# BEAUTY AND THE BEAST

Eta Haskell for JVM

# JAREK RATAJSKI

@jarek000000

Loves programming since the first line I wrote for C64

Anarchitect @ Engenius GmbH

Java developer (since 1999) with a functional heart.

# ORIGIN

Lots of developers that love Haskell

but when it comes to business…

WE ONLY DO
JAVA HERE
imgflip.com

# SOLUTIONS

# WRITE *HASKELL* USING JAVA

```
            return List.ofAll(iterableRead)
                    .foldLeft(HashMap.<String, Topic>empty(),
                            (existingMap, newElement) ->
                                    existingMap.put( newElemen
                                            existingMap.get( n
                                                    .getOrElse
                                                    .addMessag
                                    );
        } catch (IOException e) {
            return HashMap.<String, Topic>empty();
        }
```

# WRITE *HASKELL* USING SCALA

```
implicit val treeApplicative: Applicative[Tree] = new Applic
  def point[A](a: => A): Tree[A] = Tree(a, Seq.empty)

  def ap[A, B](fa: => Tree[A])(tf: => Tree[A => B]): Tree[B]
    val Tree(f, tfs) = tf

    Tree(f(fa.root), fa.children.map(t => t.map(f)) ++ tfs.m
   }
 }
```

# WRITE HASKELL USING HASKELL (ETA)

deploy to JVM

# work with Java

# TYPELEAD

Company founded to create eta and in the future provide commercial support for it.

I am not associated with Typelead.

Whatever I say show here are mine own studies, with some help of typelead developers and community.

I am neither experienced haskell nor eta developer.

What I say might be wrong or may not reflect the reality or the future.

I tried to do my best

We talk about half-finished product.

# ETA 1.2.3 INTRO

# QUICKSORT(*)

```
quicksort [] = []
quicksort (x:xs) = quicksort left ++ [x] ++ quicksort right
    where
          left  = [ y | y <- xs, y < x ]
          right = [ y | y <- xs, y >= x ]

main = do
        let result = quicksort arr
        putStrLn $ show result
        where
            arr = [1,7,9,12,90,1,-1,22,0]
```

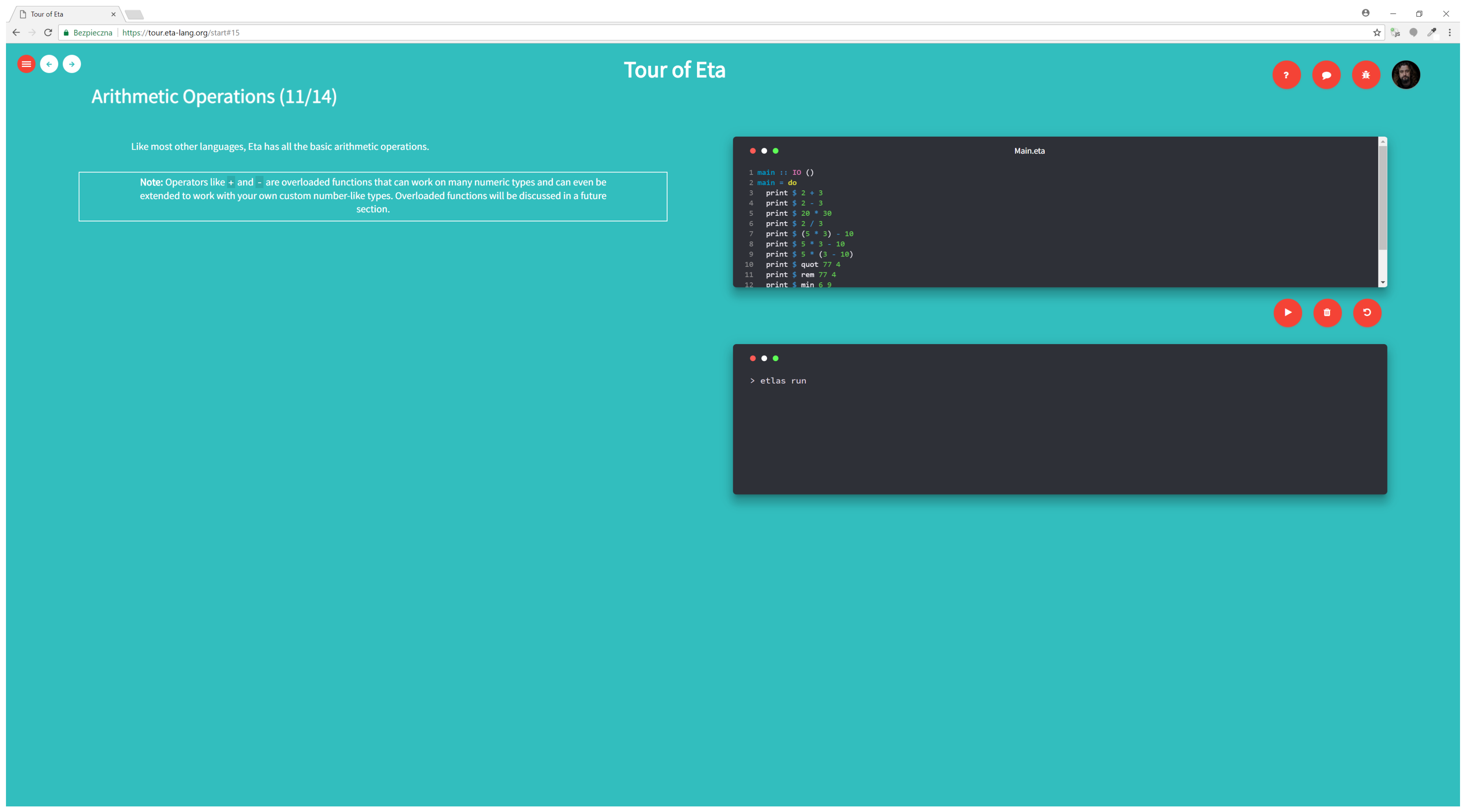
```
$ eta Main.hs

$ java -jar Main.jar
```

```
[-1,0,1,1,7,9,12,22,90]
```

# ETLAS (CABAL FOR ETA)

```
$ etlas init

$ etlas build

$ etlas run
```

# For more info see Eta tour page



## Tour of Eta

### Arithmetic Operations (11/14)

Like most other languages, Eta has all the basic arithmetic operations.

**Note:** Operators like `+` and `-` are overloaded functions that can work on many numeric types and can even be extended to work with your own custom number-like types. Overloaded functions will be discussed in a future section.

