The Dining Philosophers

Provide an online and offline venue for
software professionals to network
with each other on a professional and social basis.

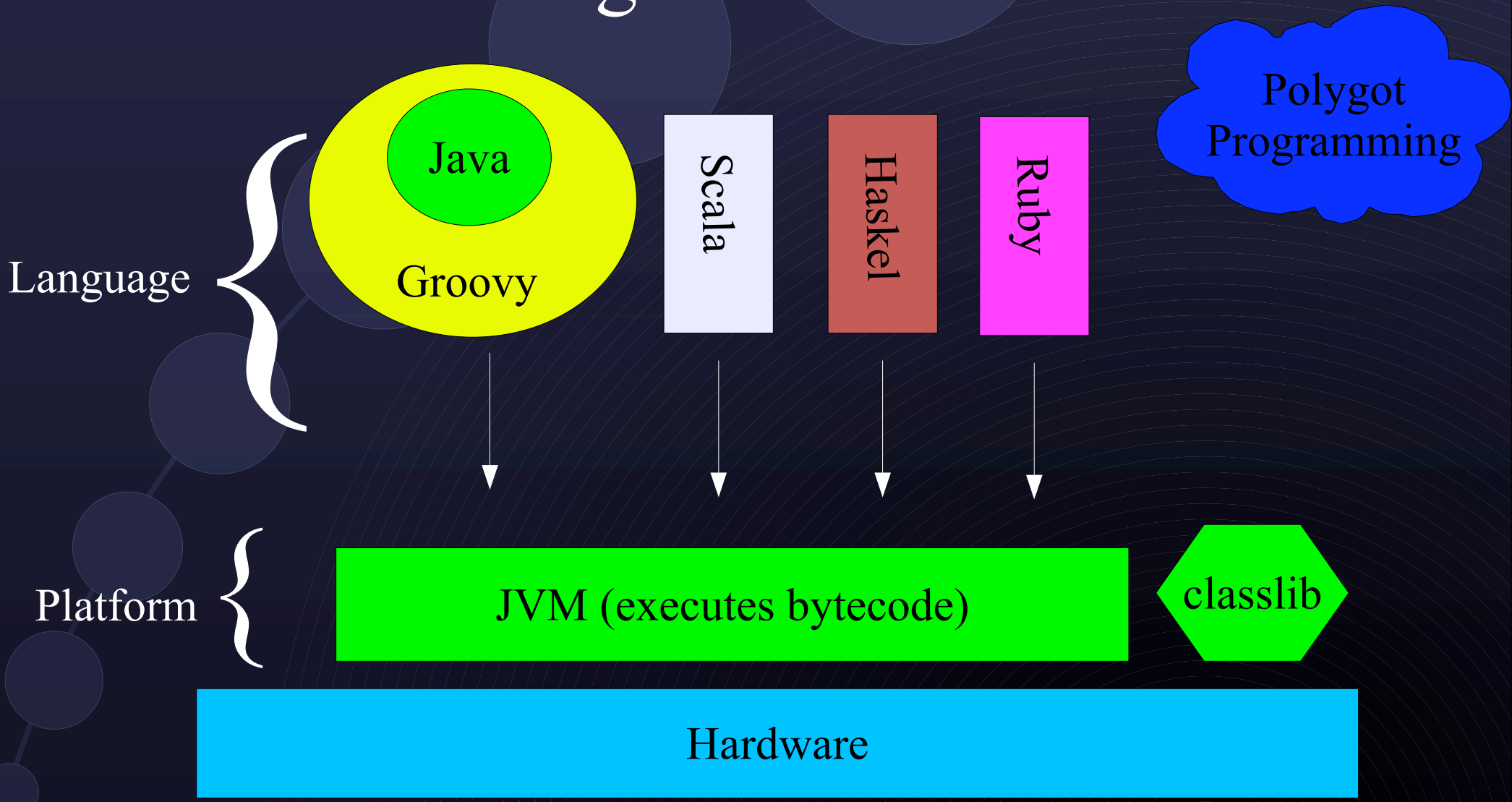
Started in Shanghai 2006

- Tech talks
- Networking events
- http://www.diningphilosophers.net

About Presenter

- Programming computers since age 10
- 10+ years of experience in computer industry
  - Unix System administration
  - Software development
- Graduated from Wharton, University of Penn 沃顿，宾夕法尼亚大学
- Enjoys learning languages(human and computer)

