

LLVM C/C++ compiler frontend in Java

Sharing the NetBeans Team's Experience

Petr Kudriavtsev
Vladimir Voskresensky
Oracle

April 26, 2017

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

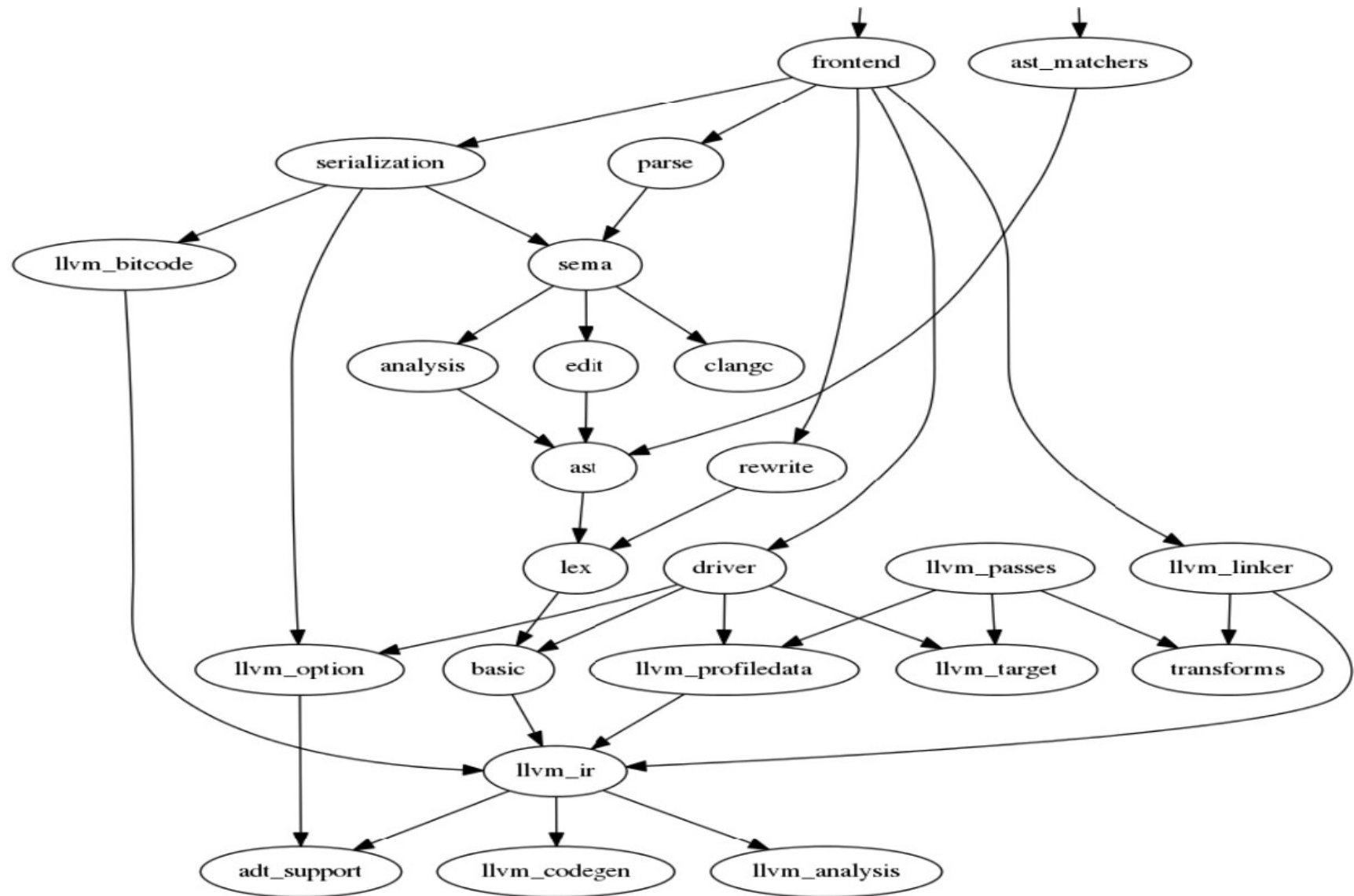
Agenda

- Clang
- Why porting?
- How it was done
- Challenges
- Results & conclusion

Clang

- C/C++ compiler frontend
- Part of LLVM project
- Supports all modern standards
- Modular
- IDE friendly
 - libclang
 - clangd

Clang modules



Why porting?

- Native Clang library requirements without functional regressions:
 - Full access to the strength of technology
 - All Java-aware platforms
 - Safety
 - Debug
 - Performance of native clang
 - JNI/JNA Bridging overhead
 - Upgrade to new Clang release

Clang Technology evaluation (JNI/JNA prototyping)

- Full access to the strength of technology
 - Including AST, ASTRecursiveVisitors, ASTMatchers, CFG ...

Clang Technology evaluation (JNI/JNA prototyping)

- Full access to the strength of technology
 - Including AST, ASTRecursiveVisitors, ASTMatchers, CFG ...
- All Java-aware platforms
 - MacOS, Linux, Windows, and Solaris
 - X86 and SPARC
 - 32 and 64bits

Clang Technology evaluation (JNI/JNA prototyping)

- Full access to the strength of technology
 - Including AST, ASTRecursiveVisitors, ASTMatchers, CFG ...
- All Java-aware platforms
 - MacOS, Linux, Windows, and Solaris
 - X86 and SPARC
 - 32 and 64bits
- Safety
 - Forgot QualType.isNull() check in your Java call? Welcome to JVM Core Dump!

Clang Technology evaluation (JNI/JNA prototyping)

- Full access to the strength of technology
 - Including AST, ASTRecursiveVisitors, ASTMatchers, CFG ...
- All Java-aware platforms
 - MacOS, Linux, Windows, and Solaris
 - X86 and SPARC
 - 32 and 64bits
- Safety
 - Forgot QualType.isNull() check in your Java call? Welcome to JVM Core Dump!
- Debug
 - We hadn't have Mixed-dev in NetBeans yet...