Main.eta

```
main :: IO ()
main = do
  print $ 2 + 3
  print $ 2 - 3
  print $ 20 * 30
  print $ 2 / 3
  print $ (5 * 3) - 10
  print $ 5 * 3 - 10
  print $ 5 * (3 - 10)
  print $ quot 77 4
  print $ rem 77 4
  print $ min 6 9
```

```
> etlas run
```

## ETA SPECIAL

# ETA =~= GHC FOR JVM

backend for GHC -> great compatibility

# STG machine

```
Fibb.fibbtcoinner
  :: forall a_a7U9 a1_a7UA.
     (GHC.Classes.Eq a_a7U9, GHC.Num.Num a_a7U9, GHC.Num.Num a
     a_a7U9 -> a1_a7UA -> a1_a7UA -> a1_a7UA
[GblId,
 Arity=6,
 Caf=NoCafRefs,
 Str=<S(C(C(S))L),U(C(C1(U)),A)><L,U(A,C(C1(U)),A,A,A,A,C(U))>
 Unf=OtherCon []] =
    \r srt:SRT:[] [$dEq_s8MJ
                   $dNum_s8MK
                   $dNum1_s8ML
                   eta_s8MM
                   eta1_s8MN
                   eta2_s8MO]
```

# SPINELESS, TAGLESS G-MACHINE

# STG

*...It defines how the Haskell evaluation model should be efficiently implemented on standard hardware. ...*

STG =~= (*bytocode* or llvm)

# 1ST PHASE HS TO STG

Eta compiler in a phase **.hs to STG**

..is simply a GHC code! (forked)

# 2ND PHASE - STG TO BYTECODE / JVM

```
0: getstatic     #127                // Field DZMZN:Leta/runti
3: ifnull        9
6: goto          35
9: ldc           #3                  // class ghc_prim/ghc/Typ
11: dup
12: astore_0
13: monitorenter
14: getstatic     #127                // Field DZMZN:Leta/runt
17: ifnull        23
20: goto          33
23: new           #129                // class ghc_prim/ghc/ty
26: dup
27: invokespecial #131                // Method ghc_prim/ghc/t
30: putstatic     #127                // Field DZMZN:Leta/runt
33: aload_0
```

# C IMPORTS

GHC supports native(C language) calls. (for instance used in Base packages)

# Eta rewrites those parts to use jvm calls.

## original GHC `Float.hs` fragment

```
foreign import ccall unsafe "isFloatNaN" isFloatNaN :: Float -
foreign import ccall unsafe "isFloatInfinite" isFloatInfinite
foreign import ccall unsafe "isFloatDenormalized" isFloatDenor
foreign import ccall unsafe "isFloatNegativeZero" isFloatNegat
foreign import ccall unsafe "isFloatFinite" isFloatFinite :: F
```

## Eta `Float.hs` fragment

```
foreign import java unsafe "@static java.lang.Float.isNaN"
  isFloatNaN :: Float -> Bool
foreign import java unsafe "@static java.lang.Float.isInfinite
  isFloatInfinite :: Float -> Bool
foreign import java unsafe "@static eta.base.Utils.isFloatDeno
  isFloatDenormalized :: Float -> Bool
foreign import java unsafe "@static eta.base.Utils.isFloatNega
  isFloatNegativeZero :: Float -> Bool
foreign import java unsafe "@static eta.base.Utils.isFloatFini
  isFloatFinite :: Float -> Bool
```

# ETLAS

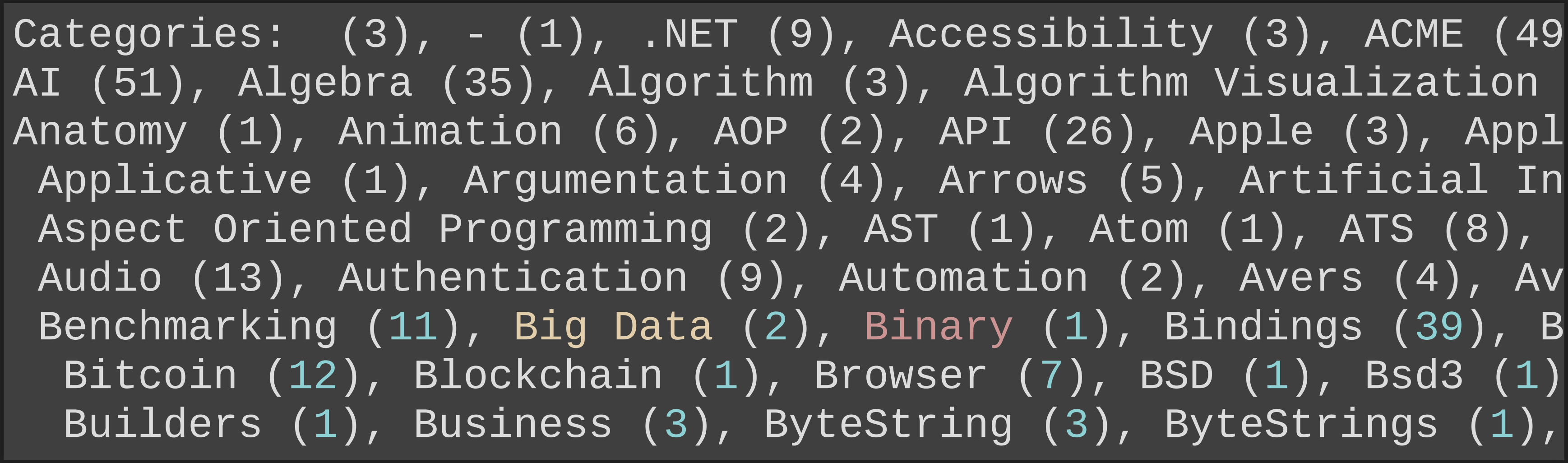
Haskell GHC developers use cabal (or stack).

Etlas is eta tool which is ~ cabal. It uses .cabal file format with extensions.

# HACKAGE

Tons of libraries for haskell.

De facto standard.



```
Categories:  (3), - (1), .NET (9), Accessibility (3), ACME (49
AI (51), Algebra (35), Algorithm (3), Algorithm Visualization
Anatomy (1), Animation (6), AOP (2), API (26), Apple (3), Appl
 Applicative (1), Argumentation (4), Arrows (5), Artificial In
 Aspect Oriented Programming (2), AST (1), Atom (1), ATS (8),
 Audio (13), Authentication (9), Automation (2), Avers (4), Av
 Benchmarking (11), Big Data (2), Binary (1), Bindings (39), B
  Bitcoin (12), Blockchain (1), Browser (7), BSD (1), Bsd3 (1)
  Builders (1), Business (3), ByteString (3), ByteStrings (1),
```

# ETA HACKAGE PATCHES

Project typelead/hackage == patches for common hackage projects.

# Mostly 1 to 1 native C to Java calls changes.

https://github.com/typelead/eta-hackage/blob/master/patches/text-1.2.2.2.patch

```
  {-# INLINE equal #-}

-foreign import ccall unsafe "_hs_text_memcpy" memcpyI
+foreign import java unsafe "@static eta.text.Utils.memcpy" m
     :: MutableByteArray# s -> CSize -> ByteArray# -> CSize -

-foreign import ccall unsafe "_hs_text_memcmp" memcmp
+foreign import java unsafe "@static eta.text.Utils.memcmp" m
     :: ByteArray# -> CSize -> ByteArray# -> CSize -> CSize -
```

# SUPPORTS COMPILE EXTENSIONS

```
{-# LANGUAGE FlexibleContexts, DataKinds, TypeFamilies, RankNT
--  whatever
```

Eta is as close as you can get with Haskell/GHC on JVM

Lots of crazy haskell codes that use GHC extensions work on Eta without any problems.

