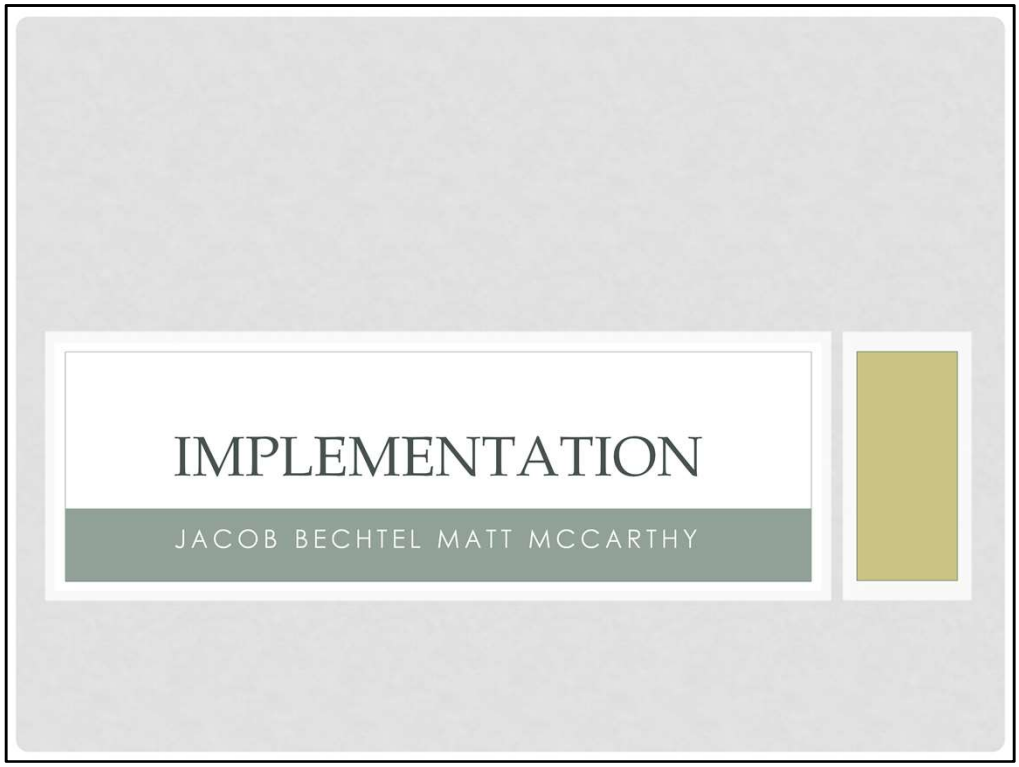


Implementation covers the standards used for coding. This involves what makes for both good code, and what forms compliant code. Compliant code is code which adheres to an Application Programming Interface (API)

CHARACTERISTICS OF GOOD IMPLEMENTATION

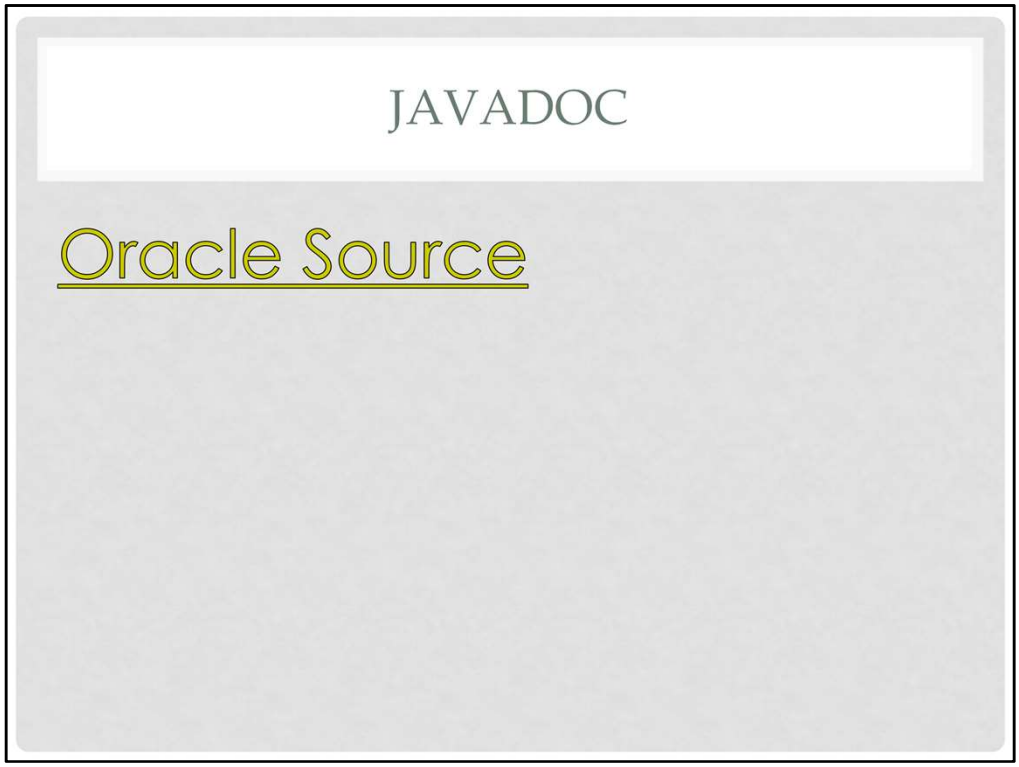
- Readability
- Maintainability
- Performance
- Traceability
- Correctness
- Completeness