Clang Technology evaluation (JNI/JNA prototyping)

- Full access to the strength of technology
 - Including AST, ASTRecursiveVisitors, ASTMatchers, CFG ...
- All Java-aware platforms
 - MacOS, Linux, Windows, and Solaris
 - X86 and SPARC
 - 32 and 64bits
- Safety
 - Forgot QualType.isNull() check in your Java call? Welcome to JVM Core Dump!
- Debug
 - We hadn't have Mixed-dev in NetBeans yet...
- Performance of native clang
 - Clang preprocessing itself is 2 times slower, parsing is 10x slower

Clang Technology evaluation (JNI/JNA prototyping)

- Full access to the strength of technology
 - Including AST, ASTRecursiveVisitors, ASTMatchers, CFG ...
- All Java-aware platforms
 - MacOS, Linux, Windows, and Solaris
 - X86 and SPARC
 - 32 and 64bits
- Safety
 - Forgot QualType.isNull() check in your Java call? Welcome to JVM Core Dump!
- Debug
 - We hadn't have Mixed-dev in NetBeans yet...
- Performance of native clang
 - Clang preprocessing itself is 2 times slower, parsing is 10x slower
- JNI/JNA Bridging overhead
 - Need to expose whole AST API

Clang Technology evaluation (JNI/JNA prototyping)

- Full access to the strength of technology
 - Including AST, ASTRecursiveVisitors, ASTMatchers, CFG ...
- All Java-aware platforms
 - MacOS, Linux, Windows, and Solaris
 - X86 and SPARC
 - 32 and 64bits
- Safety
 - Forgot QualType.isNull() check in your Java call? Welcome to JVM Core Dump!
- Debug
 - We hadn't have Mixed-dev in NetBeans yet...
- Performance of native clang
 - Clang preprocessing itself is 2 times slower, parsing is 10x slower
- JNI/JNA Bridging overhead
 - Need to expose whole AST API
- Upgrade to new Clang release

Clang Technology evaluation (JNI/JNA prototyping)

- ✘ Full access to the strength of technology
 - Including AST, ASTRecursiveVisitors, ASTMatchers, CFG ...
- ✘ All Java-aware platforms
 - MacOS, Linux, Windows, and Solaris
 - X86 and SPARC
 - 32 and 64bits
- ✘ Safety
 - Forgot QualType.isNull() check in your Java call? Welcome to JVM Core Dump!
- ✘ Debug
 - We hadn't have Mixed-dev in NetBeans yet...
- ✘ Performance of native clang
 - Clang preprocessing itself is 2 times slower, parsing is 10x slower
- ✘ JNI/JNA Bridging overhead
 - Need to expose whole AST API
- ✔ Upgrade to new Clang release

Conclusion: Clang doesn't bring any extra value?

Clang Technology evaluation (JNI/JNA prototyping)

- Full access to the strength of technology
 - Including AST, ASTRecursiveVisitors, ASTMatchers, CFG ...
- All Java-aware platforms
 - MacOS, Linux, Windows, and Solaris
 - X86 and SPARC
 - 32 and 64bits
- Safety
 - Forgot QualType.isNull() check in your Java call? Welcome to JVM Core Dump!
- Debug
 - We hadn't have Mixed-dev in NetBeans yet...
- Performance of native clang
 - Clang preprocessing itself is 2 times slower, parsing is 10x slower
- JNI/JNA Bridging overhead
 - Need to expose whole AST API
- Upgrade to new Clang release

Wait! Let's try Clang in Java!

Clang Technology evaluation (JNI/JNA prototyping)

- ✓ Full access to the strength of technology
 - Including AST, ASTRecursiveVisitors, ASTMatchers, CFG ...
- ✓ All Java-aware platforms
 - MacOS, Linux, Windows, and Solaris
 - X86 and SPARC
 - 32 and 64bits
- ✓ Safety
 - Forgot QualType.isNull() check in your Java call? Welcome to JVM Core Dump!
- ✓ Debug
 - We haven't had Mixed-dev in NetBeans yet...
- ✗ Performance of native clang
 - Clang preprocessing itself is 2 times slower, parsing is 10x slower
- ✓ JNI/JNA Bridging overhead
 - Need to expose whole AST API
- ✗ Upgrade to new Clang release

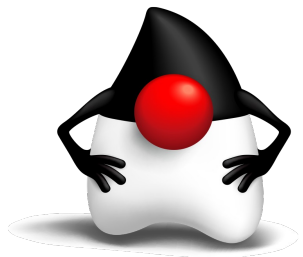
Wait! Let's try Clang in Java!



Clang - Pronunciation: /klaNG/

A loud, resonant metallic sound or series of sounds

- Oxford Dictionary



Clank - Pronunciation: /klaNGk/

A loud, sharp sound or series of sounds, typically made by pieces of metal meeting or being struck together

- Oxford Dictionary

How it was done

- Converter
 - Based on Clang
 - Inspired by `-ast-print` in Clang
 - Produces semantically equivalent code
 - Keeps code as close as possible to original
 - Keeps comments

How it was done

- Converter's output

Java

```
Cpu MyCpu(type, 0, amount); // Create
MySystem.AddModule(&MyCpu); // Add CPL
response = readChar("Enter disk module
switch (response) {
    case 'Q':
        return 2; //premature user request
    case 'R':
        type = Disk::RAID;
        break;
    case 'S':
    default:
        type = Disk::SINGLE;
        break;
}
```

C++

```
MyCpu/*J*/= new Cpu(type, 0, amount); // Create
MySystem.AddModule(/*AddrOf*/MyCpu); // Add CPL
response = readChar("Enter disk module type: (S
switch (response) {
    case 'Q':
        return 2; //premature user requested terminat
    case 'R':
        type = Disk.DiskType.RAID.getValue();
        break;
    case 'S':
    default:
        type = Disk.DiskType.SINGLE.getValue();
        break;
}
```

How it was done

- Converter's output

C++

~~Java~~

```
Cpu MyCpu(type, 0, amount); // Create
MySystem.AddModule(&MyCpu); // Add CPL
response = readChar("Enter disk module

switch (response) {
  case 'Q':
    return 2; //premature user request
  case 'R':
    type = Disk::RAID;
    break;
  case 'S':
  default:
    type = Disk::SINGLE;
    break;
}
```

Java

~~C++~~

```
MyCpu/*J*/= new Cpu(type, 0, amount); // Create
MySystem.AddModule(/*AddrOf*/MyCpu); // Add CPL
response = readChar("Enter disk module type: (S

switch (response) {
  case 'Q':
    return 2; //premature user requested terminat
  case 'R':
    type = Disk.DiskType.RAID.getValue();
    break;
  case 'S':
  default:
    type = Disk.DiskType.SINGLE.getValue();
    break;
}
```