# BASIC OPTIMISATIONS

# TCO

# Naive fibonacci

```
fibnaive  0 = 1
fibnaive  1 = 1
fibnaive  n = fibnaive ( n-1) + fibnaive ( n - 2)
```

# better

```
fibtcoinner 0 sum presum  = sum
fibtcoinner n sum presum = fibtcoinner  (n-1) (sum + presum) s

fibtco n = fibbtcoinner n 1 0
```

# Java

```
private static BigInteger fibonacci(int n, BigInteger sum, Big
        if ( n== 0) {
            return sum;
        } else {

            return fibonacci(n-1, sum.add(presum), sum);
        }
    }
```

How much java stands?

# Eta

```
fibtcoinner 0 sum presum  = sum
fibtcoinner n sum presum = fibtcoinner  (n-1) (sum + presum) s

fibtco n = fibbtcoinner n 1 0
```

First results….

```
[1,1,2,4,8,16,32,64,128,256,512...]
```

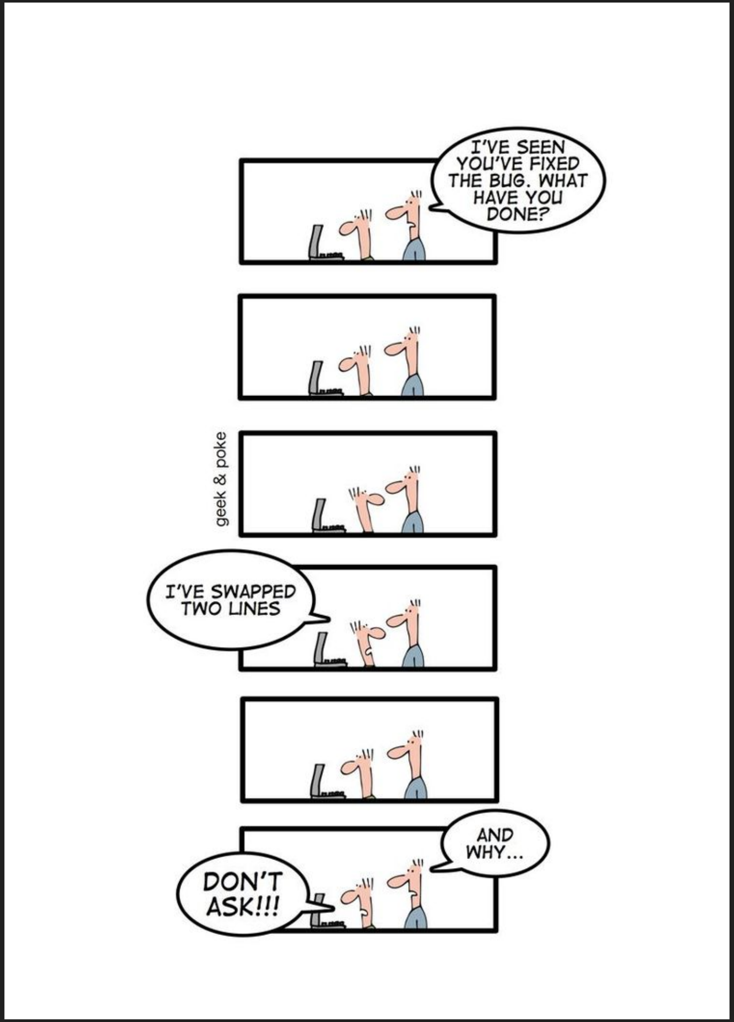
# BUG #603

It took couple of nights to fix this bug. I've Learned haskell...

# Error

```
while(var8) {
    Main.sat_s7YH var12 = new Main.sat_s7YH(var3);
    var1.R1 = var2;
    Closure var13 = Classes.zeze().enter(var1).apply2(var1, (
    if (!(var13 instanceof False)) {
        return ((Closure)var10).evaluate(var1);
    }

    Main.sat_s7YM var14 = new Main.sat_s7YM(var4, (Closure)va
    Main.sat_s7YL var15 = new Main.sat_s7YL(var3, (Closure)va
    var9 = var15;    //assign n-1
    var10 = var14;  //assign new sum
    var11 = var14;  //assign presum
    var8 = true;
```

```
while( n > 0) {
      n = n -1;
      newSum = presum + sum
      sum =  newSum
      presum = sum
}
```



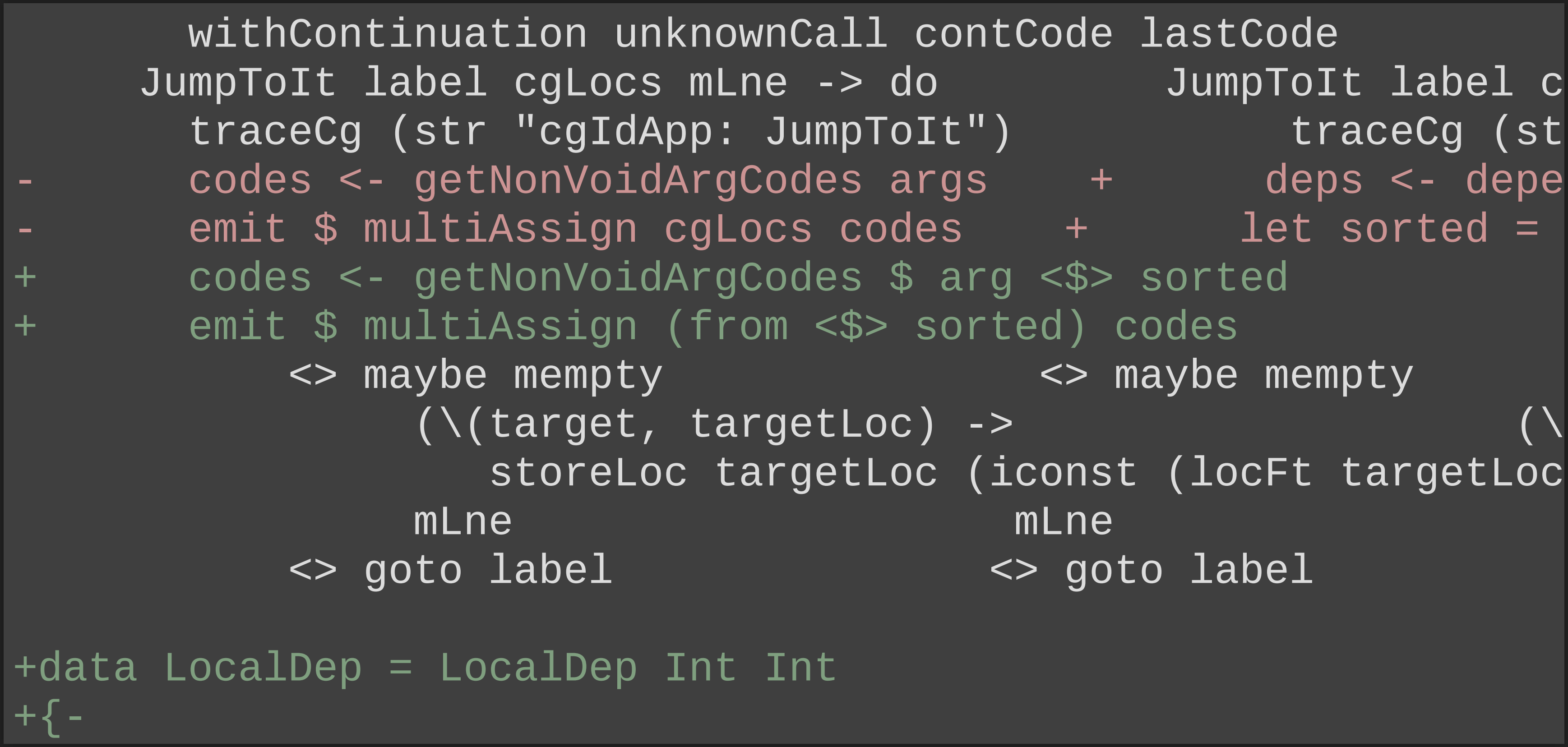
from http://geek-and-poke.com

```
while( n > 0) {
      n = n -1;
      newSum = presum + sum
      presum = sum   // swapped
      sum =  newSum  // swapped
```

