Fuzion — A New Language for the OpenJDK

Unifying Java’s concepts

TOKIWA software

Who is this guy?

Fridtjof Siebert

Email: siebert@tokiwa.software
github: fridis
twitter: @fridi_s

‘90-‘94	AmigaOberon, AMOK PD
‘97	FEC Eiffel Compiler Sparc / Solaris
‘98-‘99	OSF: TurboJ Java Compiler
‘00-‘01	PhD on real-time GC
‘02-‘19	JamaicaVM real-time JVM based on CLASSSPATH / OpenJDK, VeriFlux static analysis tool
‘20-…	Fuzion
‘21-…	Tokiwa Software

TOKIWA software

Motivation

Many languages overloaded with concepts like classes, methods, interfaces, constructors, traits, records, structs, packages, values, …

➡ Fuzion has one concept: a feature

Today's compilers and tools are more powerful

➡ Tools make better decisions

Systems are safety-critical

➡ we need to ensure correctness

Fuzion Summary

Fuzion

- ➡ uses the **feature** as its main concept
- ➡ is **statically typed**
- ➡ has **inheritance** and **redefinition**
- ➡ uses **value types** and **dynamic (ref) types**
- ➡ encourages **immutability**
- ➡ offloads tasks and decisions from developers to **tools**

Backing Company

- supports development of Fuzion
- currently three employees
- hiring
- searching for funding

Fuzion Language Tutorial

Not part of this talk

➡ online at **flang.dev**

This talk will show how

➡ Java maps to Fuzion

Fuzion vs. Java Constructs

Java class

```
class Complex {
  int re, im;
  Complex(int re, int im) {
    this.re = re;
    this.im = im;
  }
}
```

Fuzion vs. Java Constructs

Java class

```
class Complex {
  int re, im;
  Complex(int re, int im) {
    this.re = re;
    this.im = im;
  }
}
```

Fuzion feature

```
complex(re, im i32) {


}
```

Fuzion vs. Java Constructs

Java class

```
class Complex {
  [..]



}
```

Fuzion feature

```
complex(re, im i32) {



}
```

Fuzion vs. Java Constructs

Java method

```
class Complex {
  [..]
  int radiusSq() {
    return re*re + im*im;
  }
}
```

```
complex(re, im i32) {


}
```

Fuzion vs. Java Constructs

Java method

```
class Complex {
  [..]
  int radiusSq() {
    return re*re + im*im;
  }
}
```

Fuzion feature

```
complex(re, im i32) {
  radius² i32 {
    re*re + im*im;
  }
}
```

Fuzion vs. Java Constructs

Java method

```
class Complex {
  [..]
  int radiusSq() {
    return re*re + im*im;
  }
}
```

Fuzion syntax

```
complex(re, im i32) {
  radius² i32 {
    re*re + im*im;
  }
}
```

Fuzion vs. Java Constructs

Java method

```
class Complex {
  [..]
  int radiusSq() {
    return re*re + im*im;
  }
}
```

Fuzion syntax

```
complex(re, im i32) {
  radius² i32 {
    re*re + im*im
  }
}
```

Fuzion vs. Java Constructs

Java method

```
class Complex {
  [..]
  int radiusSq() {
    return re*re + im*im;
  }
}
```

Fuzion syntax

```
complex(re, im i32) {
  radius² i32 {
    re*re + im*im
  }
}
```

Fuzion vs. Java Constructs

Java method

```
class Complex {
  [..]
  int radiusSq() {
    return re*re + im*im;
  }
}
```

Fuzion syntax

```
complex(re, im i32) {
  radius² i32 {
    re*re + im*im
  }
}
```

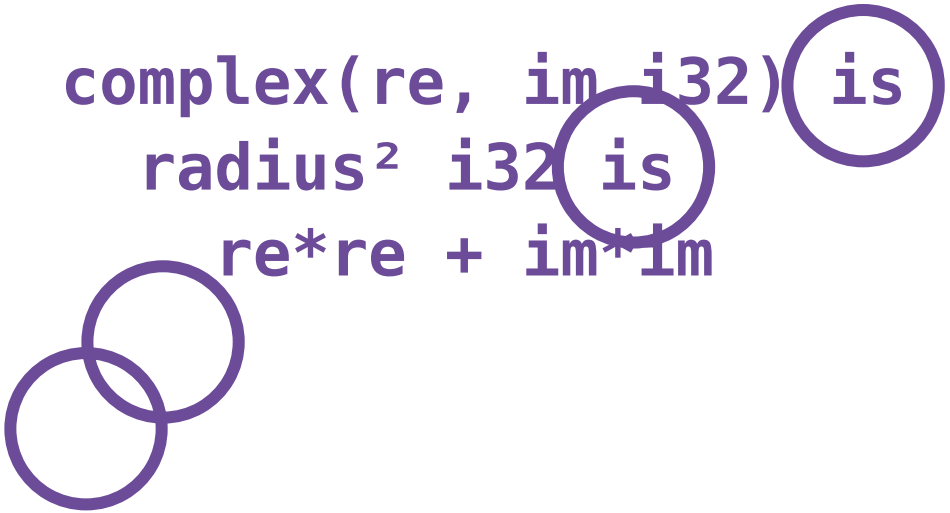
Fuzion vs. Java Constructs

Java method

```
class Complex {
  [..]
  int radiusSq() {
    return re*re + im*im;
  }
}
```

