Lecture Notes 18

Exception Handling

- Exception – Any unusual event that can be detected by hardware or software
- Exception Handling – Processing required when an exception is detected
- Exception Handler – The body of code responsible for handling exception
- Raising (or Throwing) Exception – The detection and alerting that an exception has occurred
- Implicit control transfer – Transfer is setup ahead of time but not explicitly stated
- Resumption model – Resume execution from the code that raised exception (once handled)
- Termination model – Continue execution from immediately after handler
- Finalization – Ability to specify computation to occur after subprogram terminates
- Propagating the exception – Moving outward of nested blocks to find appropriate handler
- Call unwinding (Stack unwinding) – Propagating exception out from procedure calls
- `try`-`catch` block – Try block of code, catch is exception handler (used in C++, and Java)
- `throw` – Keyword in C++ and Java to raise or throw an exception
- `finally` – Keyword in Java for specifying finalization
- Unchecked Exceptions – In Java exceptions that are descendants of `Error` or `RuntimeException` that are unchecked by the compiler
- Checked Exceptions – All exceptions except those that are unchecked
- `throws` – Java keyword that specifies which exceptions the method can `throw`
- Assertion – A condition that must be true in order for code to continue
- `assert` – Java keyword to specify an assertion
- `try`-`except`-`else`-`finally` block – Python equivalent to the try catch block in Java, but `else` is done when no exception is raised
- `raise` – Python keyword for raise an exception similar to `throw` in C++ and Java
- `rescue` clause – Exception handler in Ruby
- `else` clause – Ruby clause that behaves just like Python
- `ensure` clause – Ruby clause that is equivalent to finally
- `retry` statement – Ruby statement that will retry the code that raised an exception after the event has been handled