# Fix

Fix compiler of Haskell written in Haskell (ghc) while learning haskell.

MOMENT I SAW
CODEGEN MONAD
imgflip.com



```
       withContinuation unknownCall contCode lastCode
     JumpToIt label cgLocs mLne -> do         JumpToIt label c
       traceCg (str "cgIdApp: JumpToIt")          traceCg (st
-      codes <- getNonVoidArgCodes args    +      deps <- depe
-      emit $ multiAssign cgLocs codes    +      let sorted = 
+      codes <- getNonVoidArgCodes $ arg <$> sorted
+      emit $ multiAssign (from <$> sorted) codes
           <> maybe mempty               <> maybe mempty
                (\(target, targetLoc) ->                    (\
                   storeLoc targetLoc (iconst (locFt targetLoc
                mLne                    mLne
           <> goto label               <> goto label

+data LocalDep = LocalDep Int Int
+{-
```

# Decompiled eta. Fixed.

```
while(var8) {
    Main.sat_s7YH var12 = new Main.sat_s7YH(var3);
    var1.R1 = var2;
    Closure var13 = Classes.zeze().enter(var1).apply2(var1, (
    if (!(var13 instanceof False)) {
        return ((Closure)var10).evaluate(var1);
    }

    Main.sat_s7YM var14 = new Main.sat_s7YM(var4, (Closure)va
    Main.sat_s7YL var15 = new Main.sat_s7YL(var3, (Closure)va
    var11 = var10;  //assign presum
    var10 = var14;  //assign new sum
    var9 = var15;     //assign n-1
    var8 = true;
}
```

HOW MUCH ETA STANDS???

```haskell
main = print $ show $  fibtco 100000
```

# TRAMPOLINE

```
import Control.Monad.Trans.Cont

fibCps::Int->Cont r Int
fibCps 0 = return 1
fibCps 1 = return 1
fibCps n = do
      n1 <- fibCps $ n-1
      n2 <- fibCps $ n-2
      return $ n1 + n2


main = do
        let result = trampoline $ runCont  ( fibCps 100 ) id
        putStrLn $ show result
```

# PERFORMANCE

- JMH
- Quick sort implementations exported and called from java
- naive and real quicksort
- compared to same solutions in Java (using vavr.io)
- not very professional - just to get some overview

# Naive quicksort Eta

```
quicksort [] = []
quicksort (x:xs) = quicksort left ++ [x] ++ quicksort right
    where
          left  = [ y | y <- xs, y < x ]
          right = [ y | y <- xs, y >= x ]
```

# Naive quicksort Java/vavr

```
       private List<Integer> qsort(List<Integer> input) {
           if (!input.isEmpty()) {
               final int middle =  input.head();
               final List<Integer> left = input.tail().filter(
               final List<Integer> right = input.tail().filter
               return qsort(left).appendAll(qsort(right).prepe
           } else {
               return input;
           }
       }
```

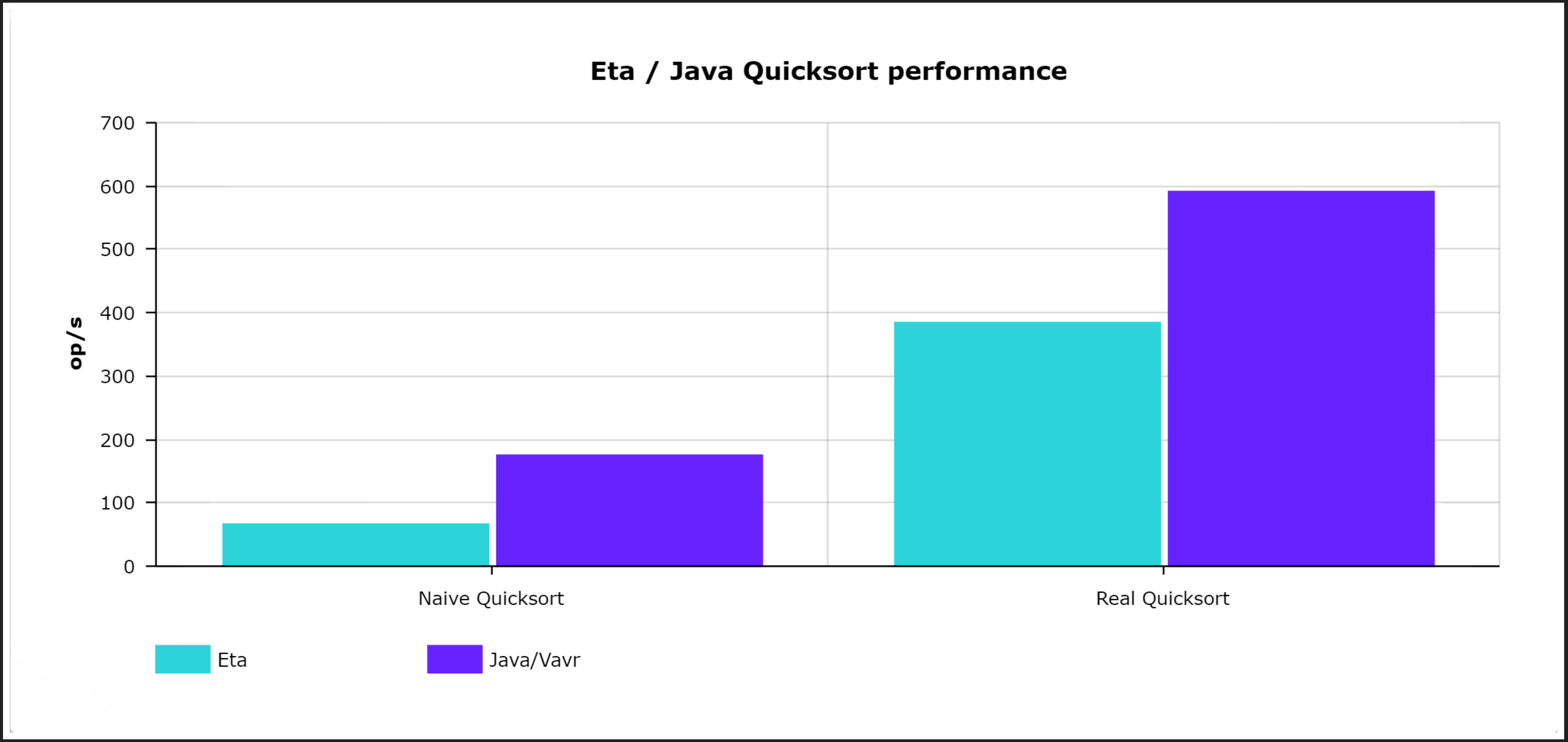
# Real quicksort ETA

```
qvsort :: (G.Vector v a, Ord a) => v a -> v a
qvsort = G.modify go where
    go xs | M.length xs < 2 = return ()
          | otherwise = do
            p <- M.read xs (M.length xs `div` 2)
            j <- M.unstablePartition (< p) xs
            let (l, pr) = M.splitAt j xs
            k <- M.unstablePartition (== p) pr
            go l; go $ M.drop k pr

myvsort ::[Int] ->[Int]
myvsort li =
    let vec = V.fromList li :: (V.Vector Int)
        sorted = qvsort vec :: (V.Vector Int)
        converted = V.toList sorted :: [Int]
```