Fuzion syntax

```
complex(re, im i32) is
  radius² i32 is
    re*re + im*im
```



Fuzion vs. Java Constructs

Java method

```
class Complex {
  [..]
  int radiusSq() {
    return re*re + im*im;
  }
}
```

Fuzion syntax

```
complex(re, im i32) is
  radius² i32 is
    re*re + im*im
```

Fuzion vs. Java Constructs

Java method

```
class Complex {
  [..]
  int radiusSq() {
    return re*re + im*im;
  }
}
```

Fuzion syntax

```
complex(re, im i32) is
  radius² =>
    re*re + im*im
```

Fuzion vs. Java Constructs

Java method

```
class Complex {
  [..]
  int radiusSq() {
    return re*re + im*im;
  }
}
```

Fuzion feature

```
complex(re, im i32) is
  radius² =>
    re*re + im*im
```

Fuzion vs. Java Constructs

Java package

```
package myMath;
class Complex {
  [..]



}
```

```
complex(re, im i32) is
  radius² =>
    re*re + im*im
```

Fuzion vs. Java Constructs

Java package

```
package myMath;
class Complex {
  [..]



}
```

Fuzion feature

```
myMath is
  complex(re, im i32) is
    radius² =>
      re*re + im*im
```

Fuzion vs. Java Constructs

Java package

```
package myMath;
class Complex {
  [..]



}
```

Fuzion feature

```
myMath is
  complex(re, im i32) is
    radius² =>
      re*re + im*im
```

unit type feature

➡ no fields

➡ no runtime code

TOKIWA software

Fuzion vs. Java Constructs

```
package myMath;
class Complex {
  [..]



}
```

```
myMath is
  complex(re, im i32) is
    radius² =>
      re*re + im*im
```

Fuzion vs. Java Constructs

Java static method

```
package myMath;
class Complex {
  static Complex origin() {
    return
      new Complex(0,0);
  }
}
```

```
myMath is
  complex(re, im i32) is
    radius² =>
      re*re + im*im
```

Fuzion vs. Java Constructs

Java static method

```
package myMath;
class Complex {
  static Complex origin() {
    return
      new Complex(0,0);
  }
}
```

Fuzion feature

```
myMath is
  complex(re, im i32) is
    radius² =>
      re*re + im*im
```

Fuzion vs. Java Constructs

Java static method

```
package myMath;
class Complex {
  static Complex origin() {
    return
      new Complex(0,0);
  }
}
```

Fuzion feature

```
myMath is
  complex(re, im i32) is
    radius² =>
      re*re + im*im
  complexes is
```

Fuzion vs. Java Constructs

Java static method

```
package myMath;
class Complex {
  static Complex origin() {
    return
      new Complex(0,0);
  }
}
```

Fuzion feature

```
myMath is
  complex(re, im i32) is
    radius² =>
      re*re + im*im
  complexes is
    origin => complex 0 0
```

Fuzion vs. Java Constructs

Java static method

```
package myMath;
class Complex {
  static Complex origin() {
    return
      new Complex(0,0);
  }
}
```

Fuzion feature

```
myMath is
  complex(re, im i32) is
    radius² =>
      re*re + im*im
  complexes is
    origin => complex 0 0
  complex => complexes
```

Fuzion vs. Java Constructs

Java calling constructor

```
var c = new myMath.
          Complex(3,4);
```

Fuzion vs. Java Constructs

Java calling constructor

```
var c = new myMath.
          Complex(3,4);
```

Fuzion calling feature

```
c := myMath.complex(3, 4)
```

Fuzion vs. Java Constructs

Java calling constructor

```
var c = new myMath.
          Complex(3,4);
```

Fuzion syntax

```
c := myMath.complex(3, 4)
```

Fuzion vs. Java Constructs

Java calling constructor

```
var c = new myMath.
          Complex(3,4);
```

Fuzion syntax

```
c := myMath.complex 3 4
```

Fuzion vs. Java Constructs

Java calling constructor

```
var c = new myMath.
          Complex(3,4);
```

Fuzion

```
c := myMath.complex 3 4
```

Fuzion vs. Java Constructs

Java importing package

```
import myMath.Complex;
[..]
var c = new Complex(3,4);
```

