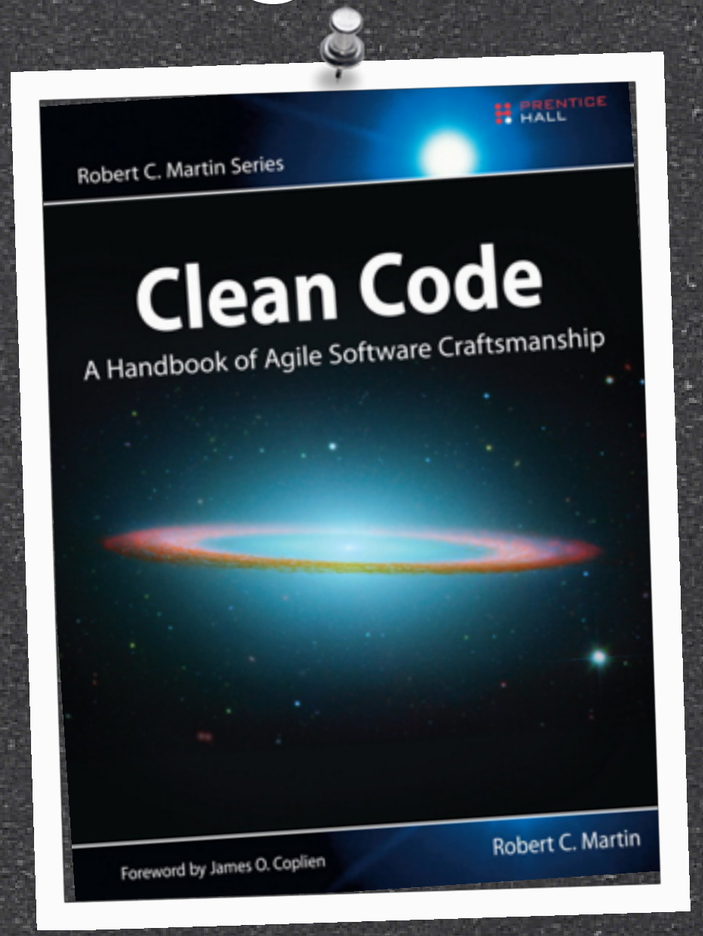
Clean Code
“Error Handling”

```
try {
  breakStuff();
  doSomeDamage();
  burnTheWorld();
} catch {
  makeItRight();
} finally {
  eitherWay();
}
```



It’s your job. Do it.

- Error handling isn't a curse, or a bother.
- Things can and do go wrong. Thus, error handling is an essential part of programming.
- And, since it involves code, it ought to be clean.

Keep it clean.

- Don’t obscure logic.
- Don’t hide the point of the code.
- Don’t duplicate.
- Don’t forget to refactor.

Try. Catch. Finally.

- try: things can go sideways; nothing has to worry about what came before
- catch: put things back into a consistent state
- finally: things to do, either way

Tabs or Spaces? Err… Checked or Unchecked?

- People do still debate this. Anyway…

Tabs or Spaces? Err… Checked or Unchecked?

- People do still debate this. Anyway…

Tabs or Spaces? Err… Checked or Unchecked?

- People do still debate this. Anyway…

Best Case		
Worst Case		

Tabs or Spaces? Err… Checked or Unchecked?

- People do still debate this. Anyway…

	Checked	
Best Case	coupling	
Worst Case	• desirable violation of encapsulation • violates open/closed • difficulty versioning	

Tabs or Spaces? Err… Checked or Unchecked?

- People do still debate this. Anyway…

	Checked	Unchecked
Best Case	coupling	no coupling
Worst Case	• desirable violation of encapsulation • violates open/closed • difficulty versioning	desirable violation of encapsulation

Don’t do this.

```
public int getStatus(String messengerID)
{
  try
  {
     return messengerMap.get(messengerID).getStatus();
  }
  catch(Exception e)
  {
     return 7;
  }
}
```

Don’t do this.

```
if (windSensorObject.getSensorAlarms() != null &&
        windSensorObject.getSensorAlarms().isEmpty() == false) {
    // Is Alarm
    this.windGroupStatus = "ALARM";
} else if (windSensorObject.getSensorWarnings() != null &&
        windSensorObject.getSensorWarnings().isEmpty() == false) {
    // Is Warning
    alarms.setIterateWindAlarm();
    this.windGroupStatus = alarms.getWindStatusIsALARM(amcsAlarmMessageWaitTime);
} else if (windAlgSensorObject.reading("ws_sensor_status") != null &&
        windAlgSensorObject.reading("wd_sensor_status") != null) {
    ...
} else {
    // Is not OK
    alarms.setIterateWindAlarm();
    this.windGroupStatus = alarms.getWindStatusIsALARM(amcsAlarmMessageWaitTime);
}
```

Don’t do this.

```
try {
      ...
      } else if (connType.equalsIgnoreCase("SERIAL")) {
        if (outputStream != null) {
            outputStream.close();
            this.outputStream = null;
        }
        if (inputStream != null) {
            inputStream.close();
            this.inputStream = null;
        }
        if (serialPort != null) {
            serialPort.close();
            this.serialPort = null;
        }
        //logger.info("Successfully closed serial port.",this.getClass().getName());
      }
      isConnected = 0;
      this.physicalConnection = 0;
} catch (IOException e) {
      //logger.error(new MessengerException(0,e,null,this.getClass().getName()));
      // TODO either remove logging completely, and comment out or comment out.
      //System.err.println("ERROR DISCONNECTING SOCKET:" + e.getMessage());
      //e.printStackTrace();
      isConnected = 2;
}
```

Don’t do this.

```
                } else {
                    logger.error(new MessengerException(0, null,
                    "Message exceeded maximum character length. Not sent: " + data,
                    null, this.getClass().getName()));
                }
            } else {
                logger.error(new MessengerException(0, "Serial Port not instanciated: " + data, null, this.getClass().getName()));
                this.physicalConnection = 0;
            }
        } catch (Exception e2) {
            logger.error(new MessengerException(0, e2, null, this.getClass().getName()));
            isConnected = 2;
            this.physicalConnection = 0;
        }
    } else {
        logger.error(new MessengerException(0, null, "Unrecognized. Cannot send data: “ + data, null, this.getClass().getName()));
        isConnected = 2;
        this.physicalConnection = 0;
    }
} catch (Exception e) {
    logger.error(new MessengerException(0, e, null, this.getClass().getName()));
    isConnected = 2;
    this.physicalConnection = 0;
}
```