# The Concurnas Programming Language

- Jason Tatton. Founder Concurnas Ltd.

# Jason Tatton

Creator and Founder of Concurnas Ltd


jason.tatton@concurnas.com

concurnas.com

discord.gg/jFHfsqR

@concurnas

linkedin.com/company/concurnas


- Coding since age 9
- 10 years experience in building automated trading systems
- Created Concurnas in 2017

# Concurnas

A JVM Programming Language!

Easy to Learn

*Pythonic Syntax*

Easy to Scale

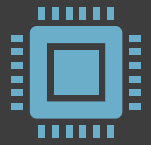
*Use the same code on a laptop, server or the cloud*

High Performance

*Runs on the JVM*

Multi Paradigm, multi Domain

*Enterprise, scientific computing, finance*



Supporting modern concurrency

*Multi-core GPU and GPU hardware*

Open Source

*MIT license*

# Hello World!

```
def gcd(x int){
  y = 3
  while(y){
    (x, y) = (y, x mod y)
  }
  x
}

for(x in 101 to 105){
  System.out.println("hello world {x} => {gcd(x)}")!
}
```

- Functions may exist in isolation
- Function return type is implicit
- Type of `y` is inferred
- All types can be used in `Boolean` expressions
- Tuples are supported
- `return` statement is implicit
- Numerical range expression
- `!` creates a concurrent 'isolate' (light-weight thread)
- String formatting
- Utilizing the JDK

Output:

```
hello world 102 => 3
hello world 104 => 1
hello world 101 => 1
hello world 105 => 3
hello world 103 => 1
```

Your output may vary!

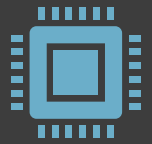
# Modern engineering problems?

Performance

*Most code needs to be fast*

Productivity

*Developer performance*



Hardware architecture

*Multi-core CPU and GPU's*

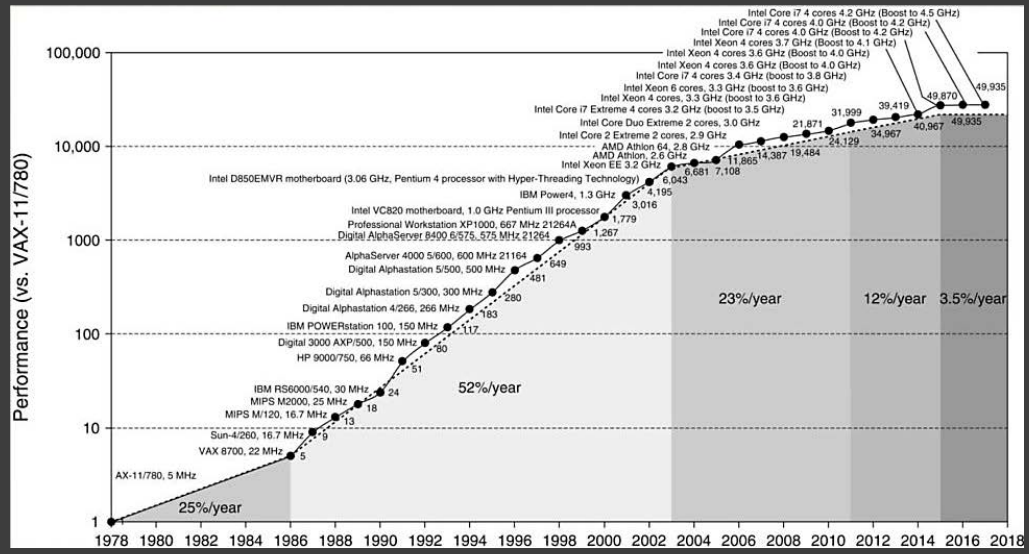
Modern problems

*Reactive Systems, AI, ML*

Heterogenous teams

*Not everyone is a software engineer*

CPU's aren't getting quicker!