# Real quicksort Java (*)

```
    list.sort(); // :-)
```

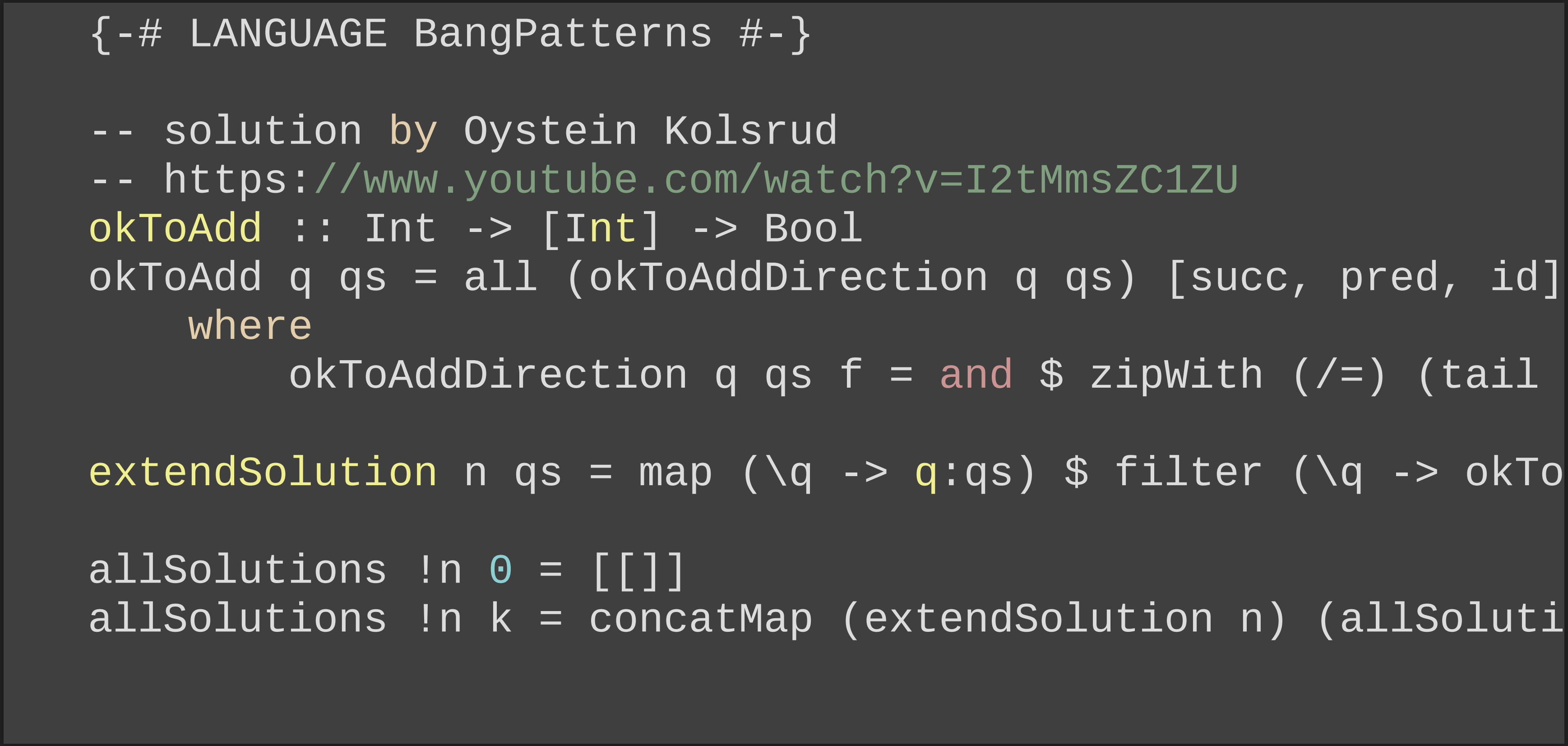
# Results



Eta / Java Quicksort performance
700
600
500
400
300
200
100
0
op/s
Naive Quicksort
Real Quicksort
Eta
Java/Vavr

# VS OTHER *HASKELLS*

12 Queens



```
{-# LANGUAGE BangPatterns #-}

-- solution by Oystein Kolsrud
-- https://www.youtube.com/watch?v=I2tMmsZC1ZU
okToAdd :: Int -> [Int] -> Bool
okToAdd q qs = all (okToAddDirection q qs) [succ, pred, id]
    where
        okToAddDirection q qs f = and $ zipWith (/=) (tail

extendSolution n qs = map (\q -> q:qs) $ filter (\q -> okTo

allSolutions !n 0 = [[]]
allSolutions !n k = concatMap (extendSolution n) (allSoluti
```

| Implementation | Task | Solutions | Time (real) |
|---|---|---|---|
| Frege | 12 Queens | 14200 Solutions | (*)45.816s |
| Eta | 12 Queens | 14200 Solutions | (*)26.472s |
| Ghc | 12 Queens | 14200 Solutions | 9.806s |

Unfair benchmark - both frege and eta were measured with JVM startup time.