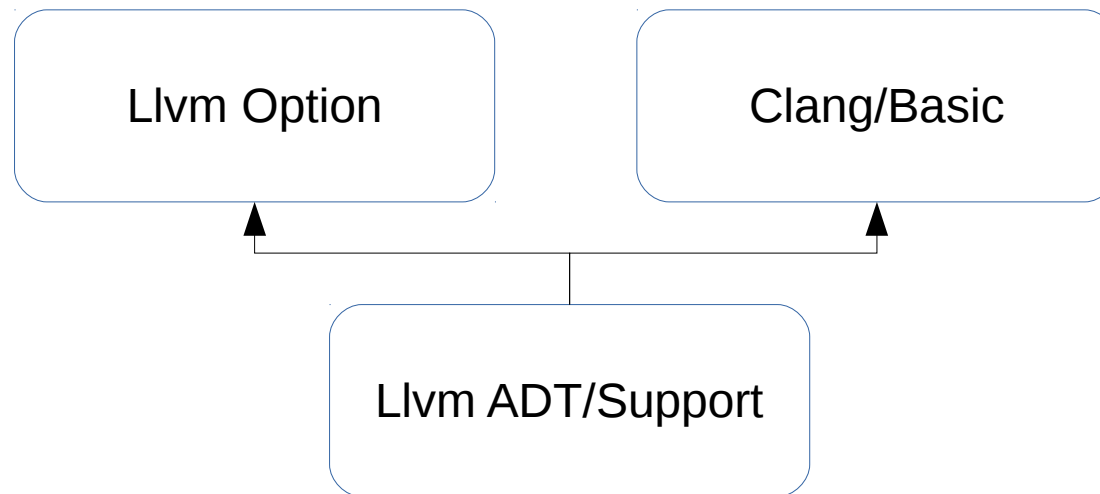
How it was done

- Bottom up approach
 - for API

Llvm ADT/Support

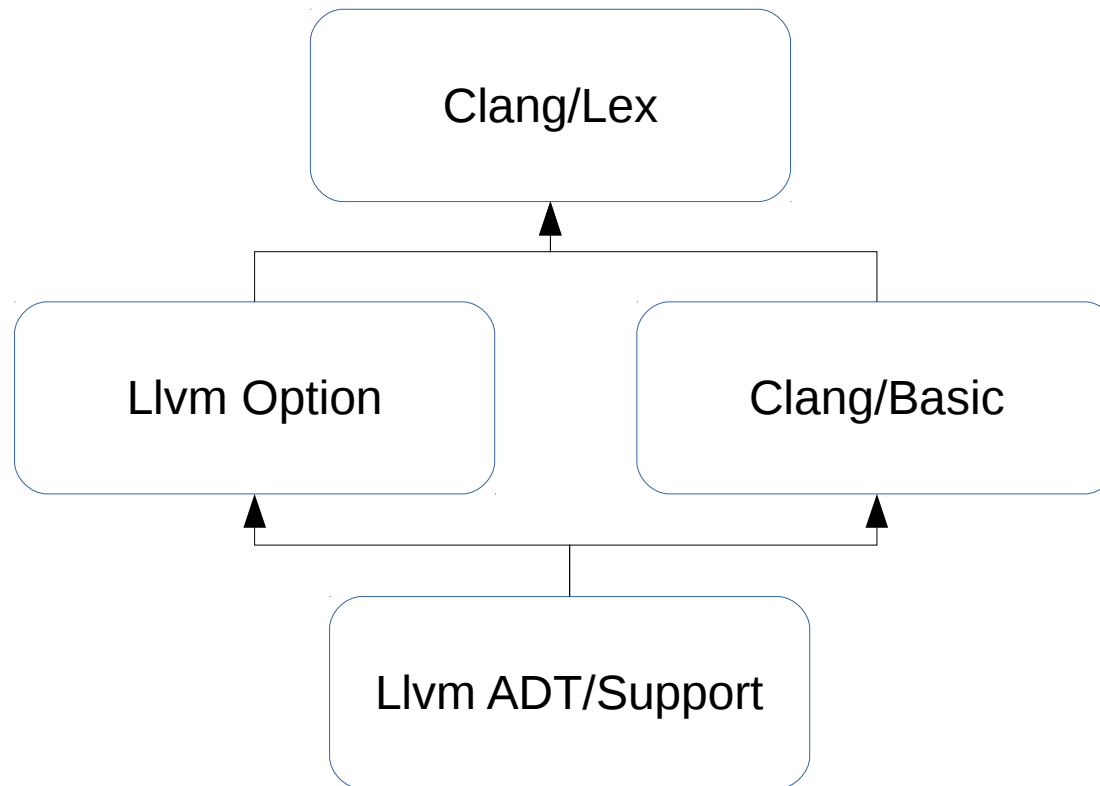
How it was done

- Bottom up approach
 - for API



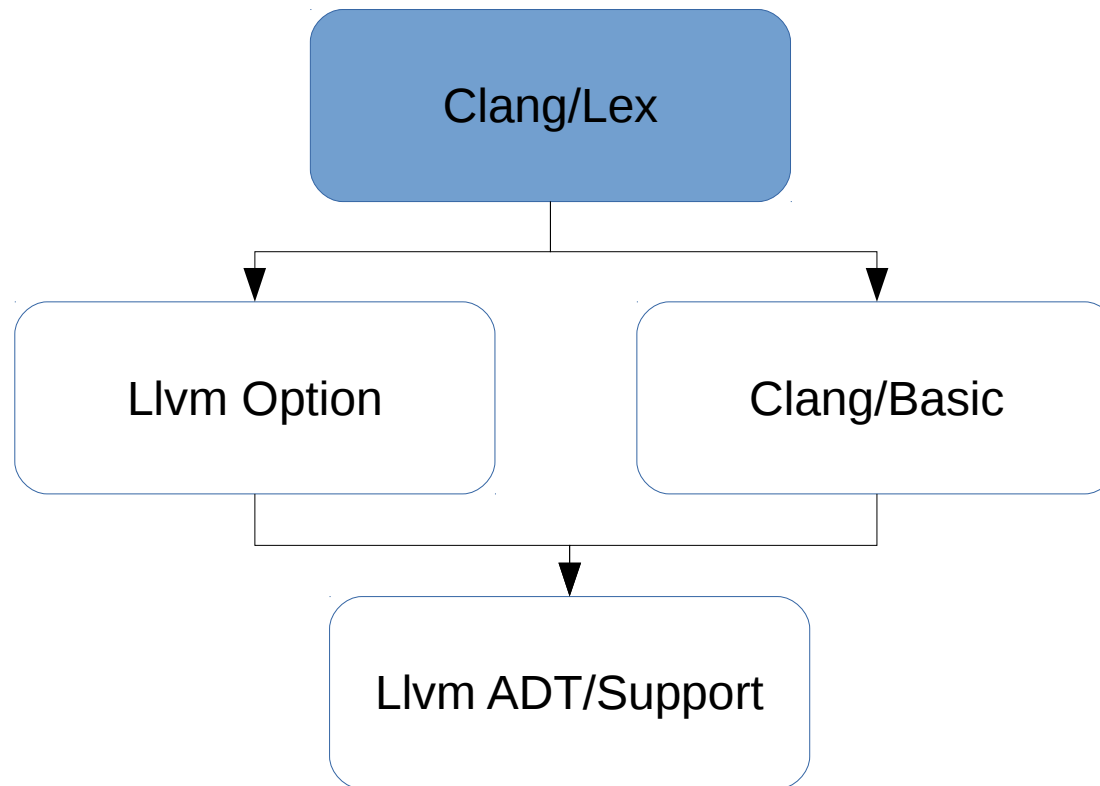
How it was done

- Bottom up approach
 - for API



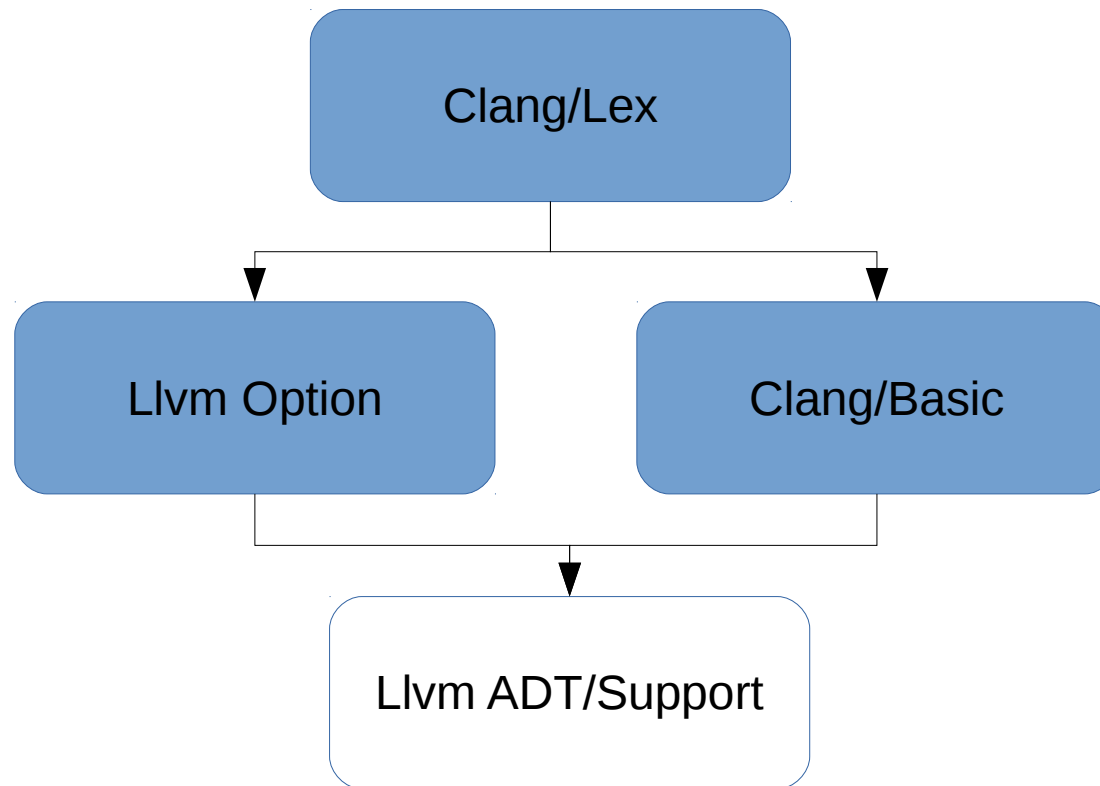
How it was done

- Followed by Top down approach
 - for implementations



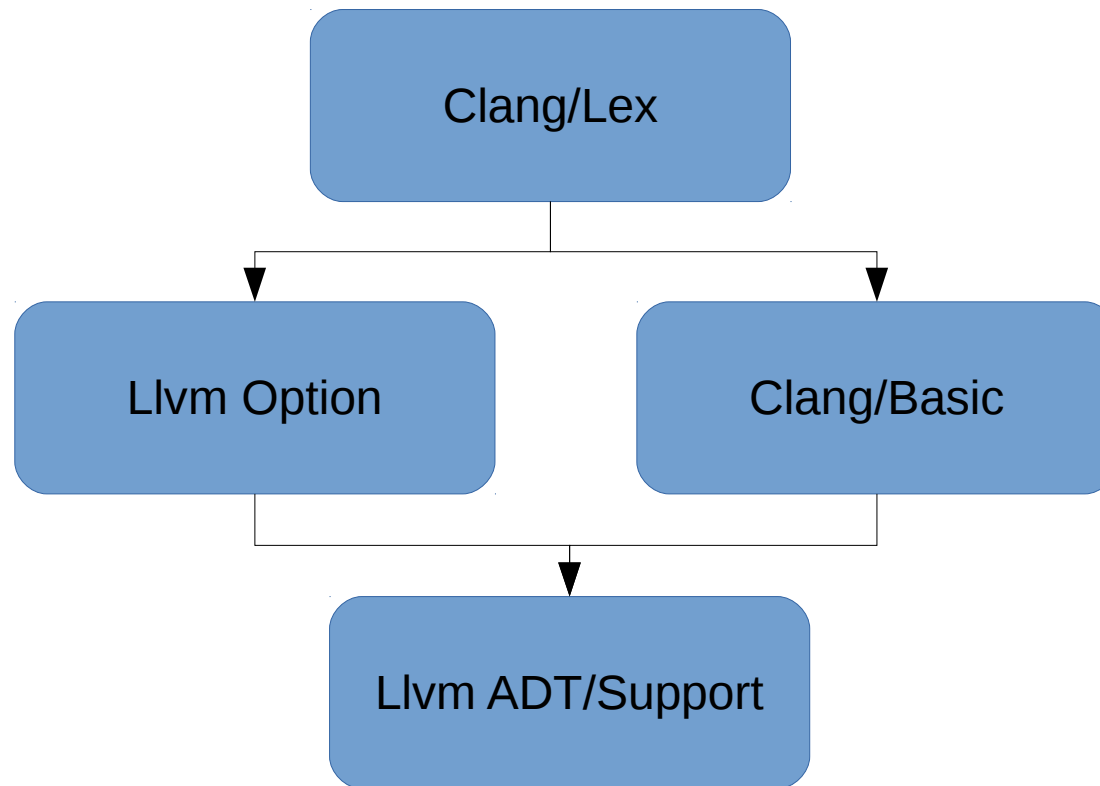
How it was done

- Followed by Top down approach
 - for implementations



How it was done

- Followed by Top down approach
 - for implementations



C++ in Java Technical Challenges

- Names collisions
 - Non-virtual methods in base and derived classes
 - In Java all methods are virtual
 - 'unsigned int' vs 'int' overloaded methods and constructors
- Temporary objects lifecycle
 - Diagnostics are not printed
- Multiple inheritance
- Compile time preprocessor-conditional code in FileSystem
 - Changed `#ifdef/#else/#endif` to runtime
- Split by TUs vs Monolithic Java classes
- `this+1` and `TrailingObjects`
- Custom new operators
- Java code performance

