"Java to Groovy"

Introduction to Groovy Workshop
Gr8Conf US 2015

Allison F
@ErinWith2Ls
MoreThanAPipeline.org

==========
~~~~~~~~~~~~~~~~~~~~
==========

Exercise 2: Duck Typing

```
class Box {
  int weight
  String barcode
}

class Human {
  Box carrying

  void lift(List boxes) {
    carrying = boxes.pop()
  }
}

class Robot {
  Box carrying

  void lift(List boxes) {
    carrying = boxes.max {Box box ->
      box.weight
    }
    boxes.remove(carrying)
  }
}

List<Box> boxes =
    [new Box(weight: 5, barcode: '123'),
     new Box(weight: 7, barcode: '456'),
     new Box(weight: 3, barcode: 'ah7'),
     new Box(weight: 1, barcode: 'b45'),
     new Box(weight: 8, barcode: 'e37')]
```
~~~~~~~~~~~~~~~~~~~~

```
println "All boxes: ${boxes*.barcode}"

def employee = new Human()

employee.lift(boxes)
println "Carrying ${employee.carrying.barcode}"
println "Remaining boxes: ${boxes*.barcode}"

employee= new Robot()

employee.lift(boxes)
println "Carrying ${employee.carrying.barcode}"
println "Remaining boxes: ${boxes*.barcode}"
```

QUESTION 1
Is the employee a Robot or a Human?
a) Robot
b) Human
c) depends...

QUESTION 2
Which box is the Robot carrying?
a) e37
b) 456
c) 123

==========

~~~~~~~~~~~~~~~~~~~~

==========

Exercise 4: Operator Overloading

TASK:
* Create a class with a member variable
* Overload an operator to create a new object
~~~~~~~~~~~~~~~~~~~~

The operators that can be overloaded correspond to the following method names:

+ plus
- minus
* multiply
/ div

Example:

```
class Box {
  int weight

  Box plus (Box otherBox) {
    return new Box(weight: this.weight + otherBox.weight)
  }
}

Box box1 = new Box(weight: 4)
Box box2 = new Box(weight: 8)

Box biggerBox = box1 + box2
println biggerBox.weight
```

==========

~~~~~~~~~~~~~~~~~~~~

==========

Collection method examples

EACH

* This method performs the closure on each element.
* The original Collection is returned

```
class Duck {
  String myName
  void fly() {
    println “${myName} flies!” }
  }
}
```
~~~~~~~~~~~~~~~~~~~~

```
List petDucks = [new Duck(myName: ‘Harold’),
                 new Duck(myName: ‘Sandy’),
                 new Duck(myName: ‘Samuel’)]

petDucks.each {
  it.fly()
}
```

COLLECT

* This method collects the results of the closure operation
* A new List is returned

```
class Box {
  int weight
}

List emptyBoxes = [new Box(weight: 3),
                   new Box(weight: 17),
                   new  Box(weight: 5)]

List boxWeights = emptyBoxes.collect {
                    it.weight
                  }
```

SPREAD *.

* This operator calls the method coming after the dot on each element
* Returns a new List with the results of the method calls

This:
```
emptyBoxes.collect { it.weight }
```

can also be written as:
```
emptyBoxes*.weight
```

==========

~~~~~~~~~~~~~~~~~~~~

==========
~~~~~~~~~~~~~~~~~~~~

Exercise 5: Collections

```
class Duck {
  String name
}
List ducks = [ new Duck(name: ‘Harold’),
               new Duck(name: ‘Sandy’),
               new Duck(name: ‘Samuel’)]

List names = []
ducks.each {
  names += it.name
}
```

A) What is returned?
B) What is in the names List?
C) Rewrite the 'each' statement using collect
D) Rewrite the 'each' statement using *.