Fuzion

```
c := myMath.complex 3 4
```

Fuzion vs. Java Constructs

Java importing package

```
import myMath.Complex;
[..]
var c = new Complex(3,4);
```

Fuzion calling feature

```
m := myMath
[..]
c := m.complex 3 4
```

Fuzion vs. Java Constructs

Java calling method

```
import myMath.Complex;
[..]
var c = new Complex(3,4);
var rSq = c.radiusSq();
```

Fuzion

```
m := myMath
[..]
c := m.complex 3 4
```

Fuzion vs. Java Constructs

Java calling method

```
import myMath.Complex;
[..]
var c = new Complex(3,4);
var rSq = c.radiusSq();
```

Fuzion calling feature

```
m := myMath
[..]
c := m.complex 3 4
r² := c.radius²
```

Fuzion vs. Java Constructs

Java calling static method

```
import myMath.Complex;
[..]
var c = new Complex(3,4);
var rSq = c.radiusSq();
var o = Complex.origin();
```

Fuzion

```
m := myMath
[..]
c := m.complex 3 4
r² := c.radius²
```

Fuzion vs. Java Constructs

Java calling static method

```
import myMath.Complex;
[..]
var c = new Complex(3,4);
var rSq = c.radiusSq();
var o = Complex.origin();
```

Fuzion calling feature

```
m := myMath
[..]
c := m.complex 3 4
r² := c.radius²
o := m.complexes.origin
```

Fuzion vs. Java Constructs

Java calling static method

```
import myMath.Complex;
[..]
var c = new Complex(3,4);
var rSq = c.radiusSq();
var o = Complex.origin();
```

Fuzion calling feature

```
m := myMath
[..]
c := m.complex 3 4
r² := c.radius²
o := m.complexes.origin
```

Fuzion vs. Java Constructs

Java calling static method

```
import myMath.Complex;
[..]
var c = new Complex(3,4);
var rSq = c.radiusSq();
var o = Complex.origin();
```

Fuzion calling feature

```
m := myMath
[..]
c := m.complex 3 4
r² := c.radius²
o := m.complex.origin
```

Fuzion vs. Java Constructs

Java calling static method

```
import myMath.Complex;
[..]
var c = new Complex(3,4);
var rSq = c.radiusSq();
var o = Complex.origin();
```

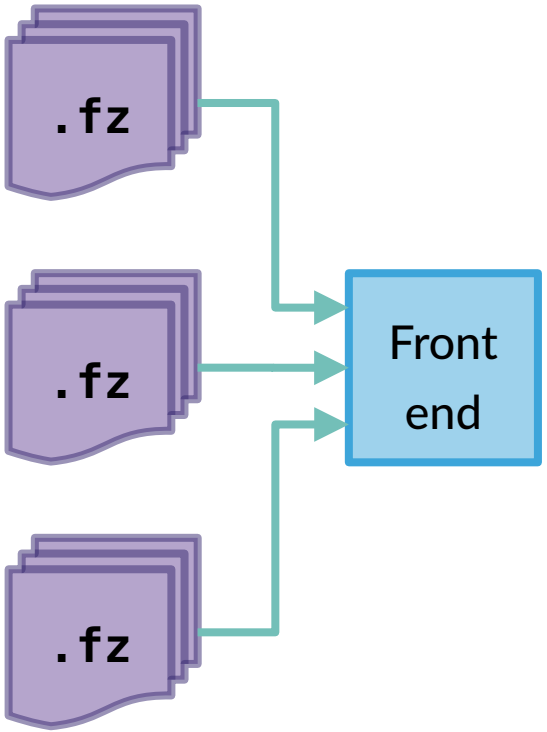
Fuzion calling feature

```
m := myMath
[..]
c := m.complex 3 4
r² := c.radius²
o := m.complex.origin
```