- *Readability:* The code can be easily read and understood by other programmers.
- *Maintainability:* The code can be easily modified and maintained. Note that this is related to readability, but it is not exactly the same; for example, this involves the use of Hungarian notation, first put forward by Charles Simonyi of Microsoft (see Simonyi, 1976), in which variable names include abbreviations for the type of variable.
- *Performance:* All other things being equal, the implementation should produce code that performs as fast as possible.
- *Traceability:* All code elements should correspond to a design element. Code can be traced back to design (and design to requirements).
- *Correctness:* The implementation should do what it is intended to do (as defined in the requirements and detailed design).
- *Completeness:* All of the system requirements are met.

CHARACTERISTICS OF GOOD IMPLEMENTATION

- *Readability:* The code can be easily read and understood by other programmers.
- *Maintainability:* The code can be easily modified and maintained. Note that this is related to readability, but it is not exactly the same; for example, this involves the use of Hungarian notation, first put forward by Charles Simonyi of Microsoft (see Simonyi, 1976), in which variable names include abbreviations for the type of variable.
- *Performance:* All other things being equal, the implementation should produce code that performs as fast as possible.
- *Traceability:* All code elements should correspond to a design element. Code can be traced back to design (and design to requirements).
- *Correctness:* The implementation should do what it is intended to do (as defined in the requirements and detailed design).
- *Completeness:* All of the system requirements are met.

READABILITY



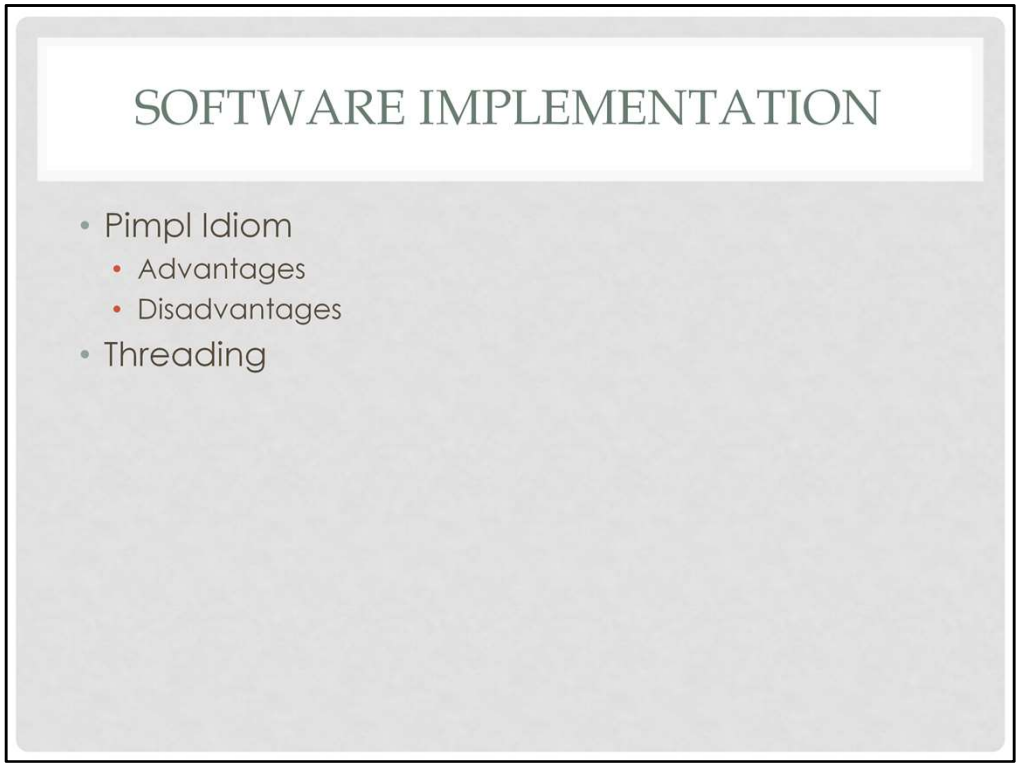
BST
C++ / Heap

CODING GUIDELINES

- Naming (Naming, Variables)
- Consistency
- Case (Camel, Underscore)
- Indentation (DO IT!)
- Function Size (RoT: 50 lines)
- Comments

WHY USE AN API

- Abstraction
- Users (Hide Implementation)
- Legos (Modularization)
- Pencils vs. Pen (Easier to change)
- Easier to Optimize
- First to contact, first to contract



Pimpl:
Pointer to Implementation C++
JAVA: Strong Bridge
Advantages:
- Information Hiding
- Reduced Coupling
- Faster Compiling
- Greater Binary Compatibility
- Lazy Allocation

Disadvantages:
- Extra Objects, Performance hit due to extra level of indirection
- Developer Inconvenience (prefix all private member accesses with mImpl->
  - Harder to read / debug
  - This affects the developer, not the user
  - Compiler will no longer catch certain errors (changing consts behind the abstraction)

Threading:
- Decide beforehand
- Choose Decomposition
- Design Solution
- Implement Serial Version BEFORE parallelizing

SOURCES

- http://www.joelonsoftware.com/articles/Wrong.html
- API Design for C++, Martin Reddy
- Essentials of Software Engineering,