Groovy
The next generation of Java



Grails

- Groovy/Java web framework
  - Underneath uses Spring, Hibernate, and Sitemesh
  - Borrows many ideas Ruby on Rails
    - Convention over configuration
    - Different configuration for different environment
      - Development
      - Testing
      - Production
    - Scaffolding
    - ActiveRecord
    - Custom tag libs
    - Extensive plugins
    - Do not need to restart web server when making changes

- Live Grails demo with IntelliJ in 20minutes
  - A blog

Dynamic vs Static typing

- Statically typed languages (for example: Java, C++)
  - Compiler checks all the type at compile time
  - Certain class of errors can be caught automatically at compile time

- Dynamically typed languages(for example: Python, Ruby)
  - The runtime checks the type at runtime
  - The type of a variable can change
    - a = 1
      a=”abc”

- Which is better? 费力

- Groovy is both dynamically and statically typed

“Ceremony vs Essence”

- Neal Ford

Haskel:

```
main = print("Hello World")
```

Java:

```
public HelloWorld{
    public static void main(String[] argv){
        System.out.println(“Hello World!”);
    }
}
```

Groovy:

```
println “Hello World”
```

Less is more ...

PrintFile.java:

```
import java.io.*;
class PrintFile{
    public static void main(String[] args){
        try {
            FileReader reader = new FileReader(args[0]);
            BufferedReader bReader = new BufferedReader(reader);
            String line = null;
            while((line = bReader.readLine())!=null){
                System.out.println(line);
            }
        }catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

PrintFile.groovy:

```
def fileName = args[0]
new File(fileName).eachLine{ println it}
```

Less is more ...

But not always

Perl

```
print pack"C*",split/\D+/,`echo "16iII*o\U@{$/=$z;[(pop,pop,unpack"H*",<> )]}\
EsMsKsN0[lN*11K[d2%Sa2/d0<X+d*lMLa^*lN%0]dsXx++lMlN/dsM0<J]dsJxp"|
dc`
```

Outline

1. Function
2. Closures
3. Classes
4. Interfaces
5. Control structures
6. Data types: arrays, lists, ranges, maps, GString
7. Comparison with Java
8. Builders
9. Processing XML
10. DSL
    1. XML
    - Swing

Function

•Java only has free standing nouns (名词) via classes but no free standing verbs(动词).

- Java methods/verbs are trapped inside of classes

•functions/methods are independent of classes in Groovy but functions are not first class types.

```
def sayHello(name){
    “hello “+name
}
```

•Return types , parameter types are optional
•Last statement evaluated is returned

Closure

- Closures are free standing verbs like functions
- Closures can be passed around as parameters

```
def c = { “hello world” }
c()

def c2 = {name-> “hello $name”}
c2(“world”)
```

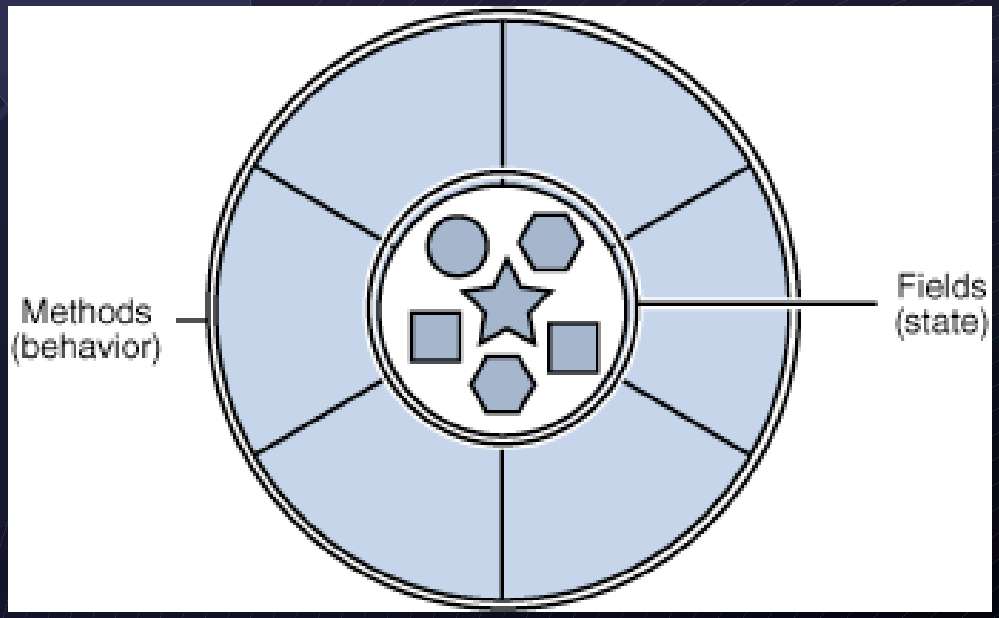
Closure

- Closures have an environment closed over it
  - Hence the name
  - Variables in that environment survive invocation of the closure
  - Think of closures as functions with bounded variables

