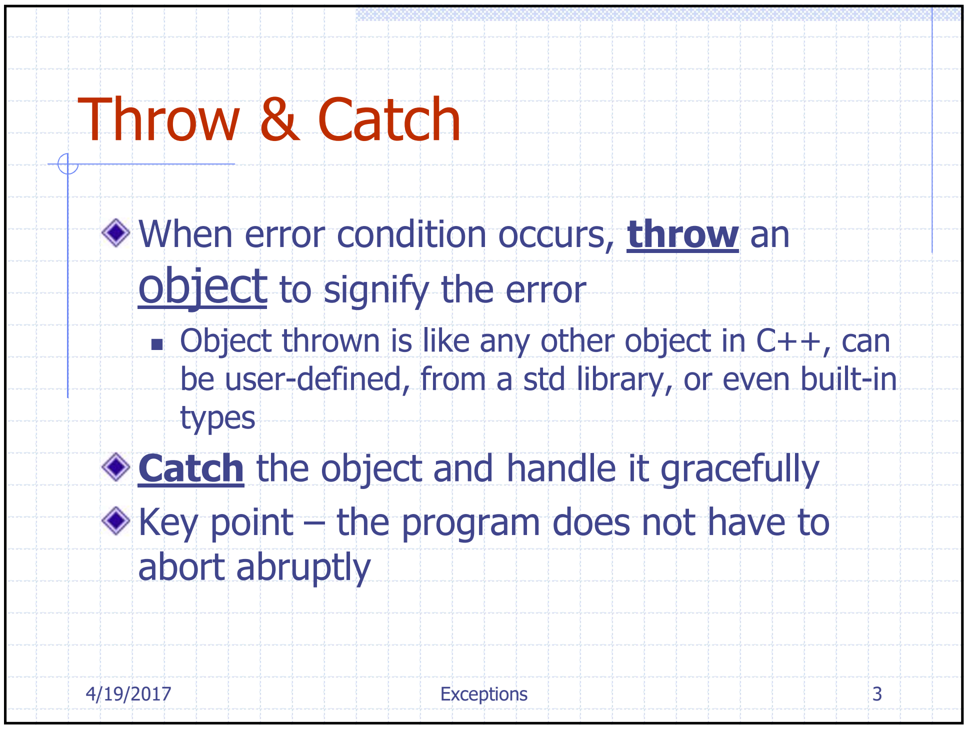


C++ Exceptions



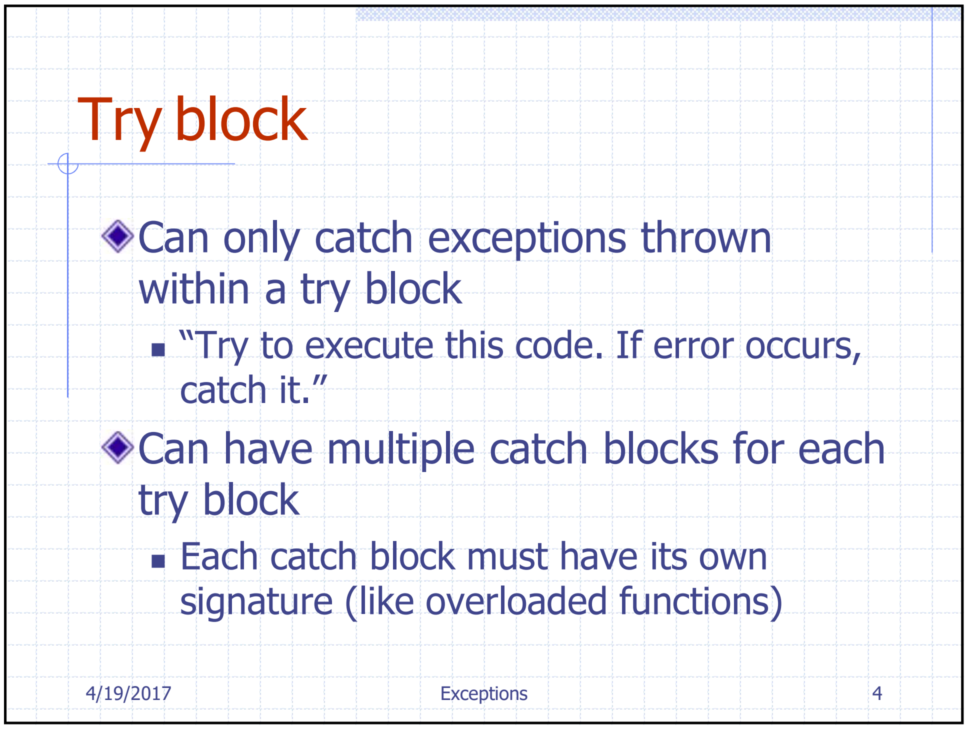
Exceptions

- Unexpected events that occur during the execution of a program
  - Result of an error condition
  - Result of an unexpected input
- Exceptions can also be thrown by the C++ run-time environment, i.e. run out of memory