# Why a new Language?

- Concurrent programming is really hard
  *Thread and lock with shared mutable state is hard*
- Why not library based solutions?
  *Work around the core issue*
- Why not augment an existing language?
  *Works ok but limits options*
- Why not modify an existing language?
  *Good luck!*
- New langages let us embrace modern techniques
  *Such as: improved object oriented model, null safety, vectorization etc*

# Concurnas Goals

- First class citizen support for concurrency
  *A new simplified model on the CPU, on the GPU*
- Easy to learn, productive and high performance
  *Familiar syntax, with optimized machine code*
- Runs on the JVM
  *Garbage Collection, use existing Java code (and the JDK)*
- Optionally lightweight syntax
  *Some code should be verbose!*
- Dynamically typed syntax with type safety
  *Like python but with Java scale*
- Principle of least surprise
  *Concurnas code needs to look familiar*
- Scale with minimal effort
  *From prototype to production to the cloud*
- Large data sets
  *More data than can fit in RAM*
- Modern software engineering practices
  *null safety, better object orientation, pattern matching*
- Extensible
  *Embed other languages in Concurnas code*
- Domain Specific Language support
  *Solving domain problems*
- Around for a long time
  *The next 30 years!*

# Time for some code!

# Expressions & Variables

```
π = Math.PI      //unicode
πString = "π=" + π //π=3.141592653589793
/* ~~~ multiline comment ~~~ */
10, 9., 10e5, 10.0f, 345345L, true, 'c' //normal constants

[1 2 3 4]    //an integer array
[1, 2, 3, 4] //an integer list
[1 2 ; 3 4]  //an integer matrix definition

2+2**4 mod 5              //expression
a == 3 or b < 9           //boolean expression
2 if something() else 15 //if expression
```

```
anint = 8
avar int = 99          //explicit type
var reassignOk = 12   //new variable
val nonReassign = 99 //new variable cannot reassign
```

# Control flow

```
for(a in [1 2 3]){
 processIt(a)
} //iterator for

for(a = 1; a <== 3; a++) {
 processIt(a)
} //c style for

while(xyz()){
 doSomething()
}

loop{ //while(true)
 if(doSomething()){
  break
 }
}
```

```
while(xyz()){
 doSomething()
}else{ //if while loop never entered
 onfail()
}
```

```
if(a){
 doThis()
}elif(b){
 doAnother()
}else{
 another()
}
```

# All blocks can return

```
speed = {
 distance = 100
 time = 24.
 distance/time
}// all bracketed blocks return
```

```
astring = if(condition()){ "value1" } else{ "value2" }
```

```
pows = for(x in 2 to 6){
 2**x
}
//pows == [4, 8, 16, 32, 64]
```

# Functions - Compact Syntax

```
//an ordinary function:
def adder(a int, b int) int {
 return a + b
}

//we can use => to compact the function to:
def adder(a int, b int) int => return a + b

//infer the return type:
def adder(a int, b int) => return a + b

//implicit return expression - most compact form!
def adder(a int, b int) => a + b
```

```
//calling a function
adder(8, 7) // == 15
```

# Functions

```
def adder(a int, b int)   => a + b
def adder(a int, b float) => a + b
def adder(a int) => adder(a, 10)//overloading
```

```
def manyAdder(a int, nums int...) => for(n in nums){n + a}

//varargs called as:
manyAdder(10, 1, 2, 3) // returns: [11, 12, 13]
```

```
def wdefaults(a=10, b=2, c=3) => a + b * c //default args

wdefaults()     //== 16

wdefaults(c=10) //named parameter == 30
```

# Exceptions

```
class ArgumentException(msg String) < Exception(msg)

def process(a int) int {
 if(a < 2){
  throw new ArgumentException("a is smaller than 2")
 }
 a ** 2
}

result = try{
 process(1)
}catch(e ArgumentException){
 0 //react as appropriate
}catch(e){
 throw e //re-throw
} finally{
 afterProcCall() //always called
}
```

# Object Oriented Programming

# Basic Objects

```
class Person(~name String, ~surname String){
 this(surname String) => this('dave', surname )
 likes = java.util.HashSet<String>()
 def addLike(like String) boolean => likes.add(like)
}
```

```
p1 = new Person(‘talyor’)
p2 = Person('amber', 'smith')
```

```
p1.addLike('sprouts')
oldname = p1.name //same as: oldname =p1.getName()
p1.name = "jon"      //same as: p1.setName("jon")
```

```
p3 = p1@ //copy operator

assert p1 == p3 //== p1.equals(p3)
assert p1 &<> p2//p1 <> p3

people = set()
people..add(p1).add(p3)
assert people.size() == 1//one item stored by value
```