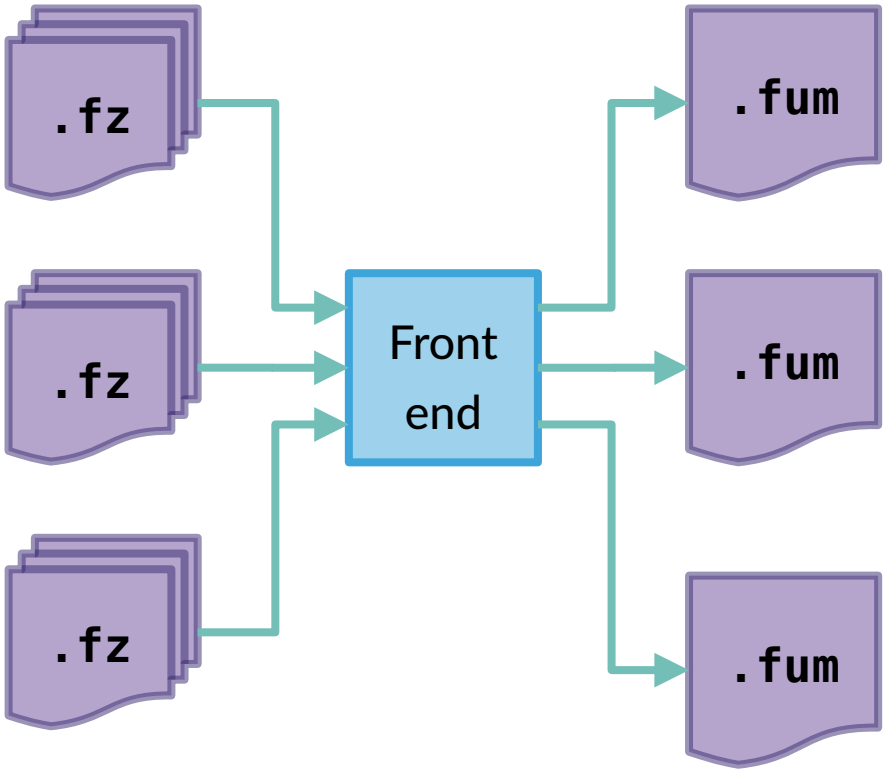
Fuzion Toolchain Design

Fuzion Toolchain Design

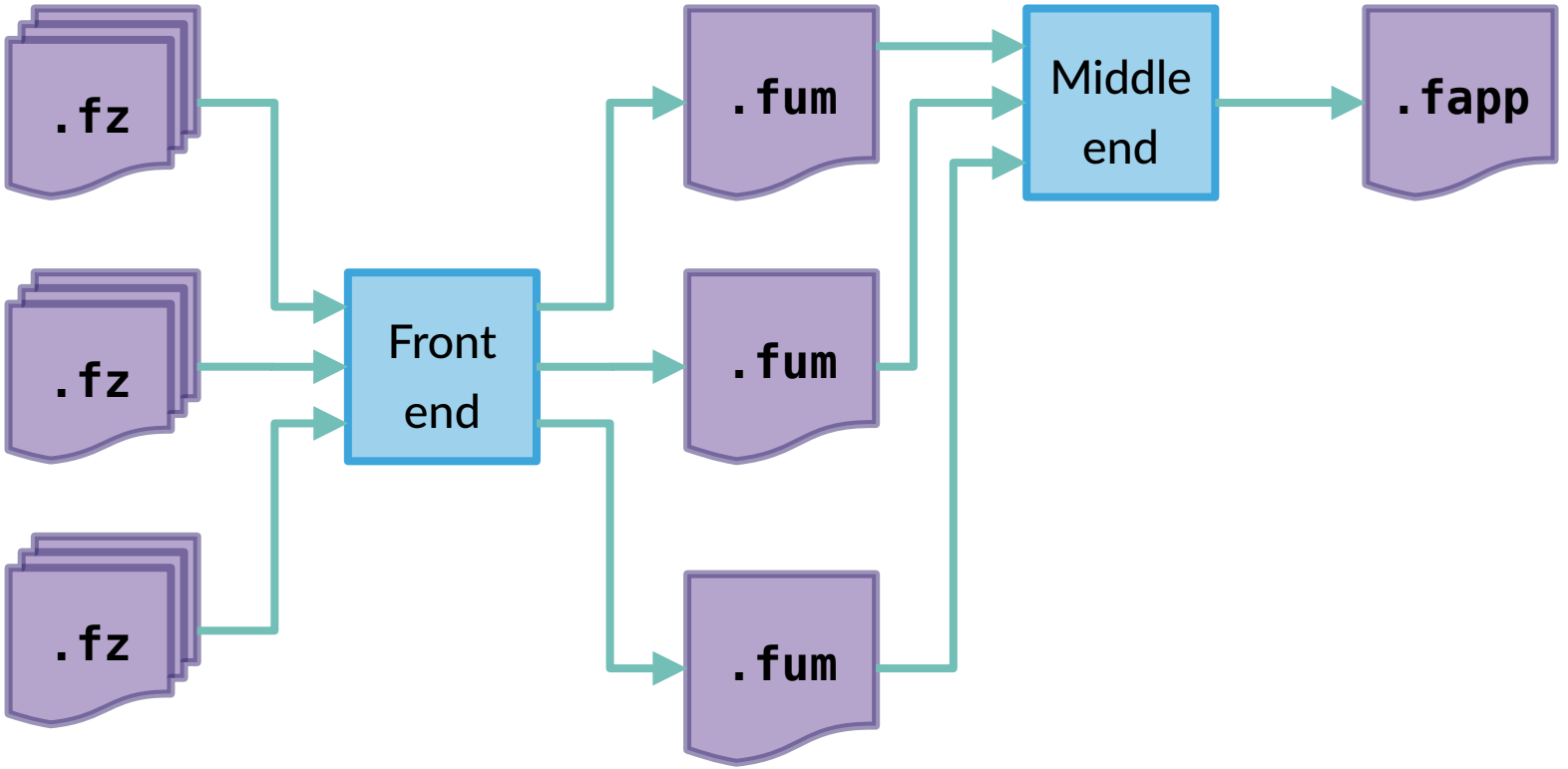
Fuzion Toolchain Design



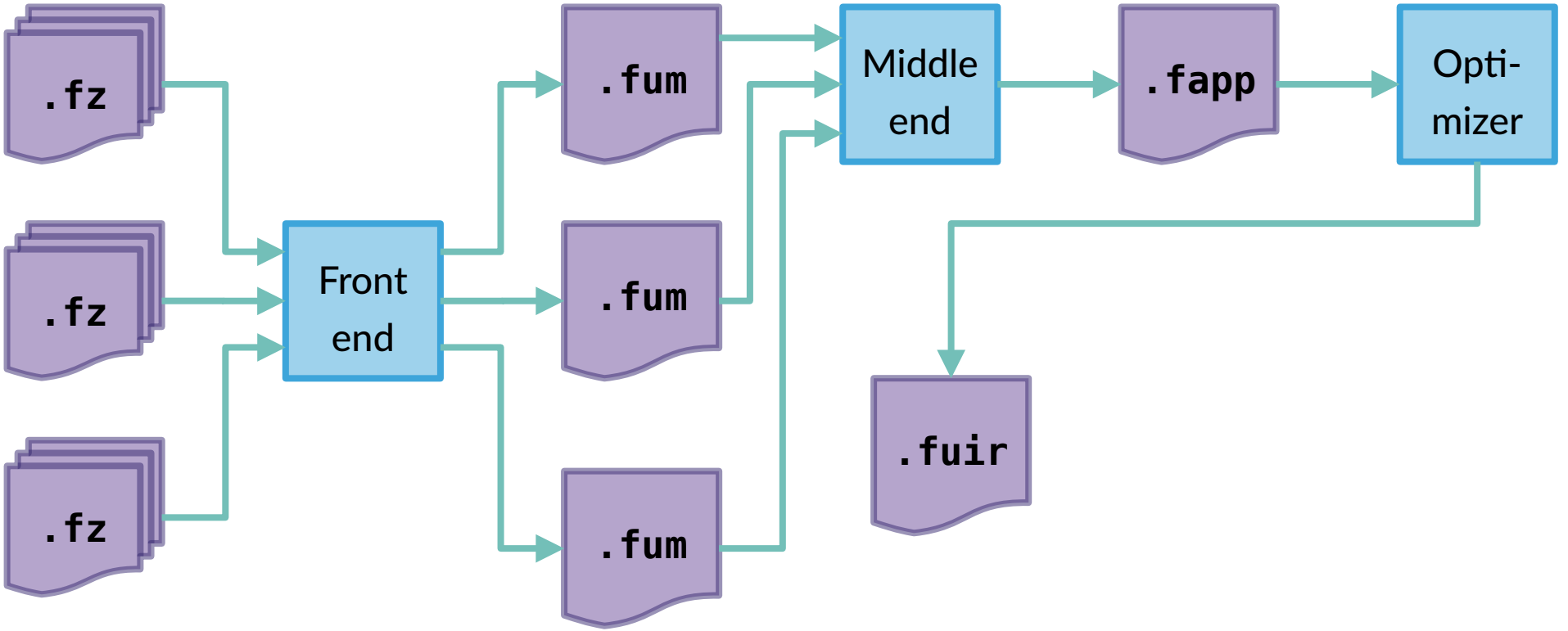
Fuzion Toolchain Design



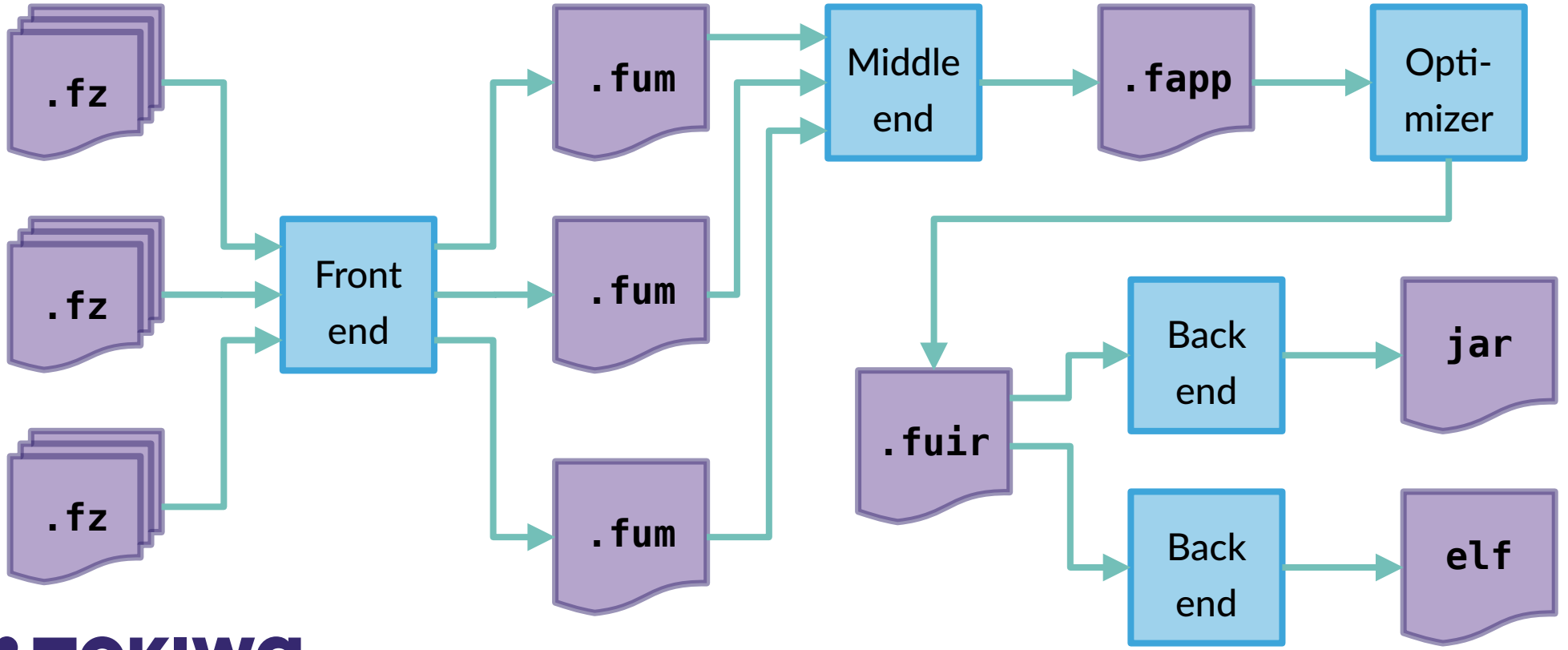
Fuzion Toolchain Design



Fuzion Toolchain Design



Fuzion Toolchain Design



FZJava Tool

Create Fuzion interface to Java module

FZJava Tool

Create Fuzion interface to Java module

FZJava Tool

Create Fuzion interface to Java module


```
java.lang.System.out.println "Hello Java 🌍!"
```

FZJava Tool: Basic approach

FZJava Tool: Basic approach

```
package x.y;
class MyClass {
  MyClass(String arg) {
  }
  void myMethod() {
  }
  static void
    myStaticMethod() {
  }
}
```

FZJava Tool: Basic approach

```
package x.y;
class MyClass {
  MyClass(String arg) {
  }
  void myMethod() {
  }