```
def foo(){
    int i = 0
    return { i++}
}

x = foo()
println "x="+x()

y = foo()
println "y="+y()

println "x="+x()
```

Closure vs Objects

- Object is data/state with method wrapped around it
- Closure are methods with data/state wrapped around it



Functions vs Closures

Groovy is Java

- 99% of Java language is valid Groovy code.
  - Rename .java to .groovy will work 99% of the time

- Unlike other languages that run on the JVM (Ruby,Python) Groovy objects map 1-to-1 to Java objects

Class

```
class Employee{
    String lastName
    def firstName
    int age
}
```

Creating new object:

- def e = new Employee()

- named parameters:

```
Employee e2 = new Employee(lastName: “Su”,
                           firstName: “Zhong Shan”)
```

Class

- Similar to Java classes
- Fields are by default private
- Getter/Setter automatically generated for fields

  def e = new Employee(firstName:"john", lastName: "doe")

  e.firstName ==> e.getFirstName()

  e.firstName = "jane" ==> e.setFirstName("jane")

Interface

- Groovy interfaces are same as Java interfaces
- Groovy way of implementing interfaces
  - Interface with single method
  - Interface with multiple method using a map

Data structures

- java.util.List
  - def family = ['mom','dad','brother','sister','uncle']
    assert family instanceof List == true
  - << append operator
    - family<<"son"
  - *. operator
    - family*.toUpperCase()
- java.util.Map
  - def ages = ['mom':50, dad: 58, sister: 20, uncle: 50]
    assert ages instanceof Map == true
- Range (java.util.List)
  def r = 0..5
  def r = 0..<5
- GString
  - Can contain arbitary expresions
  - "The current time is ${new Date()}"
  - Can span multi-lines

Loops

- Groovy only supports Java's while and for loops.
- For
  - Like Java:

```
for (int i = 0; i < 5; ++i)
        println i
```

  - Simplier and more powerful than Java's for loop
    - Works with arrays, maps, list and collection

```
for( i in 0..5)
        println i

def people = ['mom','dad','brother','sister']
for(p in people)
        println p
```

Loops

```
def  ages = ['john':  10, jane: 20, 'mike': 30]
for(a in ages){
      println a
}

for(s in 'abcdefghijklmnopqrstuvwxyz'){
      println s
}
```

Processing XML

- XmlSluper
- GPath
  - Similar to Xpath but used to navigate object graph

Builders

- Useful for building tree structure
  - HTML/XML
  - Swing GUI
- XML builder
  - demo
- Swing Builder
  - demo

Meta programming

- Java's reflection API (java.lang.reflect.*) too complex
- Every Groovy class has a Metaclass
  - Query
    String.metaClass.methods.each{println it}
    String.metaClass.methods.findAll{it.name.startsWith("to")}
  - Dynamically add methods
    String.metaClass.foobar = { println “foobar”}
    *demostrate swapCase example
  - Intercept
    - Override invokeMethod
    - Can implment Aspect Oriented Programming (AOP)

Chinese-English

- 产品 chǎn pǐn goods; merchandise; product
- 社会 shè huì society
- 社区 shè qū community
- 虚拟 xū nǐ fictitious; theoretical; virtual
- 文化 wén huà culture; civilization; cultural
- 俗语 sú yǔ local saying; idiom
- 成语 chéng yǔ proverb; idiom
- 本质 běn zhì essence; nature; innate character; intrinsic quality
- 仪式 yí shì ceremony
- 解析 jiě xī to analyze; to resolve; to parse
- 领域 lǐng yù domain; sphere; field; territory; area
- 界面 jiè miàn interface
- 脚手架 jiǎo shǒu jià scaffolding
- 竹子 zhú zi bamboo
- 建设 jiàn shè to build; to construct; construction; constructive
- 框架 kuàng jià framework
- 架构 jià gòu infrastructure; architecture; framework
- 费经 fèi jīng waste brain power
- 数据库 shù jù kù database
- 试行 shì xíng (v) try; test out sth