# Traits & Generics

```
abstract class AbstractFooClass{
 def foo() => "version AbstractFooClass"
}

trait A{ def foo() => "version A" }
trait B{ def foo() => "version B" }

class FooClass extends AbstractFooClass with B, A{
 override def foo() => "" + [super[AbstractFooClass].foo(), super[A].foo(), super[B].foo()]
}

FooClass().foo() //returns [version AbstractFooClass, version A, version B]
```

```
class Pair<X, Y>(-x X, -y Y) //generic class

p1 = Pair<String, int>("one", 1)
p2 = Pair("name", "another")
```

# Objects - Others

```
enum Food{Tomatoes, Beans, Bread, Grapes, Pizza}
```

```
annotation MapsTo{//a custom annotation
 name String
 mapTo String
 repeat = 1//annotation field default value
}
class MyClass2{
 @MapsTo(name = "mappingName", mapTo = "anotherName")
 afield int = 99//afield is annotated with MapsTo
}
```

```
class Robot{
 -path = ""
 private def add(dir String) => path +=dir
 def up() => add("U")
 def down() => add("D")
 def left() => add("L")
 def right() => add("R")
}

result = with(Robot()){
 up(); up(); left(); down(); right()
 path
}//result == UULDR
```

# Null Safe

```
aString String = "something"
aString = null //compilation error, aString is not of a nullable type.
```

```
aString String? = "something"
aString = null //this is ok
```

```
len = aString.length()       // compilation error
```

```
len = aString?.length()      // ok - null handled
```

```
len = (aString?:"").length() // ok - null handled
```

```
len = aString??.length()     // ok - null handled (sort of)
```

```
len = if(null == aString){
 -1
}else{
 aString.length() // ok - cannot be null
}
```

# Functional Programming

# Function references & Lambdas

```
def plus(a int, b int) => a + b
op2 (int, int) int = plus&

result = op2(10, 1)
```

```
op (int) int = plus&(10, int)

result = op(1)
```

```
def toEach(opon int[], func (int) int) {
 for(o in opon) {
  func(o)
 }
}

toEach([1 2 3], op)
```

```
toEach([1 2 3], a => a+10) //lambda definition
```

# Pattern Matching

```
class Person(-yearOfBirth int)

def matcher(an Object){
 match(an){
  Person(yearOfBirth < 1970) => "Person. Born: {an.yearOfBirth}"
  Person     => "A Person"
  int; < 10 => "small number"
  int        => "another number"
  x          => "unknown input"
 }
}

res = matcher(x) for x in [Person(1829), Person(2010), "oops", 43, 5]
//res == [Person. Born: 1829, A Person, unknown input, another number, small number]
```

# Concurrency!

# Isolates

```
def gcd(x int){
  y = 25
  while(y){
    (x, y) = (y, x mod y)
  }
  x
}

res1 int: = gcd(92312)!     // int: is a ref type
res2      = {gcd(2438210)}! //complex calculation here

larger    = res1 if res1 > res2 else res2
```

```
await(res1)
```

```
n = 10
nplusone = { n += 1; n }!
nminusone = { n -= 1; n }!

assert n == 10
assert nplusone == 11
assert nminusone == 9
```

# Reactive Computing

```
asset1price int:;
asset2price int:;

every(asset1price, asset2price){
 if(asset1price > asset2price){
  //... initiate trading action here!
  return//terminate future invocation of the every block
 }
}
```

```
a int:
b int:

c = every(a, b){ a + b }

every(c){
 System.out.println("latest sum: {c}")
}
```

```
c <= a + b
```

# Actors

```
actor IdGenerator(prefix String){
 cnt = 0//implicit private state
 def getNextId(){
  toReturn = prefix + "-" + cnt
  cnt += 1
  toReturn
 }
}

idGen = IdGenerator("IDX")//create an actor
anId1 = idGen.getNextId()//==> IDX-0
anId2 = idGen.getNextId()//==> IDX-1
```

```
setService = actor java.util.HashSet<int>()
setService.add(65)
```