  static void
    myStaticMethod() {
  }
}
```

```
Java.x.y.MyClass(f void) is
  unit myMethod is
    ...
```

FZJava Tool: Basic approach

```
package x.y;
class MyClass {
  MyClass(String arg) {
  }
  void myMethod() {
  }
  static void
    myStaticMethod() {
  }
}
```

```
Java.x.y.MyClass(f void) is
  unit myMethod is
    ...
Java.x.y.MyClass_static is
  new(arg string) is
    ...
  myStaticMethod is
    ...
```

FZJava Tool: Basic approach

```
package x.y;
class MyClass {
  MyClass(String arg) {
  }
  void myMethod() {
  }
  static void
    myStaticMethod() {
  }
}
```

```
Java.x.y.MyClass(f void) is
  unit myMethod is
    ...
Java.x.y.MyClass_static is
  new(arg string) is
    ...
  myStaticMethod is
    ...
Java.x.y.MyClass =>
  Java.x.y.MyClass_static
```

FZJava Tool: Small Example

```
o = new MyClass(“test”);
o.myMethod();
MyClass.myStaticMethod();
```

```
o := Java.x.y.MyClass_static
       .new “test”
o.myMethod
Java.x.y.MyClass.myStaticMethod
```

FZJava Tool: Small Example

```
                                    MyClass := Java.x.y.MyClass

o = new MyClass("test");            o := MyClass.new "test"
o.myMethod();                       o.myMethod
MyClass.myStaticMethod();           MyClass.myStaticMethod
```

FZJava Tool: Small Example

```
                                    mc := Java.x.y.MyClass

o = new MyClass("test");            o := mc.new "test"
o.myMethod();                       o.myMethod
MyClass.myStaticMethod();           mc.myStaticMethod
```

FZJava Tool: Exceptions

Checked exceptions are mapped to choice type `outcome<T>`:

- ➡ use of `match` required to extract result type `T`
- ➡ `void` result and checked exception mapped to `outcome<unit>`

Unchecked exceptions cause runtime error.

