Cairo University
Faculty of Computers and Information
Computer Science Department

Programming-2 CS213
2018/2019

Exceptions

Dr. Amin Allam

1 Exceptions

Exceptions allow handling run-time errors in an orderly fashion. They allow the separation of the error handling code from the rest of the program. Consider the following example:

```
1  class CantOpenFile {};
2  class NotEnoughSpace {};
3
4  void WorkOnFile()
5  {
6      try {
7          if(!OpenFile()) throw CantOpenFile();
8          if(!WriteOnFile()) throw NotEnoughSpace();
9          if(!CloseFile()) throw 11.4;
10     }
11     catch(CantOpenFile)
12     {
13         cout<<"Can not open file."<<endl;
14     }
15     catch(double d)
16     {
17         cout<<"double thrown with value " <<d<<endl;
18     }
19     cout<<"End of function."<<endl;
20 }
```

There are 3 keywords associated with exceptions: `try`, `catch`, and `throw`. `try` is followed by a block of statement. After that block ends, a sequence of `catch` blocks may exist, each `catch` is followed by a data type and possibly an object name (inside parentheses), then followed by a block of statements. `throw` takes an object to its right and may appear inside a `try` block. If `throw` takes a value to its right, it throws an object containing this value.

Whenever `throw` is executed, the subsequent code is skipped until the closing bracket of the innermost `try` block containing the `throw` statement, and then the `catch` blocks associated with that `try` block are checked in order. When the data type of the `catch` matches the thrown object (or its base class), its block is executed, then the control is passed after all `catch` blocks associated with that `try`. If no matching `catch` is found, the subsequent code is skipped until the closing bracket of the immediately outer `try` and its `catch` blocks are checked.

In the above example, if `OpenFile()` returns false, `throw CantOpenFile();` is called, which throws an unnamed object of type `CantOpenFile`. The remainder of the `try` block is skipped, then the block associated with `catch(CantOpenFile)` is executed, which outputs `"Can not open file."`, then all other `catch` blocks are skipped and the program outputs `"End of function."`.

If `WriteOnFile()` returns false, `throw NotEnoughSpace();` is called, which throws an unnamed object of type `NotEnoughSpace`. The remainder of the `try` block is skipped, then the program terminates because no matching `catch` is found (the program does *not* output `"End of function."`).

If `CloseFile()` returns false, `throw 11.4;` is called, which throws an unnamed object of type `double` that contains the value `11.4`. The remainder of the `try` block is skipped, then the block associated with `catch(double d)` is executed, which outputs `"double thrown with value 11.4"`, then the program outputs `"End of function."`.

Consider another example which starts by calling `WorkOnFile()`:

```
1  class FileException {};
2  class CantGetName : public FileException {};
3  class CantFindFile : public FileException {};
4  class CantOpenFile : public FileException {};
5
6  class NotEnoughSpace
7  {
8  public:
9      int id; string str;
10     NotEnoughSpace(int i, string s) : id(i), str(s) {}
11 };
12
13 void OpenFile()
14 {
15     try {
16         if(!GetFileName()) throw CantGetName();
17         if(!FindFileWithName()) throw CantFindFile();
18         if(!OpenFileWithName()) throw CantOpenFile();
19     }
20     catch(CantGetName) {cout<<"Can not get name."<<endl;}
21     catch(CantFindFile) {cout<<"Not found."<<endl; throw;}
22 }
23 void WorkOnFile()
24 {
25     try {
26         OpenFile();
27         if(!WriteOnFile()) throw NotEnoughSpace(10, "NES");
28         if(!CloseFile()) throw 11;
29     }
```

```
30      catch(CantFindFile) {cout<<"Can not find file."<<endl;}
31      catch(FileException) {cout<<"File exception."<<endl;}
32      catch(CantOpenFile) {cout<<"Can not open file."<<endl;}
33      catch(NotEnoughSpace n) {cout<<n.s<<" "<<n.id<<endl;}
34      catch(double d) {cout<<"double thrown " <<d<<endl;}
35      catch(...) {cout<<"Exception occurred."<<endl;}
36      cout<<"End of function."<<endl;
37  }
```

In the above example, if `GetFileName()` returns false, `throw CantGetName();` is called. The remainder of the `try` block is skipped, then the block associated with `catch(CantGetName)` is executed, then function `OpenFile()` returns normally and the code continues from line 27.

If `FindFileWithName()` returns false, `throw CantFindFile();` is called. The remainder of the `try` block is skipped, then the block associated with `catch(CantFindFile)` is executed, the `throw;` statement rethrows the exception outside the function, to be handled by the `try` block from within `OpenFile()` is called, then the block associated with `catch(CantFindFile)` in the function `WorkOnFile()` is executed, then the program outputs `"End of function."`.

If `OpenFileWithName()` returns false, `throw CantOpenFile();` is called. Since there is no block associated with innermost `try` with the type `CantOpenFile` is found, the exception is thrown outside the function to be handled by the `try` block from within `OpenFile()` is called. Then the block associated with `catch(FileException)` in the function `WorkOnFile()` is executed because `class CantOpenFile` is inherited from `class FileException`. The block associated with `catch(CantOpenFile)` is ignored, and finally the program outputs `"End of function."`.

If `WriteOnFile()` returns false, `throw NotEnoughSpace(10, "NES");` is called. The remainder of the `try` block is skipped, then the block associated with `catch(NotEnoughSpace n)` is executed and outputs `"NES 10"`, then the program outputs `"End of function."`.

If `CloseFile()` returns false, `throw 11;` is called, which throws an unnamed object of type `int`. No matching `catch(int)` is found and the block associated with `catch(...)` is executed since this block can catch any exception, then the program outputs `"End of function."`.

It is possible to restrict the exceptions thrown outside a function. The following function can throw only objects of type `int`, `CantGetName`, `NotEnoughSpace` outside the function:

```
1   void OpenFile() throw (int, CantFindFile, NotEnoughSpace)
2   {
3       try {
4           if(!GetFileName()) throw CantGetName(); ✔ // caught inside
5           if(!FindFileWithName()) throw CantFindFile(); ✔ // listed
6           if(!OpenFileWithName()) throw 17.2; ✖ // double not listed
7           if(!OtherProblem()) throw UnknownProblem(); ✖ // not listed
8       }
9       catch(CantGetName) {cout<<"Can not get name."<<endl;}
10  }
```