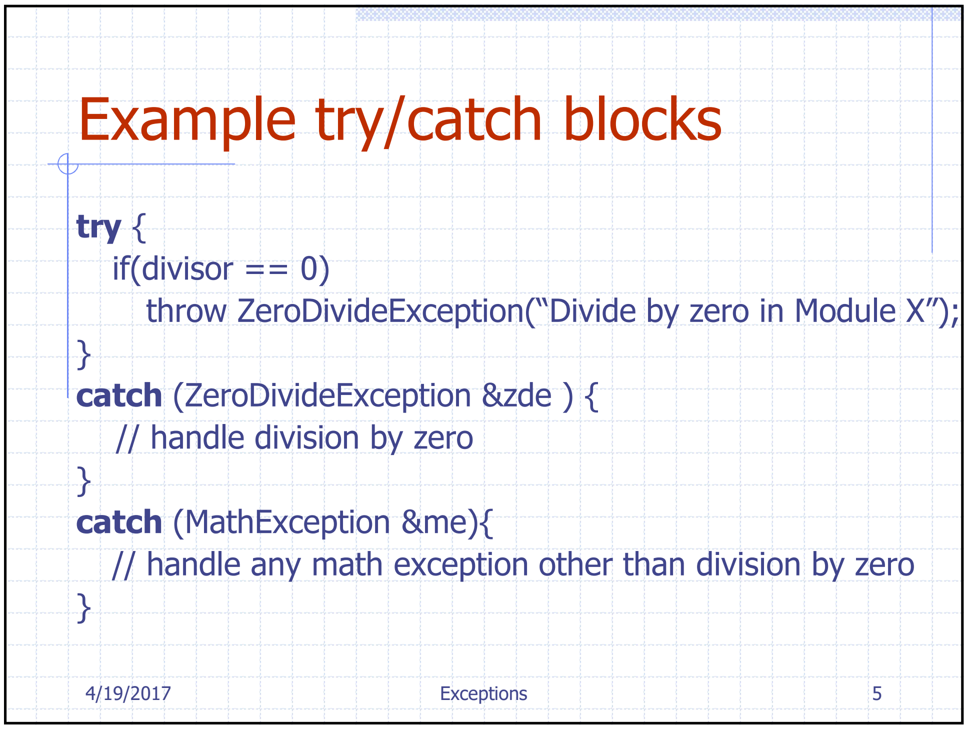
Throw & Catch

- When error condition occurs, **throw** an object to signify the error
  - Object thrown is like any other object in C++, can be user-defined, from a std library, or even built-in types
- **Catch** the object and handle it gracefully
- Key point – the program does not have to abort abruptly