- ➡ in future, may use exception monad

FZJava Tool: Inheritance

Generated features re-build inheritance relation of Java code:

- ➡ assignment compatible if the Java classes are
- ➡ ‘feels’ very similar to the Java code

FZJava Tool: Interfaces

Fuzion supports multiple inheritance for features:

- ➡ interfaces treated like classes

FZJava Tool: Overloading

Java overloading more permissive.

In case of name clash

- ➡ short names for ‚common‘ types
- ➡ fallback: name mangling

FZJava Tool: Example

```
webserver is
  net := Java.java.net
  io  := Java.java.io

  port := 8080
  serversocket := net.ServerSocket.new port
  match serversocket
    err error => say "#### $err ####"
    ss Java.java.net.ServerSocket =>
      match ss.accept
        ...
```

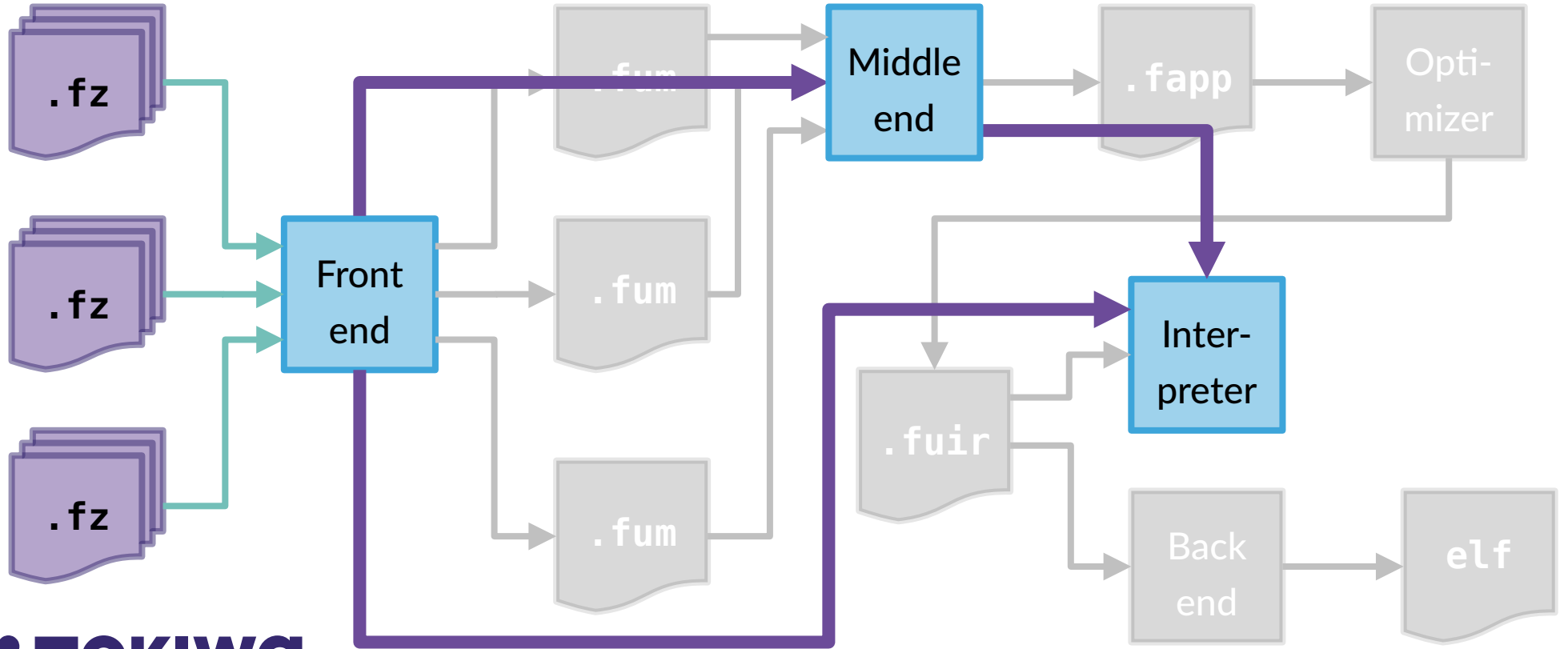
TOKIWA software

Fuzion Interpreter Backend

First quick & dirty back end

➡ direct accesses to front end

Fuzion Interpreter Backend



Fuzion Interpreter Backend

First quick & dirty back end

➡ direct accesses to front end

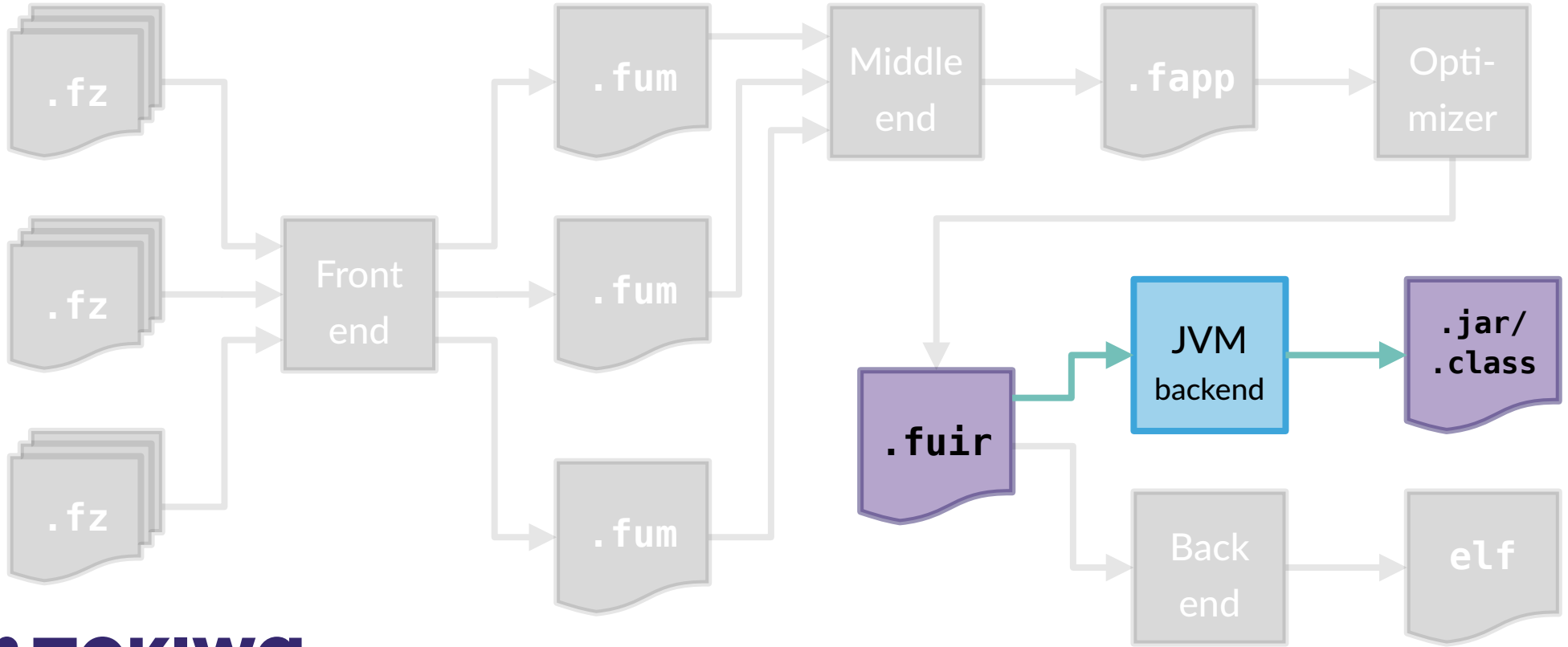
Fuzion Interpreter Backend

First quick & dirty back end

- ➡ direct accesses to front end
- ➡ Fuzion Instances represented as Java Objects with
  - `Object[]` for reference fields
  - `int[]` for non-reference fields
- ➡ Not optimized

Planned Bytecode Backend

Planned Bytecode Backend