==========

~~~~~~~~~~~~~~~~~~~~

==========

Exercise 6: Reflection

```
package us.gr8conf.workshop

class Box {
    int weight
    String barcode
}

class Human {
    public Box carrying
    public String name

    void lift(List boxes) {
        carrying = boxes.pop()
    }
}

println "Human class package: ${Human.package}"
println 'Fields in Human class:'
Human.fields.each { println "* ${it}" }
println 'Methods in Human class:'
Human.methods.each { println "* ${it}" }
```
~~~~~~~~~~~~~~~~~~~~

==========
~~~~~~~~~~~~~~~~~~~~
==========

Installing Java: bit.ly/downloadjdk7

Set $JAVA_HOME environment variable

On Mac: export JAVA_HOME=`/usr/libexec/java_home -v 1.7`
(in .bashrc, then source .bashrc)

On Windows: setting environment variables is different in each version

---

Installing Groovy:

Mac/Linux/Cygwin:

> curl -s get.gvmtool.net | bash
> gvm install groovy

Windows:
Installation instructions: http://bit.ly/1Vruzvl

==========
~~~~~~~~~~~~~~~~~~~~
==========

Exercise 7: From Java to Groovy

7.1) Write the Java code, compile it, and run it

Flier.java:

```
public interface Flier {
    public String fly();
}
```

Bird.java:

```
public class Bird implements Flier {

    int quacks = 0;

    public String fly() {
        return "  Flap, flap, flap";
    }

    public void quack() {
        quacks++;
        System.out.println("Quack!");
    }
}
```

Plane.java:

```
import java.util.Date;

public class Plane implements Flier {

    Date takeoffTime;
    Date landingTime;

    public String fly() {
        takeoffTime = new Date();
        return "  This is your captain speaking, we have
taken off";
    }

    public void land() {
        landingTime = new Date();
    }
}
```

Superman.java:

```
public class Superman implements Flier {

    int numRescues;

    public String fly() {
```

```
        return "  Dun, dun, dah!";
    }

    public boolean rescue() {
        numRescues++;
        return true;
    }
}
```

LookUpInTheSky.java:

```
public class LookUpInTheSky {

    public static void main(String[] args) {

        Bird bird = new Bird();
        Plane plane = new Plane();
        Superman clark = new Superman();

        Flier[] fliers = {bird, plane, clark};

        System.out.println("Look, up in the sky!");

        for (int i = 0; i < fliers.length; i++) {
            System.out.println("It's a " +
                  fliers[i].getClass().getName());
            System.out.println(fliers[i].fly());
        }
    }
}
```

Compile the code:

javac *.java

Run the code:

java LookUpInTheSky

7.2) Compile and run with Groovy

Compile the code:

groovyc -j *.java

Run the code:

java LookUpInTheSky

7.3) Code with Groovy

Change the extension of LookUpInTheSky to LookUpInTheSky.groovy

Recompile:

groovyc -j *.java *.groovy

Run it:

Mac:

java -cp $GROOVY_HOME/lib/groovy-2.4.3.jar:. LookUpInTheSky

Windows:

java -cp "%GROOVY_HOME%\lib\groovy-2.4.3.jar:." LookUpInTheSky

---

7.3.a) New features, coded in Groovy

Add another Flier… But make it a Groovy class and use Duck Typing instead of implementing the interface

Example: Spaceship

* Spaceship should not implement the `Flier` interface

* Spaceship needs a `String fly()` method

* Change `Flier[]` to `List` in LookUpInTheSky. Change `fliers.length` to `fliers.size`

* Add a Spaceship object to fliers

Compile and run

7.3.b) Tech Debt: Doing things “the Groovy way”

* Replace `System.out.println` with `println`

* Replace String concatenation with Groovy Strings
  eg, `"Hello ${x}"`

* Replace the for loop with .each and a Closure
  eg, `[new Duck(), new Plane()].each { println it }`

* User “Groovier” reflection
  eg, `println Duck.class.name` prints class name

* Use implicit returns
  explicit use of the return statement is not required