Error Handling

Chapter 7 – Clean Code

by Michael Feathers

Error Handling

- Error handling is important, but if it obscures logic, it’s wrong.

- Many code bases are dominated by error handling.
- 80/20 rule: 80% of the time is spent in 20% of the code.
  - What is in that other 80% of the code?
    - Some of it is seldom used functions.
    - The rest is often Error Handling code

- Often it is nearly impossible to see what code does because of scattered error handling.

Error Handling

- Use Exceptions Rather than Return Codes
  - Pp. 104,105 shows example

```
int findPos (data[] arr, data val)
{
   if ( array == null )  return -3;
   if (array.length == 0 ) return -2;
   // find pos of val and return pos
   …
   // if not found
   return -1;
}
```

Error Handling

- Use Exceptions Rather than Return Codes

```
int pos = findPos ( arr1, val1)
if ( pos == -3 ) {
  // process error recovery
}
else if ( pos == -2 ) {
  // process error recovery
}
else if ( pos == -1 ) {
  // process error recovery
}
else {
  // Normal execution
}
```

Error Handling

- Write you Try-Catch-Finally Statement First
  - Forces the Programmer to “Define the Scope” for error handling earlier
  - Keeps the programmer from adding “extra” try scopes
  - As additional exceptions are used in the code, additional catch clauses are added

Error Handling

- Martin says: “Use Unchecked Exceptions” when appropriate
- The Java programming language does not require methods to catch or to specify unchecked exceptions (RuntimeException, Error, and their subclasses)
- Controversy Described at: https://docs.oracle.com/javase/tutorial/essential/exceptions/runtime.html

Error Handling

- Define the Normal Flow
  - Keep Normal Flow code separate from Error Handling
- Don’t Return Null
  - Throw an exception instead
- Don’t Pass Null
  - Should code automatically throw NullPointerException?
  - Should you throw InvalidArgumentException?
  - Use asserts?