Try block

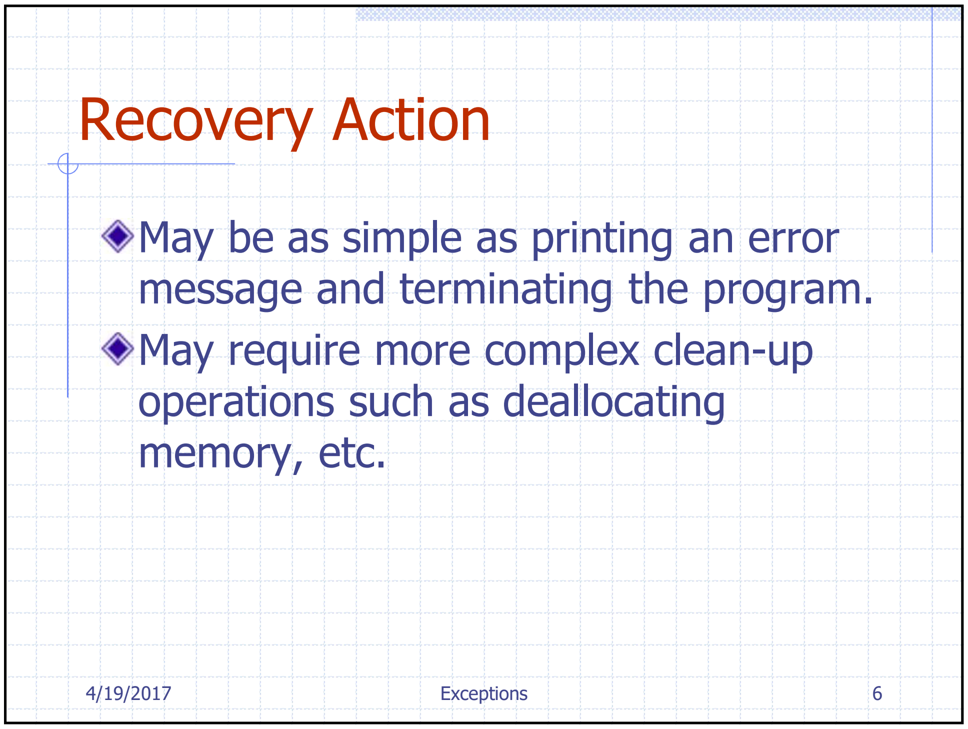
- Can only catch exceptions thrown within a try block
  - "Try to execute this code. If error occurs, catch it."
- Can have multiple catch blocks for each try block
  - Each catch block must have its own signature (like overloaded functions)



Example try/catch blocks

```
try {
   if(divisor == 0)
      throw ZeroDivideException(“Divide by zero in Module X”);
}
catch (ZeroDivideException &zde ) {
   // handle division by zero
}
catch (MathException &me){
   // handle any math exception other than division by zero
}
```

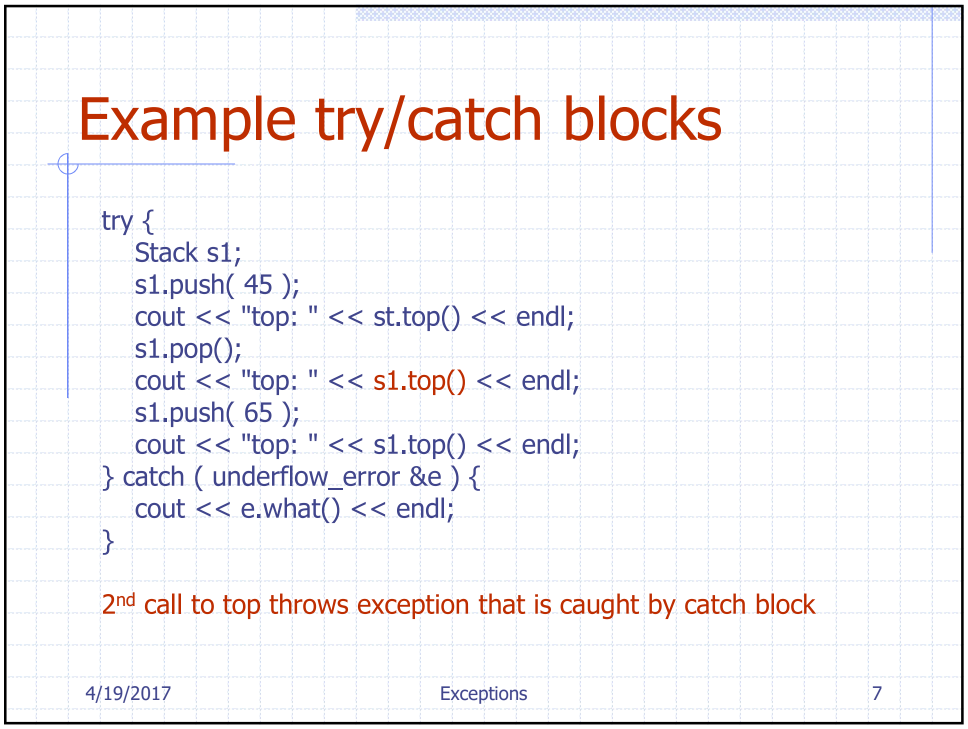
4/19/2017 Exceptions 5



Recovery Action

- May be as simple as printing an error message and terminating the program.
- May require more complex clean-up operations such as deallocating memory, etc.