# Distributed Computing

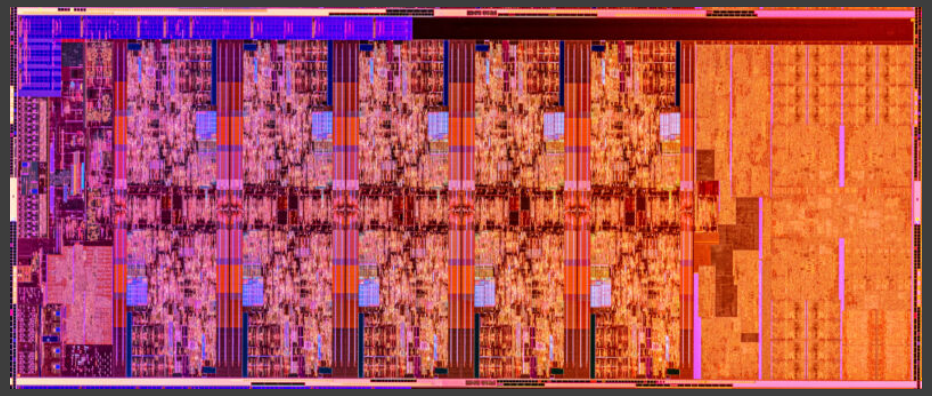
```
//A remote server:
remServer = new com.concurnas.lang.dist.RemoteServer(port = 42001)
remServer.startServer()
//wait until time to terminate...
remServer.close()
```

```
//A client:
rm = Remote('localhost', port = 42001)
//execute code remotely, returning a ref:
ans int: = {10+10}!(rm.onfailRetry())
rm.close()
//ans == 20
```

# GPU Computing

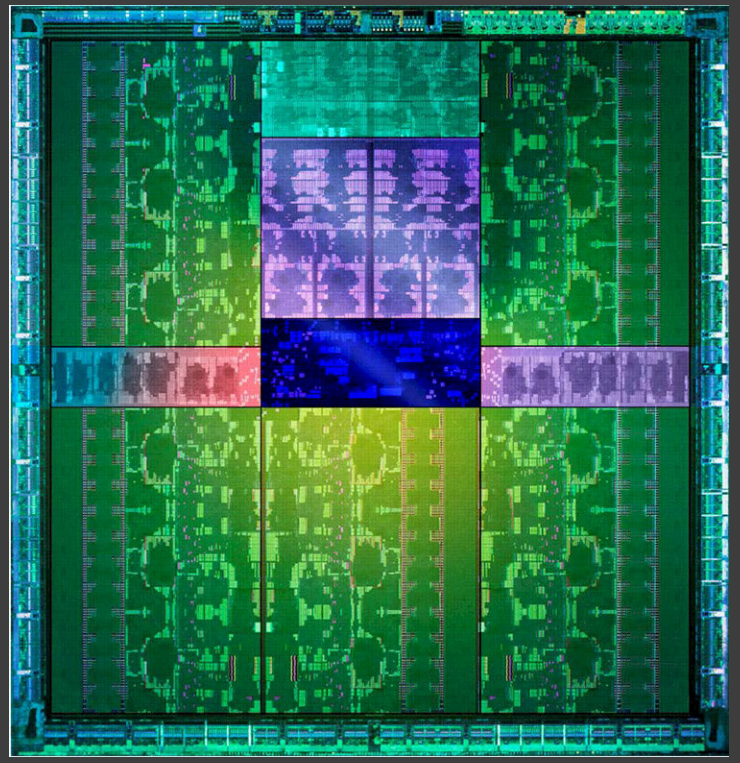
# Motivation

Modern CPU Die



*64 cores*

Modern GPU Die



*4,000+ cores*

# Utilizing the GPU

```
gpukernel 1 twoArrayOp(global in A float[], global in B float[], global out result float[])
{
 idx = get_global_id(0)
 result[idx] = A[idx]**2 + B[idx] + 10
}
```

```
//select a GPU device...
device = gpus.GPU().getGPUDevices()[0].devices[0]

//we create three arrays of size 10 on this GPU, 2 as input
inGPU1 = device.makeOffHeapArrayIn(float[].class, 10)
inGPU2 = device.makeOffHeapArrayIn(float[].class, 10)
result = device.makeOffHeapArrayOut(float[].class, 10) //1 output

//now we write to the arrays on the GPU
c1 := inGPU1.writeToBuffer([ 1.f 2 3 4 5 6 7 8 9 10 ])
c2 := inGPU2.writeToBuffer([ 1.f 2 1 2 3 1 2 1 2 1 ])
```

```
inst = twoArrayOp(inGPU1, inGPU2, result)
compute := device.exe(inst, [10], c1, c2)//run 10 cores to process
ret = result.readFromBuffer(compute)
```