All of them are solvable

✓ Names collisions

- Non-virtual methods in base and derived classes
 - In Java all methods are virtual
- 'unsigned int' vs 'int' overloaded methods and constructors

✓ Temporary objects lifecycle

- Diagnostics are not printed

✓ Multiple inheritance

✓ Compile time preprocessor-conditional code in FileSystem

- Changed `#ifdef/#else/#endif` to runtime

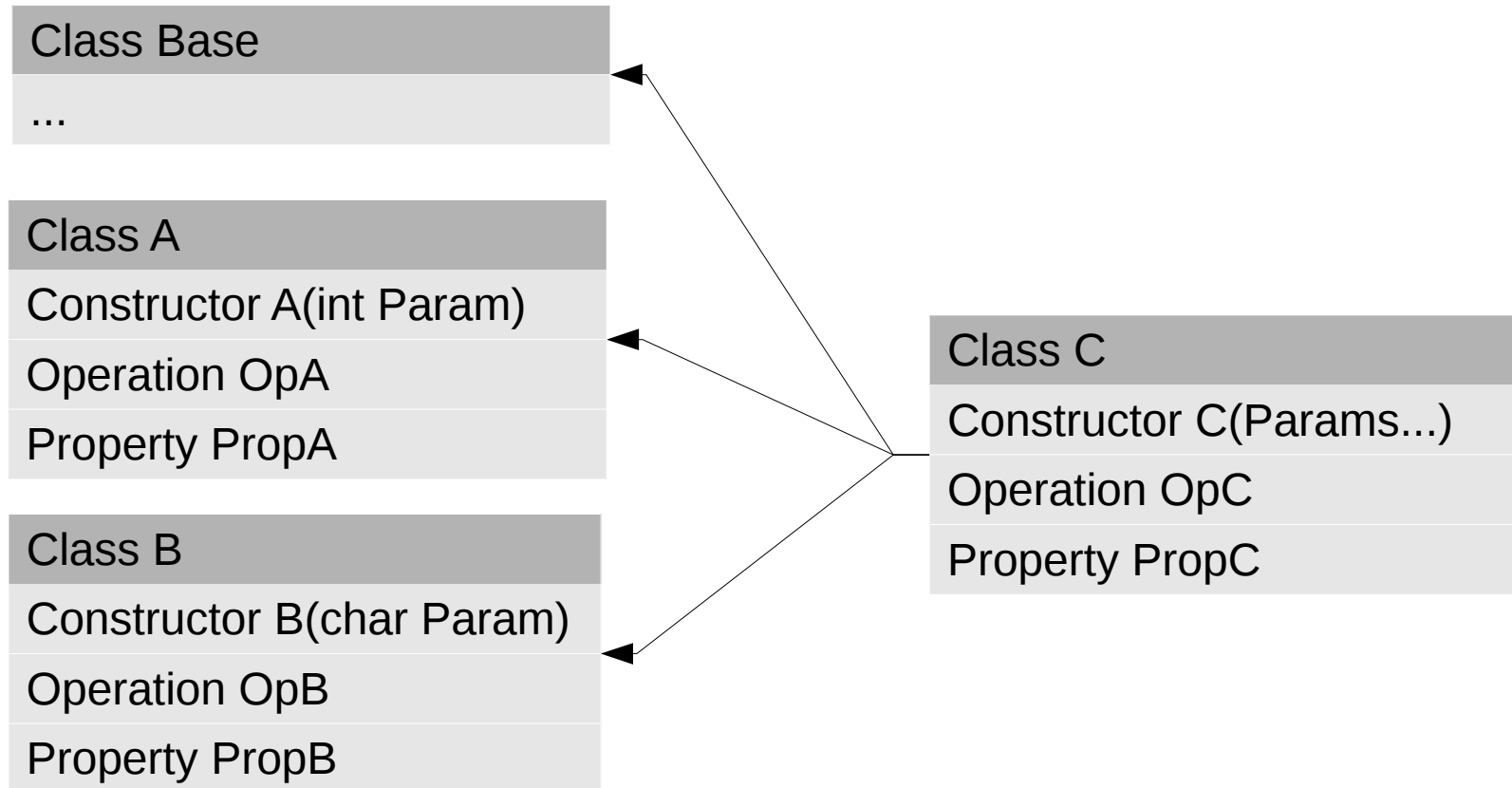
✓ Split by TUs vs Monolithic Java classes

