### **Object-Oriented Programming**

### **EXCEPTIONS**

- Exceptions provide a systematic, object-oriented approach to handle runtime errors generated by C++ classes.
- To qualify as an exception, such errors must occur due to some action taken within a program and must be the ones the program itself can discover.
  - For example, a constructor in a user-written Point class might generate an
    exception if the application tries to initialize an object with coordinates that
    are beyond the limits.
  - A constructor can generate an exception if it cannot allocate memory for a dynamic class member. ptr = new Type[x]; // is ptr == nullptr?
  - o A program can check if a file was opened or written successfully and generate an exception if it was not.

http://akademi.itu.edu.tr/en/buzluca



1999 - 2024 Feza BUZLUCA

Δ11

# **Object-Oriented Programming**

## Error handling without exceptions:

- In C/C++ language programs, an error is often signaled by returning a particular value from the function in which it occurred.
- For example, many math functions return a special value to indicate an error, and disk file functions often return NULL or 0 to signal an error.
- Each time you call one of these functions, you check the return value.

http://akademi.itu.edu.tr/en/buzluci



1999 - 2024 Feza BUZLUCA

A1.2

### **Object-Oriented Programming**

# Error handling without exceptions (contd):

## Problems (without exceptions):

- The program must examine every single call to such a function.
   Surrounding each function call with an if/else statement and inserting statements to handle the error (or to call an error-handler routine) make the listing long and hard to read.
- It is not practical for some functions to return an error value.
   For example, imagine a min() function that returns the minimum of two values.
   All possible return values from this function represent valid outcomes.
- The problem becomes more complex when classes are used because errors may take place without a function (constructor) being explicitly called.

For example, suppose an application defines objects of a class:

```
SomeClass obj1, obj2, obj3;
```

There's no value left to use as an error return.

How will the application determine if an error occurred in the class constructor?

The constructor is called implicitly, so there's no return value to be checked.

http://akademi.itu.edu.tr/en/buzluca



1999 - 2024 Feza BUZLUCA

Δ13

# Object-Oriented Programming

### **Exception Syntax**

- If an error is detected in a function, it informs the application that an error has
  occurred.
- When exceptions are used, this is called throwing an exception.
- In the application, a separate section of code is installed to handle the error.
- This code is called an exception handler or catch block: it catches the
  exceptions thrown by the function.
- Any code in the application that uses objects of the class is enclosed in a try block.
- The exception mechanism uses three keywords: throw, catch, and try.

#### Throwing an exception:

Syntax of a function any Function that throws an exception:

Here exception code can be any variable or constant of any built-in type (as char, int, char \*) or it can also be an object that defines the exception.

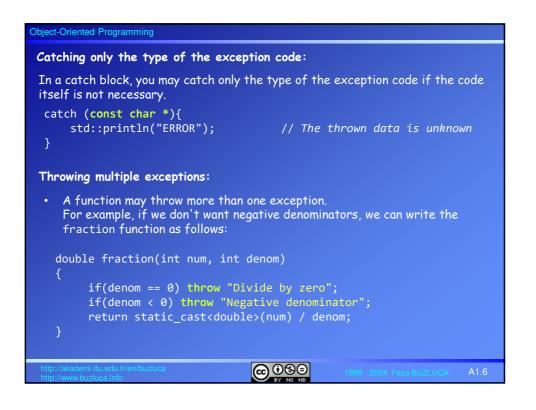
http://akademi.itu.edu.tr/en/buzluci



1999 - 2024 Feza BUZLUCA

A1.4

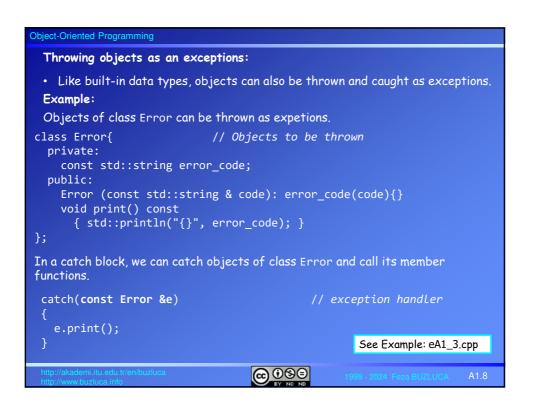
```
Object-Oriented Programming
Example:
• A fraction function: It takes the numerator and denominator as parameters.
• If the denominator is zero, an exception must be thrown.
    double fraction(int num, int denom)
       if(denom == 0) throw "Divide by zero";
                                                         // Exception condition
       return static_cast<double>(num) / denom; // Normal operation
    int main()
       int numerator, denominator;
       std::print("Enter the numerator: "); std::cin >> numerator;
std::print("Enter the denominator: "); std::cin >> denominator;
                                                                        Try block
       catch (const char* problem){
                                                          try block.
            std::println("End of Program");
                                                     See Example: eA1_1.cpp
                                       @ ⊕ ⊕ ⊕
```



```
Object-Oriented Programming
Throwing exceptions of different types:

    A function may also throw multiple exceptions of different types.

double fraction(int num, int denom)
   if(denom == 0) throw "Divide by zero";
                                                         // throws char *
   if(denom < 0) throw "Negative denominator";</pre>
                                                         // throws char *
   if(denom > 1000) throw -1;
                                                        // throws int
   return static_cast<double>(num) / denom;
 If a function throws exceptions of different types, then a separate catch
 block must be written for each exception type.
       result = fraction(numerator , denominator);
 catch (const char * problem) {
     std::println("Problem = {}", problem);
 catch (int) {
                                  // Catch block for int (value is not taken)
     std::println("ERROR");
                                                   See Example: A1_2.cpp
                                    @ ⊕ ⊕
```



```
Object-Oriented Programming
  Example (contd):
  int main()
   char input[20];
   String* str{};
   bool again;
   do{
    again = false;
    std::print(" Enter a string: "); std::cin >> input;
     str = new String{ input }; // calls the constructor to create an obj.
    catch (const char * error){
                                      The only way to exit the do-while loop is to
     std::println("{}", error);
                                      provide strings shorter than 10
     again = true;
                                      characters. Otherwise, the object is not
                                      created.
   }while(again);
   str->print();
   delete str;
   return 0;
                                                       See Example: eA1_4.cpp
                                     @ ⊕ ⊕
```