# Working with Data

# Working with Data

```
anArray = [1 2 3 4 5 6]
aList = [1,2,3,4,5,6]
aMatrix = [1 2 3 ; 4 5 6]
aMap = {"one" -> 1, "two" -> 2, "three" -> 3}

cont = "one" in aMap //checking for a value in a map
del aMap["one"]      //remove element from aMap
arrayValue = anArray[2] //individual value from array
arrayValue = aMatrix[0,1] //individual value from matrix
subarray = anArray[4 ...] //a sub array; [5 6]
```

```
longNames = aMap[key] for key in aMap if key.length() > 3
ret = i+10 for i in aList if i mod 2 == 0
```

```
def getDetails() => ("dave", 27) //returns a tuple

(name, age) = getDetails() //tuple decomposition
```

```
reversed(enumerate(zip([1,2,3], [4,5,6])))
// [(2, (3, 6)), (1, (2, 5)), (0, (1, 4))]
```

# Vectorization & ranges

```
mat = [1 2 ; 3 4]

mat2 = mat^*2 + 1 //==> [3 5 ; 7 9]

mat^^*2 + 1       //in-place vectorized operation.

mat3 = mat + 1    //implicit vectorization
```

```
numRange = 0 to 10         //a range of: [0, ..., 10]
tepRange = 0 to 10 step 2 //a range of: [0, 2, ..., 10]
revRange = tepRange reversed //a reversed range of: [10, 8, ..., 0]
decRange = 10 to 0 step 2 //a range of: [10, 8, ..., 0]
infRange = 0 to           //an infinite sequence [0,... ]
steInfRa = 0 to step 2    //a stepped infinite sequence [0, 2,... ]
decInfRa = 0 to step (-1) //a stepped infinitely decreasing sequence [0, -1,... ]

val = x for x in numRange //list comprehension over a range
check = 2 in numRange     //checking for the presence of a value
```

# Other Languages

```
from com.mycompany.myproduct.langs using mylisp, myFortran, myAPL
calc = mylisp||(+ 1 2 (* 2 3))|| // == 9

myFortran|| program hello print *, "Hello World!" end program hello|| //prints "Hello World!"

lotto = myAPL || x[⍋x←6?40] || //6 unique random numbers from 1 to 40
```

```
from com.mycompany.myproduct.langs using mylisp, mySQL, myAPL
class Person(name String, yearOfBirth int)
people list<Person>;

millennials = mySQL||select name from people where yearOfBirth between 1980 and 2000||

myAPL||fact{×/⍳⍵}||
fact(10) //use of function defined in myAPL. returns: 3628800
```

```
from com.mycompany.myproduct.langs using mylisp
aString = "i'm a String!“

invalidCode = mylisp||(+ 1 2 (* 2 aString))|| //results in compilation time error
moreInvalidCode = mylisp||(+ 1 2 (* 2 3)||    //oops! Missing a closing ')'
```

# That’s most of the code!

# Performance

- On a par with Java
  *~5% overhead from stack rolling and unrolling*
- Isolates scale better
  *Than conventional threads*
  *Easier to reason about*
- Developer productivity
  *Concurnas requires less code to do more*
  *Null safety leads to better code*
- The JVM is always improving
  *Concurnas gets incremental upgrades for free*

# Future Developments – 2020!

- Better development tools
  *IDE (including debugger)*
  *Documentation generator*
  *Code formatter*
  *Gradle build plugin*
- Better Off Heap Memory Management
  *Value types*
  *Better language support*
- Faster Compiler
  *Runtime is really quick*
  *Compiler needs some work*
- Better GPU Computing
  *Objects*
  *Simplified GPU computing*
  *Cleaner interface*
- Automatic Differentiation
  *For Finance*
  *For Machine Learning*

# Concurnas Ltd.

## Commercial support

*For Concurnas*
*Packages to suit all sizes of client*

## Consultancy

*Language development*
*Financial services, enterprise computing*

## Sponsored development

*Have a feature you'd like to see in Concurnas?*
*Get in touch!*

# Further Information

- Concurnas website
  concurnas.com
  *Documentation*
  *Articles*
  *Videos*

- Download
  concurnas.com/download.html
  *Or use: SKDMAN! -* sdkman.io

```
sdk install concurnas
```

- Join us on discord
  discord.gg/jFHfsqR
  *Help and support*
  *Ideas and chat*

- Contribute
  github.com/Concurnas/Concurnas

# Questions?