

Offset	Topic
	<ul style="list-style-type: none"> ● Intro ● Quick review of No More Heroes <ul style="list-style-type: none"> ● http://en.wikipedia.org/wiki/No_More_Heroes_(video_game) ● Not for kids ● Some of the humor, gore a bit over the top ● Has a certain charm, like Samurai Champloo ● Use of motion sensors a bit disappointing ● A solid button mashing fighter, though ● Very silly special mode, with English mangling of desert names ● Riding through the city loses its appeal, a quick jump mode would be nice ● Load times are also sometimes a bit intrusive ● I like it but it won't appeal to a lot of folks
	<ul style="list-style-type: none"> ● Word of the Week: crippleware ● http://catb.org/jargon/html/C/crippleware.html
	<ul style="list-style-type: none"> ● Hacking 101: Aspect Oriented Programming ● What is an aspect? <ul style="list-style-type: none"> ● http://en.wikipedia.org/wiki/Aspect_oriented_programming ● Originally developed by Gregory Kiczales at Xerox PARC ● An attempt to improve separation of concern ● Encapsulation in OO is also about separating concerns ● Many principles attempt to improve separation <ul style="list-style-type: none"> ● Do not repeat yourself ● Single responsibility ● Open, closed ● Separation means components overlap as little as possible ● The term loosely coupled or decoupled also applies ● AOP introduces a cross cutting concern as a new form of encapsulation ● Called that because it address some issue that applies across a mix of other types ● Where did AOP come from <ul style="list-style-type: none"> ● Logical extension of duck typing ● Close to categories in Smalltalk ● Code independent of a class' state ● Category, aspect operates across all types, fulfilling some independent concern ● Most common/popular implementation, AspectJ <ul style="list-style-type: none"> ● Bytecode re-engineering of Java

Offset

Topic

- Introduces lower level semantics
- An aspect is the encapsulation of a cross cutting concerns
- Join points are where they apply to existing code
- Advice is additional behavior as applied from the aspect to the existing object
- Point cut is a query, a way of specify where to inject a join point
- Join points seem more generalizable
- A common example of a join point is a method entrance/exit or a member read/write
- Limitations
 - Common specific examples logging, security, transactions
 - Some of these cannot be separated all the time
 - How can a generic piece of code always know where to start a transaction?
 - Requires more complex cut points or an overall more remedial design
 - These examples also often have to do with tooling, not application intelligence
 - Think AOP is a good approach for a container, services
 - Doesn't make as much sense for business rules, core application logic
 - Proponents argue removal of cross-cutting concerns increases readability
 - I am concerned about added cost of debugging
 - Surprises in behavior because of non-local concerns
 - At some point, concerns have to become at least minimally tangled
 - Otherwise, perfectly de-coupled components never interact, never do anything
- Why I think OO makes more sense
 - OO is an easier to understand metaphor
 - People are good at dealing with objects
 - Understand their relationships, how two things may or may not work together
 - AOP has similar promise
 - But finding obvious examples is harder
 - The temptation is to characterize a cross cutting concern as just another object
 - Can certainly get similar benefit with well defined interfaces, contracts, simple behavioral code
 - I think this re-inforces my point about limiting its use
- Like dynamism, other languages approach AOP
 - Proxies, interceptors
 - Facilities exist to get at some naturally arise join points

Offset

Topic

- Some systems, like Java EE, reverse the join point, allowing a component author to declare rather than requiring an external point cut
- Of course, AOP is implemented in existing languages, like C++ was original in C
- The fact that no language fully embraces AOP questions how generally applicable
- Good to be aware of what it is, the problems it can better solve
- As always, need to be careful of golden hammer

• Outro

- Contact me
 - Email to feedback@thecommandline.net
 - Web site at <http://thecommandline.net/>
 - IM to command.line@skype
 - Listener comment line is 240-949-2638
 - del.icio.us tag is "for:cmdln"
 - <http://twitter.com/cmdln>
- I'd like to thank libsyn.com for AAC hosting and Wouter de Bie for MP3 hosting
- These notes and the show audio and music are covered by a Creative Commons license
 - <http://creativecommons.org/licenses/by-nc-sa/3.0/us/>
 - Attribution, non-commercial, share alike