# JAVA INTEROPABILITY

# JWT - JAVA TYPES

```
data JColor = JColor @java.awt.Color
  deriving Class
```

# FOREIGN IMPORT

```
foreign import java unsafe "getGreen" getGreen
  :: Java JColor Int
```

# Java is a *Monad*.

```haskell
-- Execute a Java action in the IO monad.
java :: Java c a -> IO a

-- Execute a Java action in the IO monad with respect to the
-- given object.
javaWith :: (Class c) => c -> Java c a -> IO a

-- Execute a Java action in the Java monad of another class
-- with respect to the given object.
(<.>) :: (Class c) => c -> Java c a -> Java b a
withObject :: (Class c) => c -> Java c a -> Java b a

-- Chain Java actions.
(>-) :: (Class b) => Java a b -> Java b c -> Java a c
```

# FOREIGN EXPORT

```
foreign export java "@static eta.example.MyExportedClass.sort"
   sort :: JIntArray -> JIntArray
```

# STYLES OF INTEROPERATIBILIY

# FULL HASKELL WAY

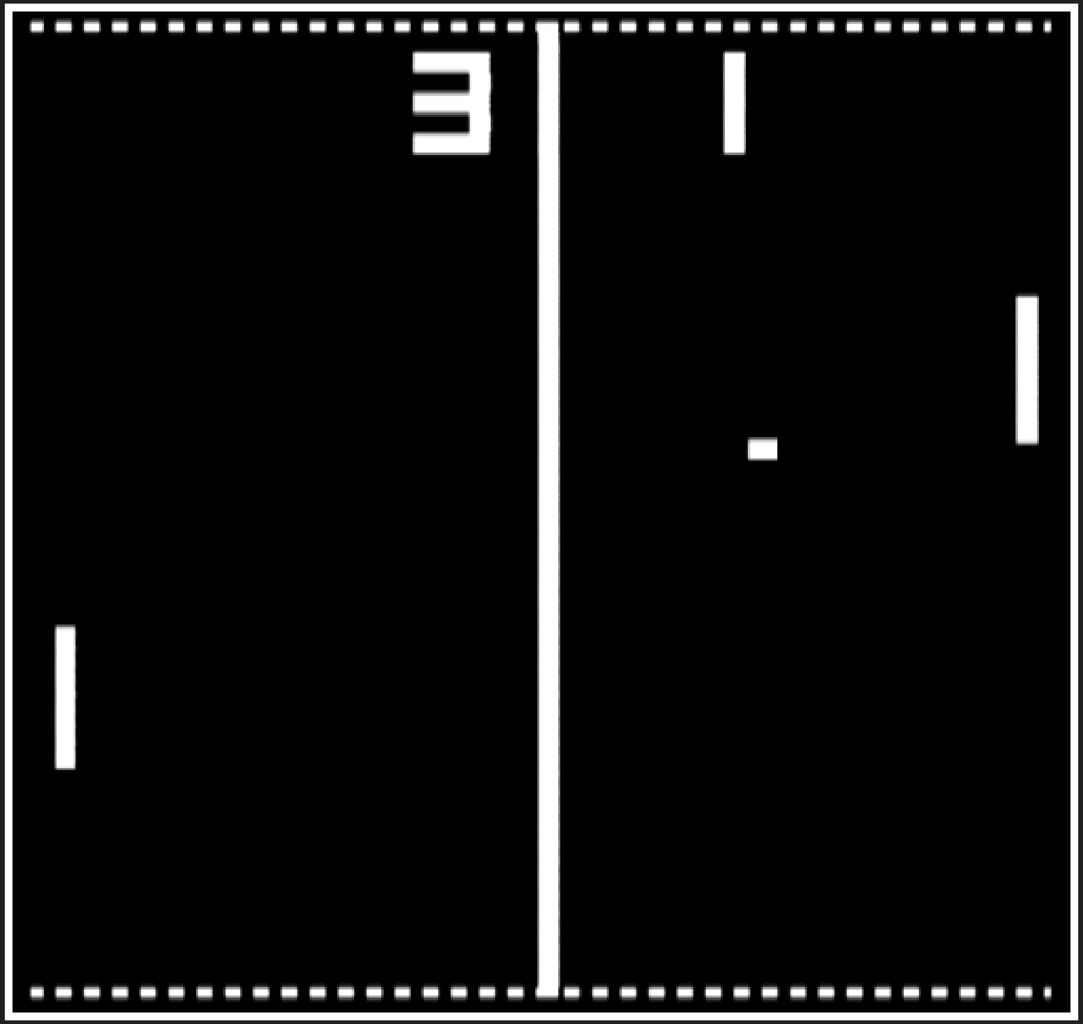
## Example: WAI Servlet

```haskell
appAll :: FilePath -> Application
appAll filePath req respond = case pathInfo req of
  ["state"]         -> appState (unsafePerformIO $ newMVar 0) r
  ["stream"]        -> appStream req respond
  ["request-info"] -> appShowReq req respond
  ["static-file"]  -> appFile filePath req respond
  _                 -> appSimple req respond
```