✓ `this+1` and `TrailingObjects`

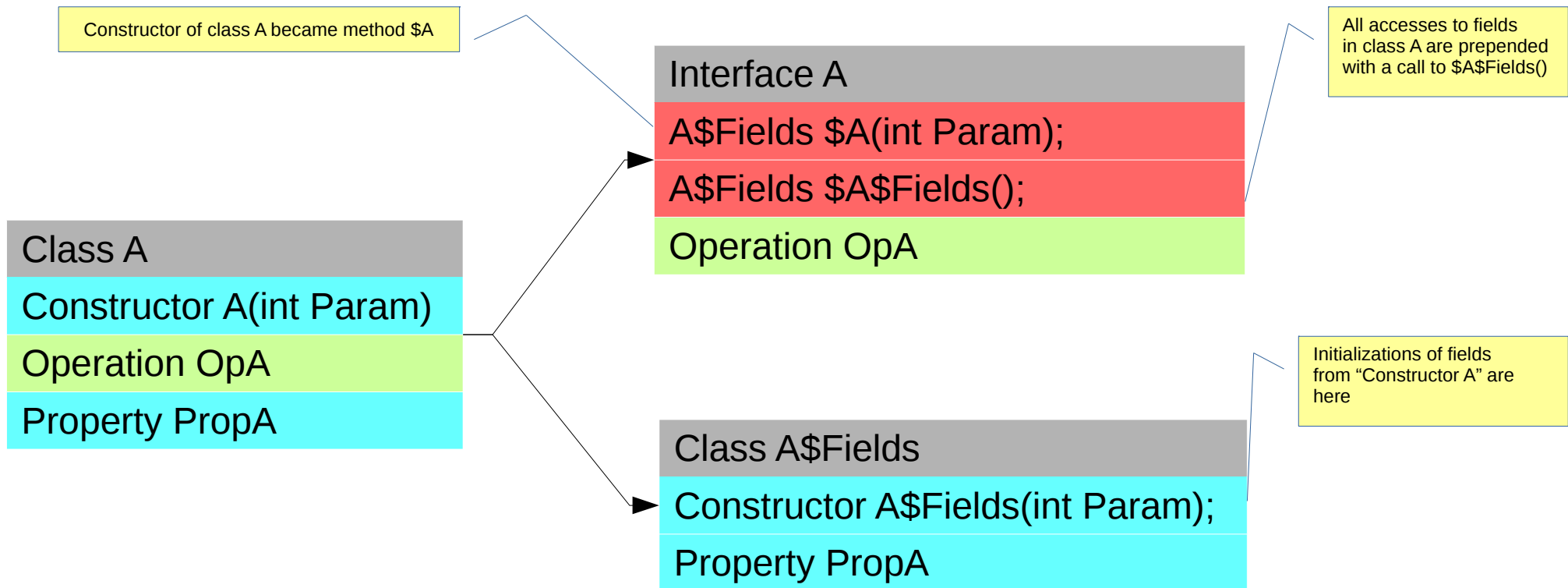
✓ Custom new operators

✓ Java code performance

Multiple inheritance: class C : Base, A, B {...}

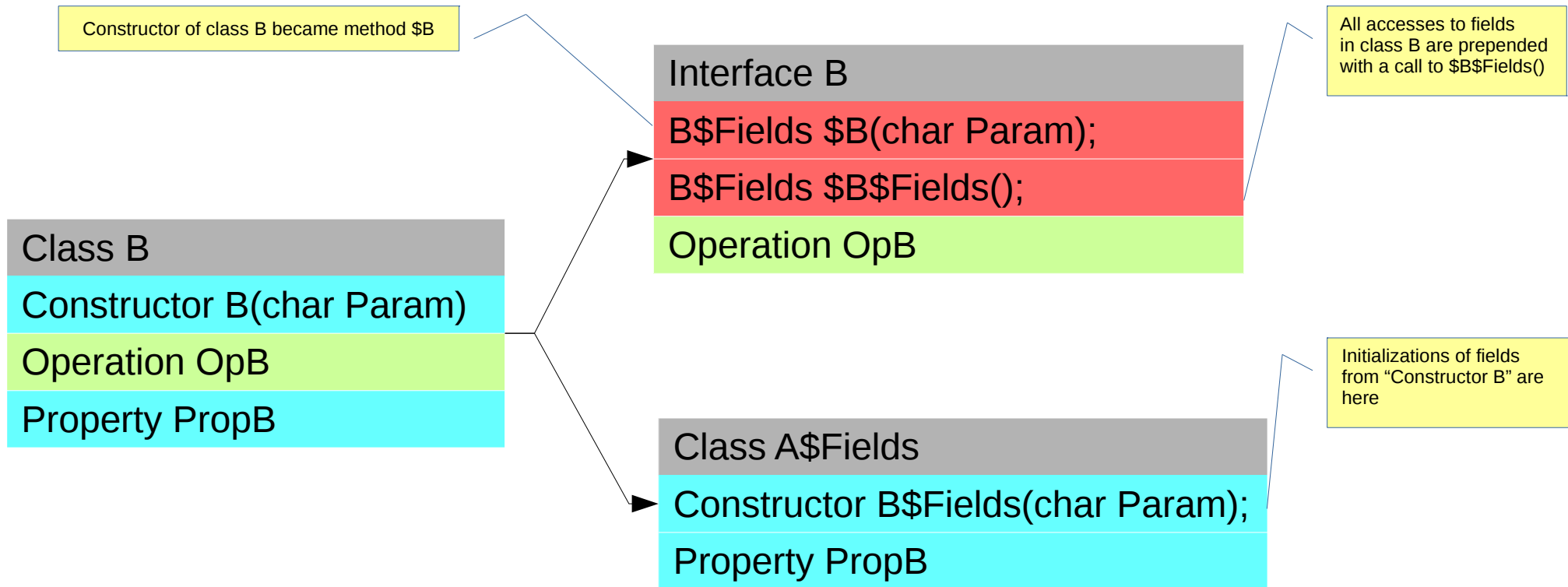


Java: class A → interface A

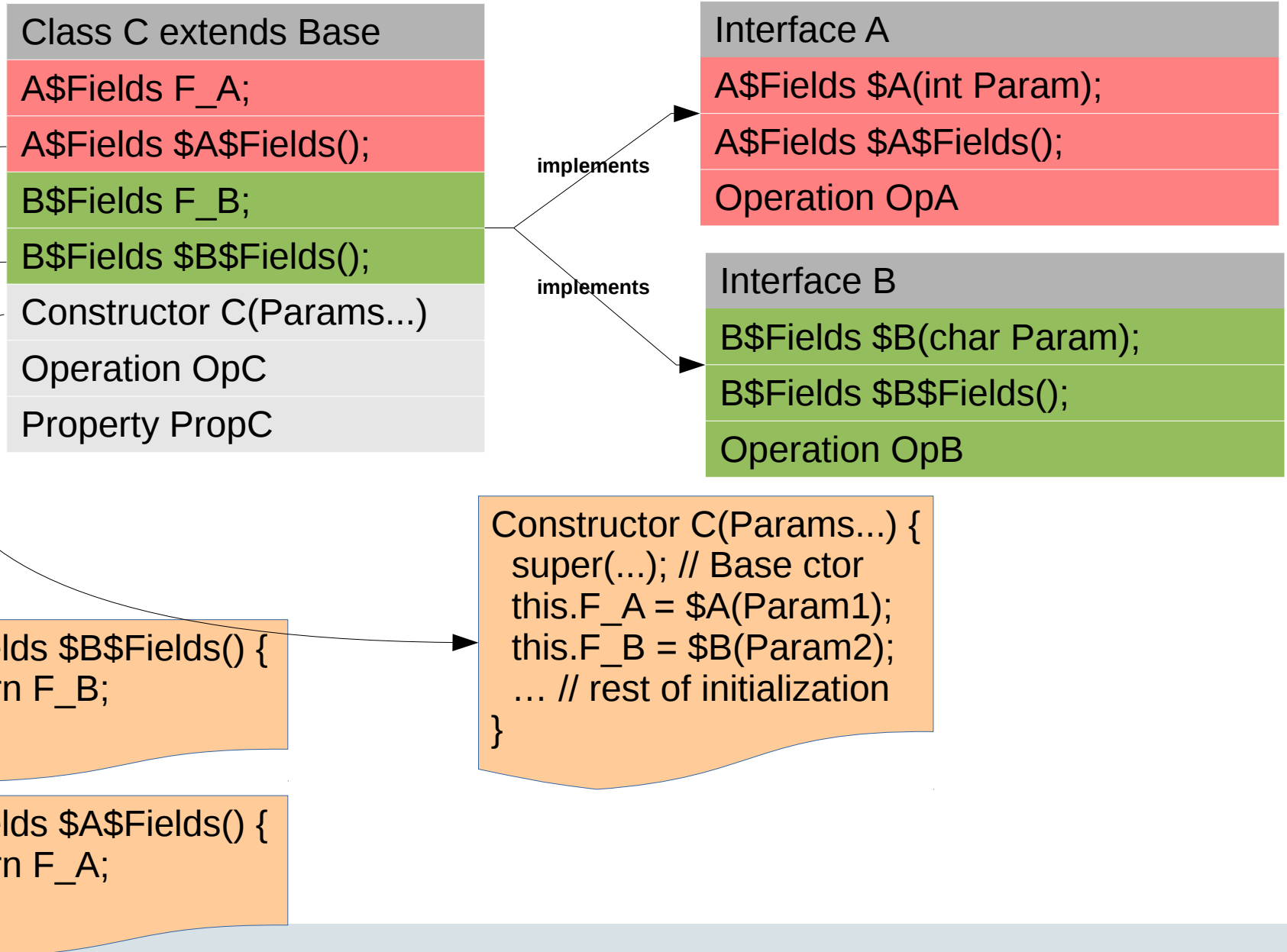


Java: class B → interface B

Exactly the same way as it is done with class A



Java: class C



Method in a class turned into interface

```
clang::DeclContext::decls_begin
```

```
public default /*interface*/ DeclContext.decl_iterator decls_begin() /*const*/ {  
    if (this.hasExternalLexicalStorage()) {  
        this.LoadLexicalDeclsFromExternalStorage();  
    }  
    return new decl_iterator($DeclContext$Fields().FirstDecl);  
}
```

Temporary object lifecycle

C++

```
struct AAA {  
    AAA(int value);  
    operator bool();  
    ~AAA();  
};
```

```
int main() {  
    if (AAA(5)) {  
        return 1;  
    }  
    return 0;  
}
```



AST

```
FunctionDecl 0x61465f0 <line:8:3, lin  
  -CompoundStmt 0x6146b48 <col:14, lin  
    -IfStmt 0x6146ad8 <line:9:5, line:  
      -<<<NULL>>  
      -<<<NULL>>  
      -ExprWithCleanups 0x6146a68 <lin  
        -ImplicitCastExpr 0x6146a50 <c  
          -CXXMemberCallExpr 0x6146a28  
            -MemberExpr 0x61469f0 <col  
              -CXXFunctionalCastExpr 0  
                -CXXBindTemporaryExpr  
                  -CXXConstructExpr 0x  
                    -IntegerLiteral 0x  
  -CompoundStmt 0x6146ab8 <col:17,  
    -ReturnStmt 0x6146aa0 <line:10  
      -IntegerLiteral 0x6146a80 <c  
  -<<<NULL>>  
-ReturnStmt 0x6146b30 <line:12:5,  
  -IntegerLiteral 0x6146b10 <col:1
```

Temporary object lifecycle

AST