Planned Bytecode Backend

Input is FUIR intermediate code

- ➡ Fuzion **clazzes**
  - Features specialized for
    - actual type parameters
    - actual outer clazz
  - five kinds: routine, field, intrinsic, abstract, choice
- ➡ Clazzes of kind routine contain code
  - 14 instructions: assign, call, match, current, const, pop, …
  - no loops, only one branch: match

Planned Bytecode Backend

Probable implementation decisions

- ➡ Java interfaces to implement dynamic dispatch
  - `invokeinterface` highly optimized by JVM
  - `invokedynamic` probably more expensive
- ➡ Fuzion dynamic (ref) instances as Java classes
  - implementing interfaces
  - specialized for type arguments
- ➡ Fuzion value instances as inline primitive values

Fuzion: Next Steps

Development Plan

- ➡ intermediate files: .fum, .fapp, .fuir
- ➡ simple analysis tools: field init, immutability
- ➡ C back-end: GC, floats, etc.
  - interfacing C library code
- ➡ Standard Library
- ➡ Modeling I/O, thread communication and immutability
  - using automatic monadic lifting?

Conclusion

Fuzion is an exciting new language

- ➡ Java used to implement tools
- ➡ OpenJDK important execution target
- ➡ we need
  - to grow our team
  - get developer feedback
  - secure long-term funding
- ➡ please get involved!

http://flang.dev

siebert@tokiwa.software

github.com/tokiwa-software/fuzion