4/19/2017 Exceptions 6

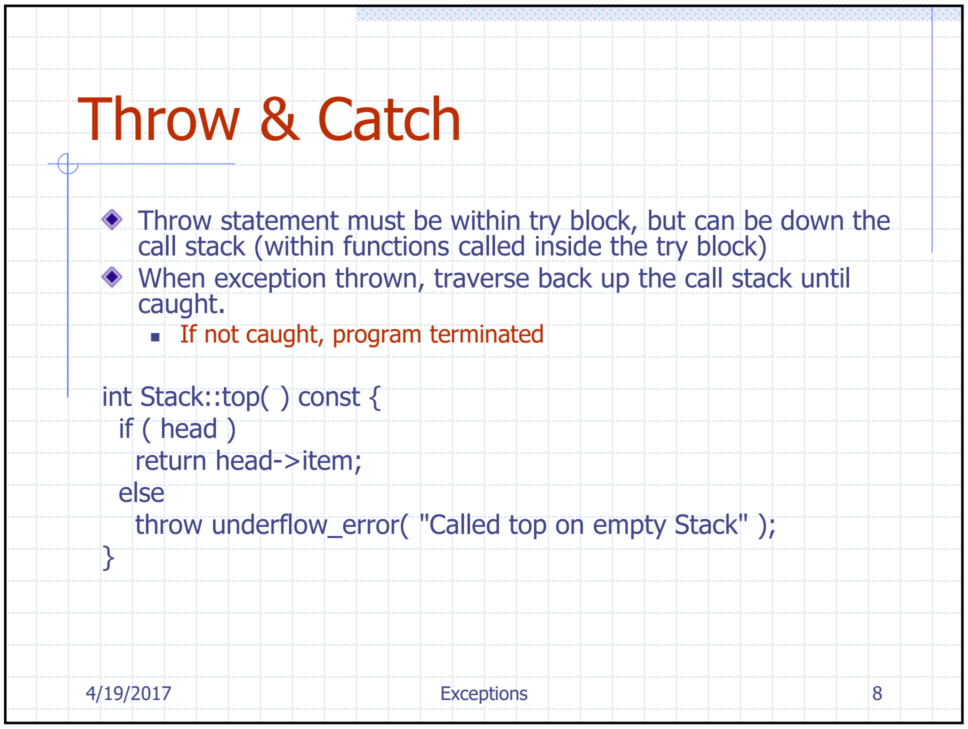


Example try/catch blocks

```
try {
    Stack s1;
    s1.push( 45 );
    cout << "top: " << st.top() << endl;
    s1.pop();
    cout << "top: " << s1.top() << endl;
    s1.push( 65 );
    cout << "top: " << s1.top() << endl;
} catch ( underflow_error &e ) {
    cout << e.what() << endl;
}
```