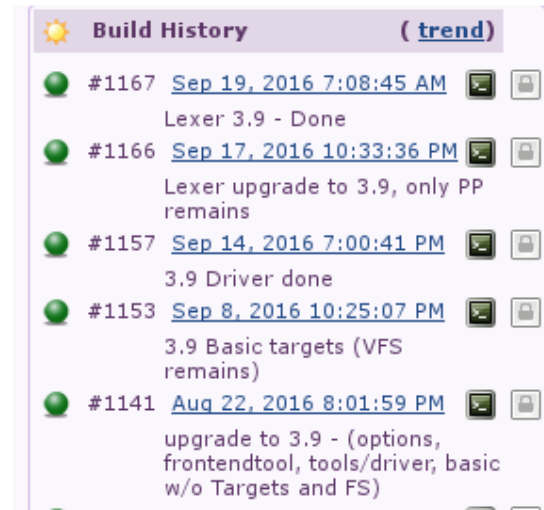
```
-FunctionDecl 0x61465f0 <line:8:3, lin
  -CompoundStmt 0x6146b48 <col:14, lin
    -IfStmt 0x6146ad8 <line:9:5, line:
      | -<<<NULL>>>
      | -<<<NULL>>>
      | -ExprWithCleanups 0x6146a68 <lin
        -ImplicitCastExpr 0x6146a50 <c
          -CXXMemberCallExpr 0x6146a28
            -MemberExpr 0x61469f0 <col
              -CXXFunctionalCastExpr 0
                -CXXBindTemporaryExpr
                  -CXXConstructExpr 0x
                    -IntegerLiteral 0x
          -CompoundStmt 0x6146ab8 <col:17,
            -ReturnStmt 0x6146aa0 <line:10
              -IntegerLiteral 0x6146a80 <c
            -<<<NULL>>>
          -ReturnStmt 0x6146b30 <line:12:5,
            -IntegerLiteral 0x6146b10 <col:1
```

















Java

```
public static int main() {
    JavaCleaner $c$ = $createJavaCleaner();
    try {
        if ($c$.clean($c$.track(new AAA(5)).$bool())) {
            return 1;
        }
        return 0;
    } finally {
        $c$.destroy();
    }
}
```

Clank: Upgrade to Clang 3.9

- Tooling
 - Analyze diffs
 - Analyze dependencies
 - Detect Changed Entities
 - Prepare TODO actions
 - Process Moved and Renamed actions first
 - Drive upgrade
 - Mark progress
 - Track progress

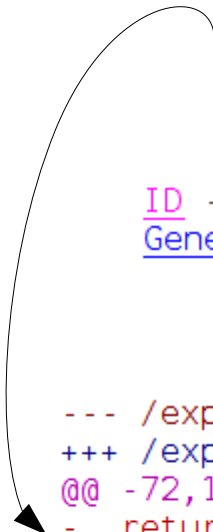


	Build History	(trend)
	#1167	Sep 19, 2016 7:08:45 AM  
	Lexer 3.9 - Done	
	#1166	Sep 17, 2016 10:33:36 PM  
	Lexer upgrade to 3.9, only PP remains	
	#1157	Sep 14, 2016 7:00:41 PM  
	3.9 Driver done	
	#1153	Sep 8, 2016 10:25:07 PM  
	3.9 Basic targets (VFS remains)	
	#1141	Aug 22, 2016 8:01:59 PM  
	upgrade to 3.9 - (options, frontendtool, tools/driver, basic w/o Targets and FS)	