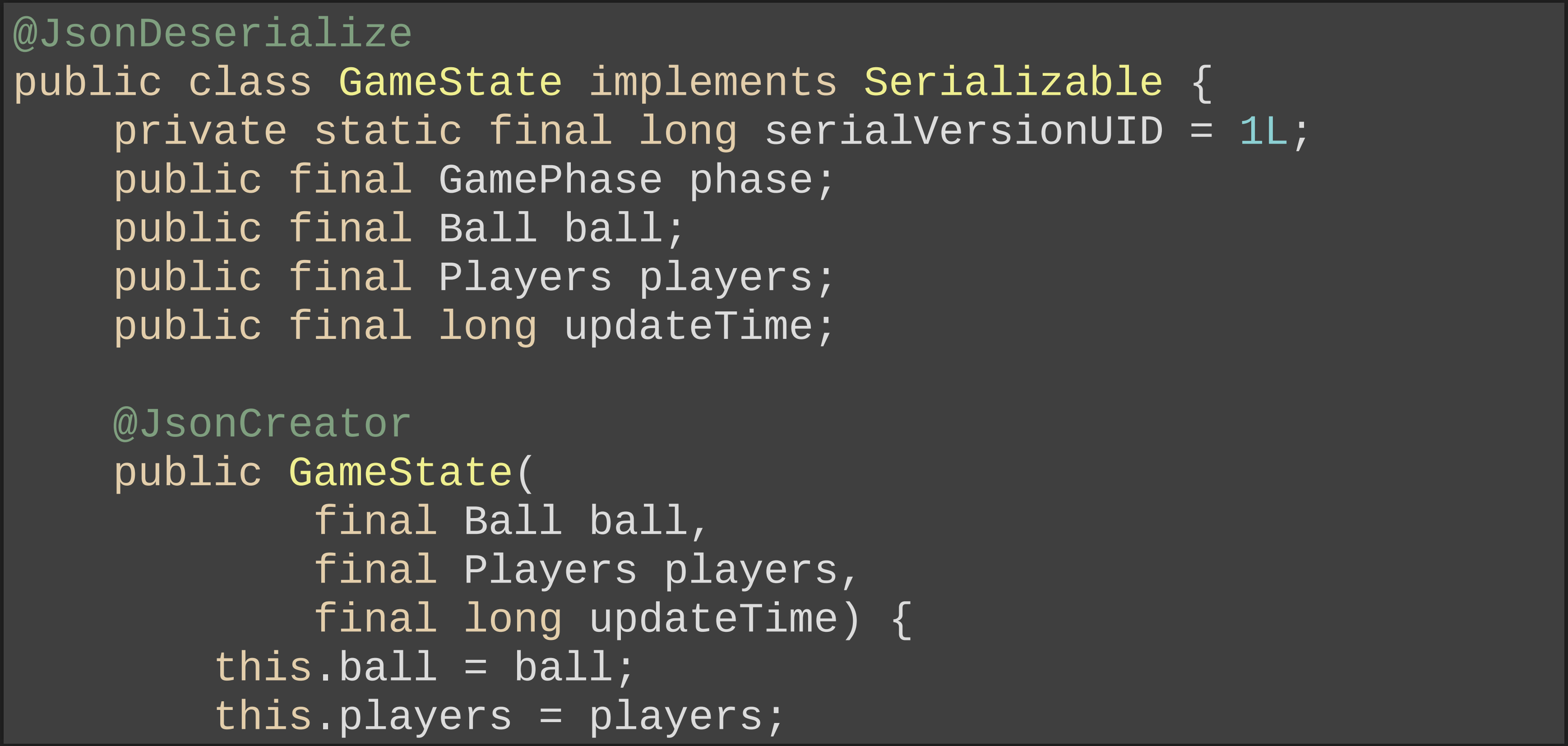
# CLASSES IN JAVA LOGIC IN HASKELL

- Types defined in java
- Haskell functions wok on Java objects
- Support and use of Java frameworks, serializations, db bindings, jsons.

Hint: in 2018a most of java frameworks do not need classical/ ugly *JavaBeans* anymore.


3
1

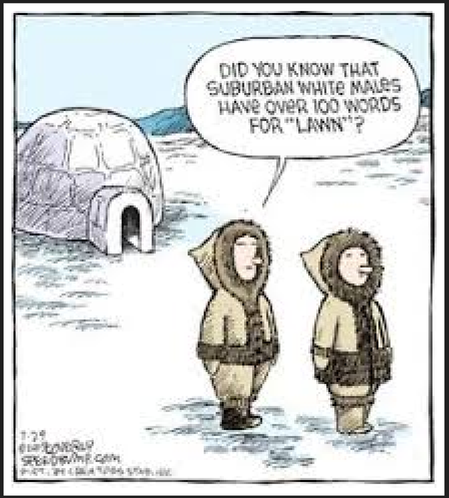
```java
@JsonDeserialize
public class Ball extends GameObject {
    private static final long serialVersionUID = 1L;
    public final Vector2D speed;

    @JsonCreator
    public Ball(float x, float y, Vector2D speed) {
        super(x, y);
        this.speed = speed;
    }
```



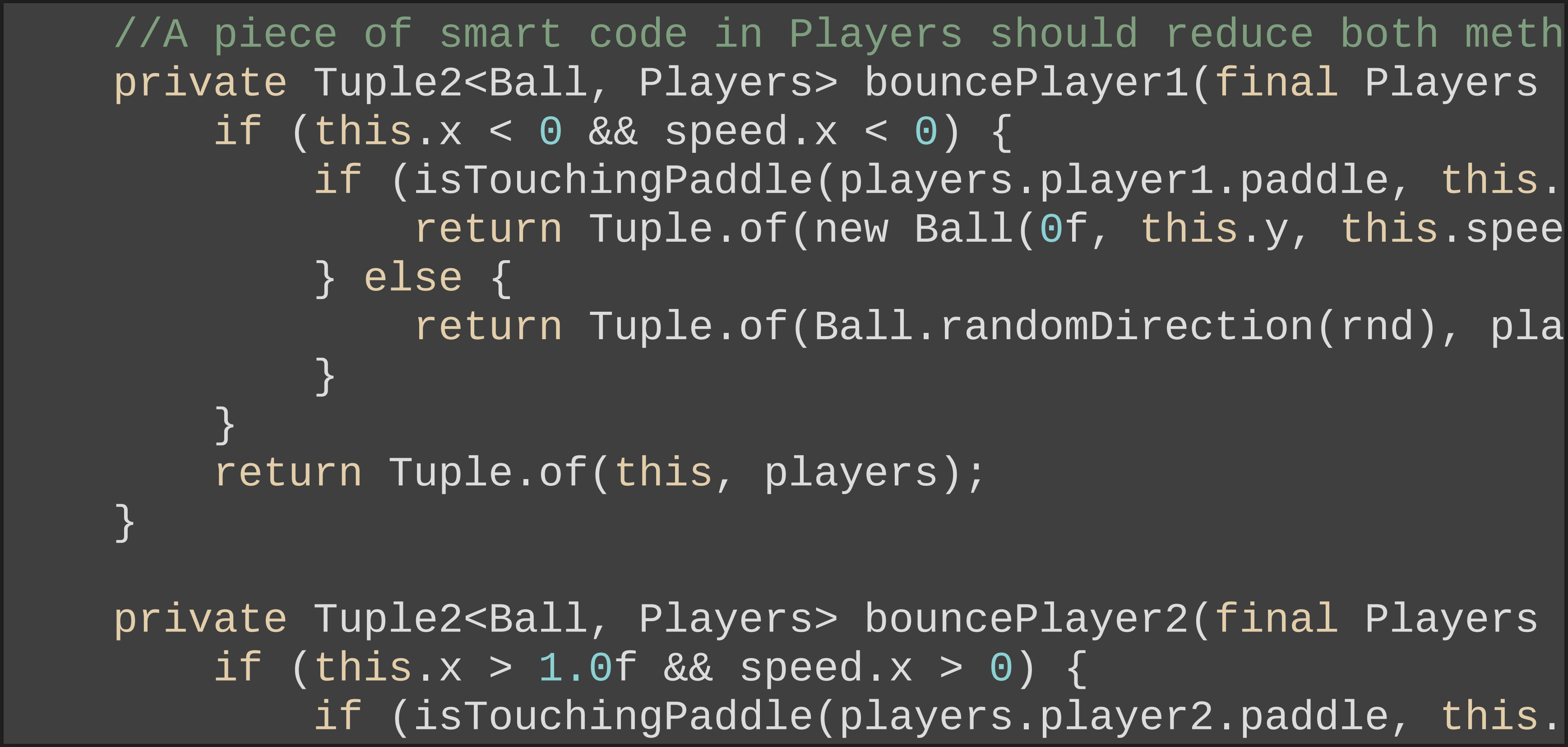
```java
@JsonDeserialize
public class GameState implements Serializable {
    private static final long serialVersionUID = 1L;
    public final GamePhase phase;
    public final Ball ball;
    public final Players players;
    public final long updateTime;

    @JsonCreator
    public GameState(
            final Ball ball,
            final Players players,
            final long updateTime) {
        this.ball = ball;
        this.players = players;
```

```
foreign import java unsafe "@new" newGameState  :: Ball.Ball -

foreign import java unsafe "@field phase" phase :: GameState -

foreign import java unsafe "@field ball" ball :: GameState ->

foreign import java unsafe "@field players" players :: GameSta

foreign import java unsafe "@field updateTime" updateTime :: G

push::GameState->Int64->J.Random->IO GameState
push state time rnd
         | (aPhase == GamePhase.started ) = pushStarted state
         | otherwise  =  return state
         where aPhase = phase state
```