2nd call to top throws exception that is caught by catch block

4/19/2017 Exceptions 7

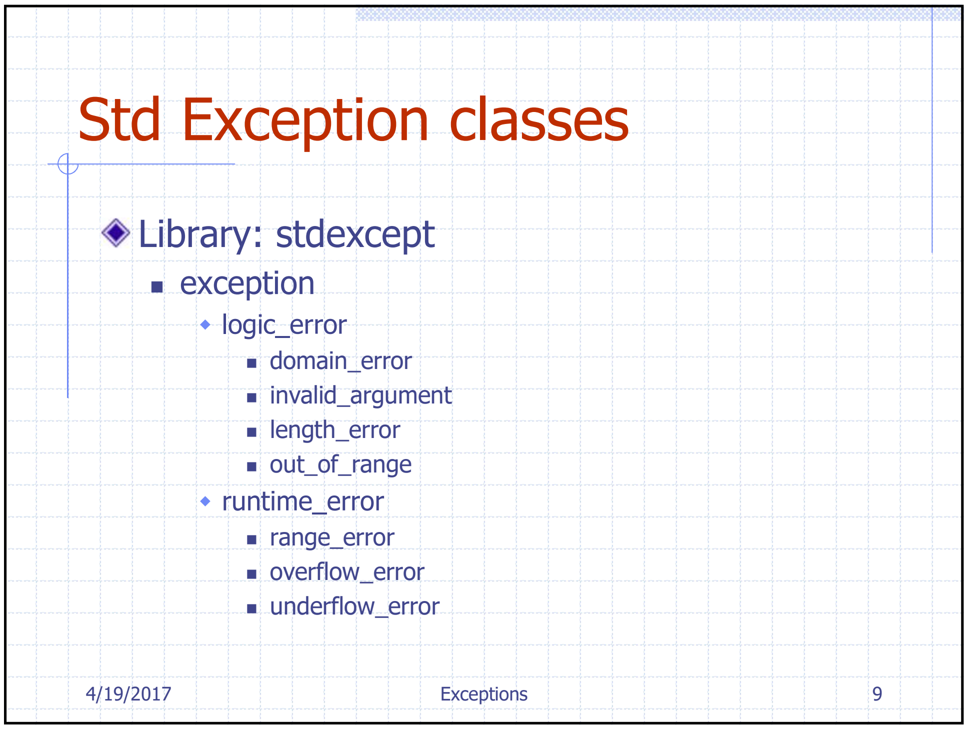


Throw & Catch

- Throw statement must be within try block, but can be down the call stack (within functions called inside the try block)
- When exception thrown, traverse back up the call stack until caught.
  - If not caught, program terminated

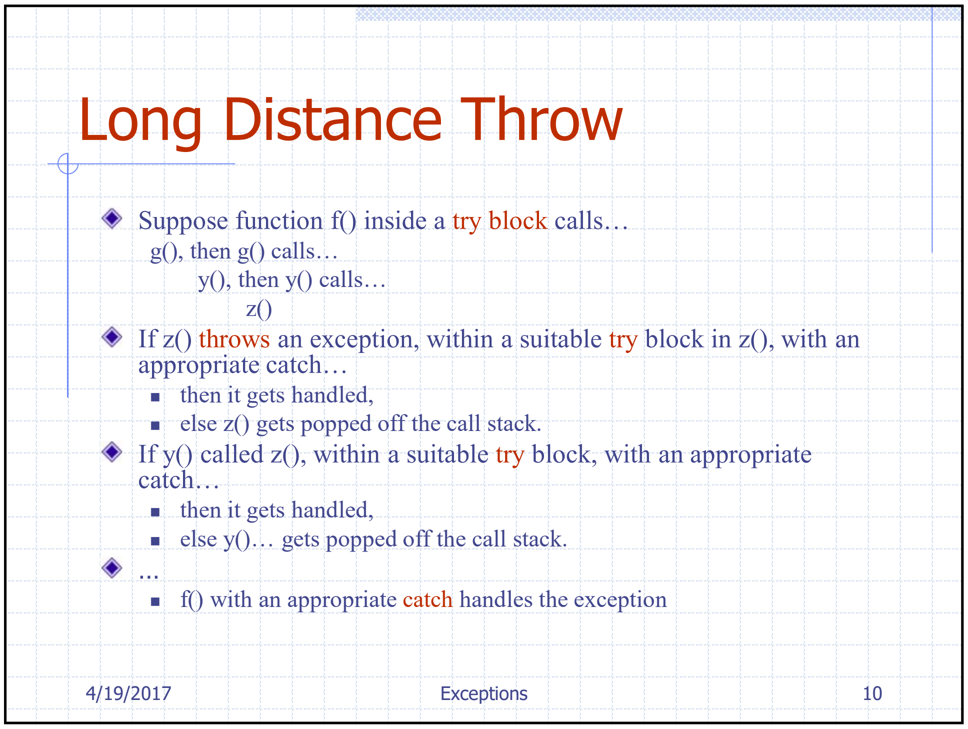
```
int Stack::top( ) const {
  if ( head )
    return head->item;
  else
    throw underflow_error( "Called top on empty Stack" );
}
```

4/19/2017 Exceptions 8



Std Exception classes

- Library: stdexcept
  - exception
    - logic_error
      - domain_error
      - invalid_argument
      - length_error
      - out_of_range
    - runtime_error
      - range_error
      - overflow_error
      - underflow_error



Long Distance Throw

- Suppose function f() inside a try block calls…

  g(), then g() calls…

  y(), then y() calls…

  z()
- If z() throws an exception, within a suitable try block in z(), with an appropriate catch…
  - then it gets handled,
  - else z() gets popped off the call stack.
- If y() called z(), within a suitable try block, with an appropriate catch…
  - then it gets handled,
  - else y()… gets popped off the call stack.
- …
  - f() with an appropriate catch handles the exception