Clank: Upgrade to Clang 3.9

- Update view

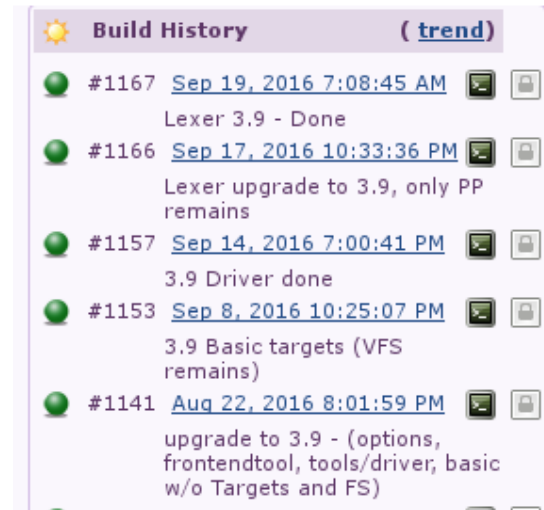
```
Builtin.java(changed/total directs: 1/3, changed/total children: 4/43)
Generate with body, Generate without body, Generate with body in output
Context(changed/total directs: 3/31, changed/total children: 3/31)
Generate with body, Generate without body, Mark as updated
isPure - ADDED (Insert after)
Generate with body, Generate without body, Mark as updated
builtinIsSupported - CHANGED
Generate with body, Generate without body, Mark as updated
isTSBuiltin - COMMENT
Generate with body, Generate without body, Mark as updated
ID - INCLUDE
Generate with body, Generate without body, Mark as updated
```



















```
--- /export/devarea/LLVM38/llvm/tools/clang/lib/Basic/Builtins.cpp
+++ /export/devarea/LLVM39/llvm/tools/clang/lib/Basic/Builtins.cpp
@@ -72,1 +72,3 @@
- return !BuiltinsUnsupported && !MathBuiltinsUnsupported &&
+ bool OclCUnsupported = LangOpts.OpenCLVersion != 200 &&
+     BuiltinInfo.Langs == OCLC20_LANG;
+ return !BuiltinsUnsupported && !MathBuiltinsUnsupported && !OclCUnsupported &&
```

Clank: Upgrade to Clang 3.9

- Tooling
 - Analyze diffs
 - Analyze dependencies
 - Detect Changed Entities
 - Prepare TODO actions
 - Process Moved and Renamed actions first
 - Drive upgrade
 - Mark progress
 - Track progress
- 1 person - 4 weeks for 1.1MLoc
- Improve Upgrade Tools based on feedback



	Build History	(trend)
	#1167	Sep 19, 2016 7:08:45 AM  
	Lexer 3.9 - Done	
	#1166	Sep 17, 2016 10:33:36 PM  
	Lexer upgrade to 3.9, only PP remains	
	#1157	Sep 14, 2016 7:00:41 PM  
	3.9 Driver done	
	#1153	Sep 8, 2016 10:25:07 PM  
	3.9 Basic targets (VFS remains)	
	#1141	Aug 22, 2016 8:01:59 PM  
	upgrade to 3.9 - (options, frontendtool, tools/driver, basic w/o Targets and FS)	

Results

- C/C++ Frontend in Java

Results

- C/C++ Frontend in Java
 - Provides AST

Clang

Clank

<code>TranslationUnitDecl 0x48421b0 <<invalid slo</code>	1	⇒	1	<code>TranslationUnitDecl 0x4201c465 .</code>
<code> -TypedefDecl 0x4842738 <<invalid slo</code>	2		2	<code> -TypedefDecl 0x740fb309 <<inva</code>
<code> ` -BuiltinType 0x4842420 '__int128'</code>	3		3	<code> ` -BuiltinType 0x3ad83a66 '__it</code>
<code> -TypedefDecl 0x4842798 <<invalid slo</code>	4		4	<code> -TypedefDecl 0x6e535154 <<inva</code>
<code> ` -BuiltinType 0x4842440 'unsigned</code>	5		5	<code> ` -BuiltinType 0x15a34df2 'uns:</code>
<code> -TypedefDecl 0x4842ac8 <<invalid slo</code>	6		6	<code> -TypedefDecl 0x5b38c1ec <<inva</code>
<code> ` -RecordType 0x4842880 'struct __NS</code>	7		7	<code> ` -RecordType 0x48e1f6c7 'struc</code>
<code> ` -CXXRecord 0x48427e8 '__NSConsta</code>	8		8	<code> ` -CXXRecord 0x192c3f1e '__NS</code>
<code> -TypedefDecl 0x4842b58 <<invalid slo</code>	9		9	<code> -TypedefDecl 0x1807e3f6 <<inva</code>
<code> ` -PointerType 0x4842b20 'char *'</code>	10		10	<code> ` -PointerType 0x0f1da57d 'char</code>
<code> ` -BuiltinType 0x4842240 'char'</code>	11		11	<code> ` -BuiltinType 0x37271612 'ch</code>
<code> -TypedefDecl 0x4842e78 <<invalid slo</code>	12		12	<code> -TypedefDecl 0x0194fad1 <<inva</code>
<code> ` -ConstantArrayType 0x4842e20 'stru</code>	13		13	<code> ` -ConstantArrayType 0x65f8f5a</code>
<code> ` -RecordType 0x4842c40 'struct __</code>	14		14	<code> ` -RecordType 0x02638011 'sti</code>
<code> ` -CXXRecord 0x4842ba8 '__va_lis</code>	15		15	<code> ` -CXXRecord 0x7a675056 '__</code>
<code>` FunctionDecl 0x4842320 </home/teop</code>	16		16	<code>` FunctionDecl 0x00459f04 </home</code>

Results

- C/C++ Frontend in Java
 - Provides AST
 - Provides CFG

```
int foo() {  
    if (true) {  
        return 1;  
    }  
    return 0;  
}
```




```
int foo()  
[B4 (ENTRY)]  
  Succs (1): B3  
  
[B1]  
  1: 0  
  2: return [B1.1];  
  Preds (1): B3(Unreachable)  
  Succs (1): B0  
  
[B2]  
  1: 1  
  2: return [B2.1];  
  Preds (1): B3  
  Succs (1): B0  
  
[B3]  
  1: true  
  T: if [B3.1]  
  Preds (1): B4  
  Succs (2): B2 B1(Unreachable)  
  
[B0 (EXIT)]  
  Preds (2): B1 B2
```

Results

- C/C++ Frontend in Java
 - Provides AST
 - Provides CFG
 - Provides diagnostics

```
int foo() {  
    if (true) {  
        return &foo;  
    }  
    return 0;  
}
```



```
/home/toor/NetBeansProjects/CppApplication_1/main.cpp:3:12: error: cannot  
initialize return object of type 'int' with an rvalue of type 'int (*)()  
'  
    return &foo;  
           ^~~~~
```

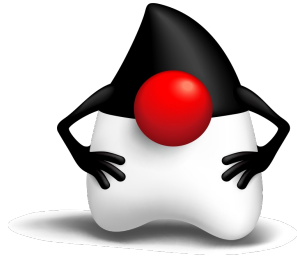
Results

- C/C++ Frontend in Java
 - Provides AST
 - Provides CFG
 - Provides diagnostics
 - Provides different tools
 - Static Analyzer
 - clang-tidy
 - clang-format

Conclusion

- C/C++ conversion to Java is possible
 - Clank
 - ~1.6M lines of code
 - Up to 10x slower than native after conversion
 - On par after optimizations
 - Preprocessor is already included in NetBeans IDE

Clank



Thank you!

Clang

