ECET 3810:W12 Exceptions

- Java exception keywords: *throws, throw, try, catch, and finally*

Ex.

```
try
{
      int number = Integer.ParseInt(stringField.getText());
      resultField.setText(“Doubled value is “+(2*number));
}
catch (NumberFormatException nfe)
{
                    resultField.setText(“Error in number:“ + nfe.toString() );
}
```

- “when a try-block produces an exception, its execution terminates – it is abondoned. A consequence of this is that any variables declared within it become inaccessible and, in particular, they cannot be used in the catch-block, even when it is in the same method.”
- “The throw statement is often executed as the result of an *if*: it causes the current method to be abandoned, and starts a search for a matching *catch*.”

Ex.

```
public static int ParseInt(String s)
                    throws NumberFormatException {
      …code for the method…
throw new NumberFormatException();
      …etc
}
```

Exceptions - continued

- Java Exception Inheritance Class ~ approximately 30 total
- *checked* means that you must deal with them or your program will not compile

- Throwable
  - Error (unchecked)
    - LinkageError
    - VirtualMachineError
      - OutofMemoryError
  - Exception (all, checked, except for RuntimeException)
    - RunTimeException (unchecked)
      - ArithmeticException
      - IndexOutofBoundException
        - ArrayIndexOutofBoundsException
      - IllegalArgumentException
        - NumberFormatException
    - IOException (checked)
      - FileNotFoundException
      - MalformedURLException
    - InterruptedException (checked)
    - Other classes of Exception, all checked

Exceptions - continued

- Three cases of using *throw*
  - Thrown exception must be acknowledged, i.e. must use try-catch
  - Programers choice wether or not to use try-catch. If ignored by the programmer, the exception gets passed up to the method that invoked the current method, or to a superior class, and so on until the first appropriate catch is found. If no catch is found, the Java will display an exception message.

    If, however, your program uses keyboard I/O, then its execution will terminate.
  - Throwing exceptions which are not stated in a method's header; e.g. arrayIndexOutofBoundsException thrown when command line argument count inconsistent with args
- One can try all exceptions of a class
  - Ex.

```
try {
        //some code
}
catch (Exception e) {
        //handle it
}
```

- *finally* – "the last chance part of an exception handler." Used to ensure that a final action is applied to an exiting method. For instance, a file-handling method may be terminated prematurely, thus leaving the file open.In this case, you would use a finally-block after the last catch block,