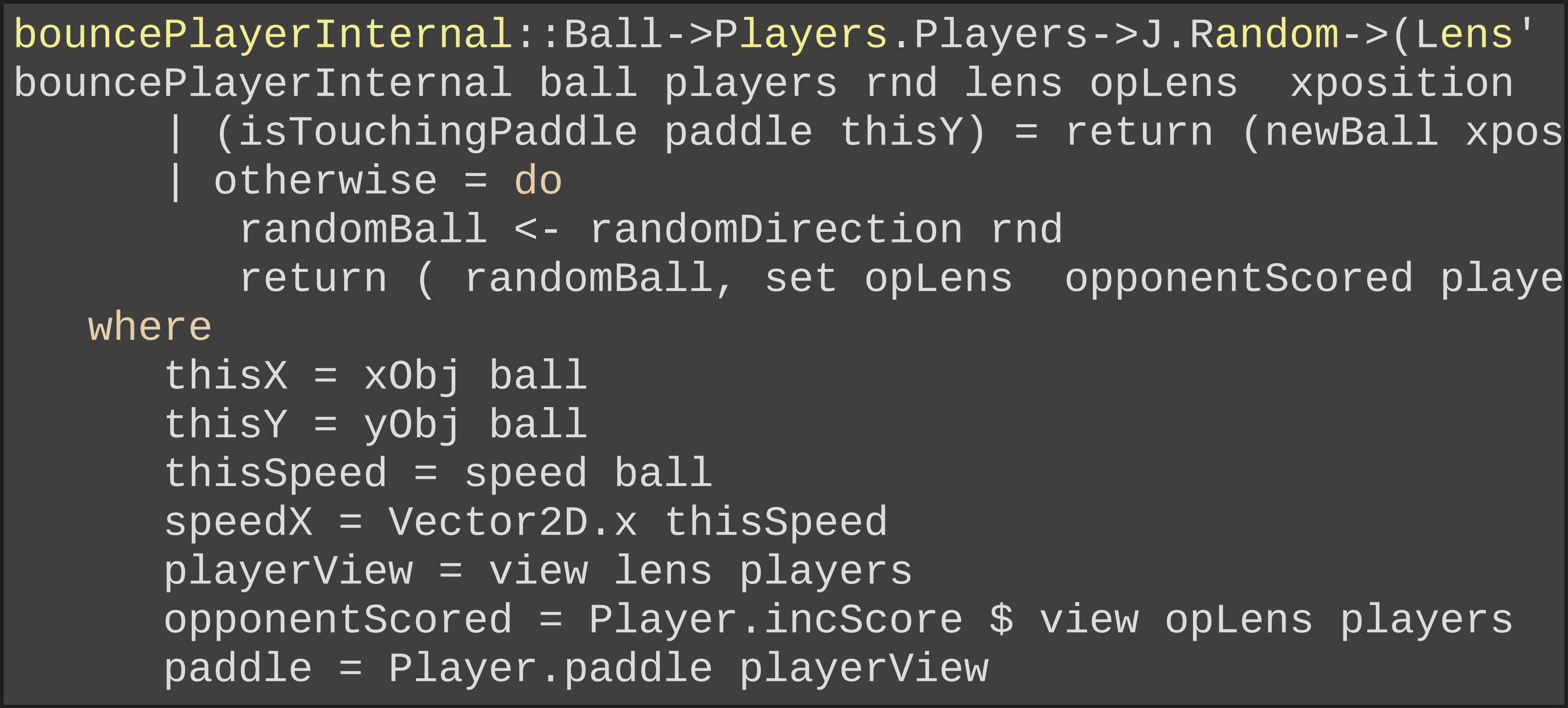
# Linguistic determinism


DID YOU KNOW THAT SUBURBAN WHITE MALES HAVE OVER 100 WORDS FOR "LAWN"?
7-29
SPEEDBUMP.COM

from http://postcogtopics.blogspot.com/2016/



```
    //A piece of smart code in Players should reduce both meth
    private Tuple2<Ball, Players> bouncePlayer1(final Players 
        if (this.x < 0 && speed.x < 0) {
            if (isTouchingPaddle(players.player1.paddle, this.
                return Tuple.of(new Ball(0f, this.y, this.spee
            } else {
                return Tuple.of(Ball.randomDirection(rnd), pla
            }
        }
        return Tuple.of(this, players);
    }

    private Tuple2<Ball, Players> bouncePlayer2(final Players 
        if (this.x > 1.0f && speed.x > 0) {
            if (isTouchingPaddle(players.player2.paddle, this.
```



```haskell
bouncePlayerInternal::Ball->Players.Players->J.Random->(Lens'
bouncePlayerInternal ball players rnd lens opLens  xposition
      | (isTouchingPaddle paddle thisY) = return (newBall xpos
      | otherwise = do
         randomBall <- randomDirection rnd
         return ( randomBall, set opLens  opponentScored playe
   where
      thisX = xObj ball
      thisY = yObj ball
      thisSpeed = speed ball
      speedX = Vector2D.x thisSpeed
      playerView = view lens players
      opponentScored = Player.incScore $ view opLens players
      paddle = Player.paddle playerView
```

# HAVA

```
ballBounceP :: Ball.Ball ->
Players.Players -> J.Random -> IO
Players.Players
```

I SMELL
JAVA
imgflip.com

# POINTER REF WAY

Data in haskell, businell logic in haskell. Java as Controller.

We need to expose haskell *objects* to java.

# Game of life

```
data Color = Color  {red :: Int,
                                                             g
                                                             b


data  Cell  = Dead | Alive {color :: Color}

type Row = Array Int Cell
type Plane = Array Int Row

type GOLState = StablePtr Plane

initEmptyXP:: Int -> Int -> IO GOLState
initEmptyXP wi hi = newStablePtr $ makePlane wi hi

newStateXP::GOLState -> IO GOLState
```

```
    public static int newState(int var0) {
        return ((StablePtr)Runtime.evalIO(new Ap2Upd(TopHandle
    }
```

# PROBLEMS

- lot of imports to write for every simple java class
  - this will be fixed thanks to ffi tool
- it took me a while to find out how to pass state between haskell and java
- other bug found (and resolved)
- java monad / io monad - not totally intuitive (for a newbie)

# ETA VS FREGE

I used Frege very shortly.

- **Frege** is more mature
- Interoperation with Java is easier with Frege
- Frege will not be close GHC in the near future
  - at the semantics level
  - at the base libraries level

# ETA FOR YOU

# ETA NOW

Eta is 0.7.0b2 is not production quality yet

If You think of eta in production soon -> talk to **Typelead**.

They want to provide commercial support - ask them for conditions.

If you are haskell developer that wants to evaluate haskell on JVM

*Try it now!*

If you are JVM / JavaDeveloper that wants to learn and play with Haskell

*Play now!*

# ETA COMMUNITY

Small.

Great!

You can help! There are lot of small things to do.

Future of eta lies in your hands